i.MX28 Applications Processor Reference Manual

Document Number: MCIMX28RM
Rev. 1, 2010

Contents

Section Number | Title | Page

Section Number Title Page

Section Number **Title** **Page**

Section Number Title Page

Section Number Title Page

Section Number Title Page

Section Number	Title	Page

Section Number **Title** **Page**

Section Number Title Page

Section Number **Title** **Page**

Section Number Title Page

Section Number	Title	Page

Section Number | Title | Page

Section Number | Title | Page

Chapter 14
External Memory Interface (EMI)

Section Number	Title	Page

Section Number	Title	Page

Section Number	Title	Page

Section Number	Title	Page

Section Number **Title** **Page**

Section Number	Title	Page

Chapter 19
Digital Control (DIGCTL) and On-Chip RAM

Section Number Title Page

Section Number Title Page

Section Number | Title | Page

Chapter 25
Controller Area Network (FlexCAN)

Section Number **Title** **Page**

Section Number	Title	Page

Section Number | Title | Page

Section Number | Title | Page

Section Number **Title** **Page**

Section Number	Title	Page

Section Number | Title | Page

Chapter 31
USB High-Speed On-the-Go Host Device Controller

Section Number | Title | Page

Section Number Title Page

Section Number | Title | Page

Chapter 1
Product Overview

1.1 i.MX28 Overview

The i.MX28 is a low-power, high performance application and a multi-media processor aimed at the following market segments:

- Human Machine Interface (HMI) panels: industrial, home.
- Industrial drive, PLC, I/O control display, factory robotics display.
- Graphical remote controls.
- Handheld scanners and printers.
- Electronic point-of-sale (POS) terminals.
- Smart energy gateways/meters.
- Media gateways.
- Portable medical devices.
- Media phones, VoIP.
- Automotive infotainment.

1.2 Hardware Features

The Hardware Features of the i.MX28 are as follows:

- ARM926 CPU Running at greater than 450 MHz at 1.45 V
  - Integrated ARM926EJ-S CPU
  - 32-Kbyte data cache and 16-Kbyte instruction cache
  - ARM Embedded Trace Macrocell (ETM CoreSight 9)

    - Parallel JTAG interface
    - Resistor-less boot mode selection using integrated OTP values

- 128 Kbytes of Integrated Low-Power On-Chip RAM
- 128 Kbytes of Integrated Mask-Programmable On-Chip ROM
- High Performance and Flexible External Memory Interface (EMI)
    - Supports DDR2-400 (1.8 V), LP-DDR1-400 (1.8 V) and LV-DDR2-400 (1.5 V)
    - 16-bit data-width with up to two chip-selects
    - Rich arbitration features and high-performance AXI & AHB bus interface
    - Integrated ODT and control for DDR2 applications
    - Up to 200 MHz DDR clock frequency with voltage over drive
- 1280 bits of On-Chip One-Time-Programmable (OCOTP) ROM
- Two 802.3 compatible 10/100 Ethernet MAC supporting IEEE® 1588 protocols and one 3-port L2 switch
- Two FlexCAN2 interfaces supported
    - Full Implementation of the CAN protocol specification, Version 2.0B
    - 64 Message Buffers of zero to eight bytes data length
    - Low power modes, with programmable wake up on bus activity

- Two instances of Universal Serial Bus (USB) High-Speed — Up to 480 Mb/s
    - One USB 2.0 on-the-go (OTG) device/host and one USB 2.0 host
    - Fully integrated high/full/low-speed/ Physical Layer Protocol (PHY)
    - OTG capable (uncertified by USB-IF)

- Power Management Units
    - Single DC-DC switched converter supporting Li-Ion batteries.
    - Features multi-channel outputs for VDDIO (3.3 V), VDDD (1.2 V), VDD1P5 (1.5 V), VDDA (1.8 V) and regulated 4.2 V source

    - 1.5 V linear regulator with programmable current limits for load and battery charge circuits
    - Silicon speed and temperature sensors enable adaptive power management over temperature and silicon process
    - Integrated temperature based power shut-down safety modes

- Optimized for very long battery life
- 16-Channel Low-Resolution ADC
    - Seven independent channels and nine dedicated channels
    - 4/5 wires resistive touchscreen controller
    - 8 × 8 key pad panel matrix
    - Temperature sensor controller
    - Absolute accuracy of 1.3%
    - Up to 0.5% with bandgap calibration
- Single Channel High Speed ADC (HSADC)
    - Up to 2 Msps sample rate
    - Sample precision configurable for 8/10/12 bits
    - Three trigger modes supporting to start ADC conversion
- Security Features
    - Read-only unique ID for digital rights management algorithms
    - Secure boot using 128-bit AES hardware decryption
    - SHA-1 and SHA-256 hashing hardware
    - Customer-programmed (OTP) 128 bit AES key is never visible to software
    - High Assurance Boot (HAB4)
- Wide Assortment of External Media Interfaces
    - Up to eight NAND Flash memories with hardware management of device interleaving

  - High-speed MMC, secure digital (SD)
  - Hardware BCH ECC engine allowing for up to 20-bit correction with programmable NAND page layout.

- Dual Peripheral Bus Bridges with 32 DMA Channels
  - Multiple peripheral clock domains to save power while optimizing performance
  - Direct Memory Access (DMA) with sophisticated linked DMA command architecture saves power and off-loads the CPU

- Highly Flexible Display Controller
  - Up to 24-bit RGB (DOTCK) modes
  - Up to 24-bit system-mode including VSYNC and WSYNC modes
  - 8-bit data ITU-R/BT.656 D1 digital video stream output mode (PAL/NTSC), with on-the-fly RGB to YCbCr color-space-conversion
  - Flexible input formats

- Pixel Processing Pipeline (PXP)
  - Provides full path from color-space conversion, scaling, alpha-blending to rotation without intermediate memory access
  - Bi-linear scaling algorithm with cropping and letterboxing with up to 4:1 downscaling (720p to QVGA for example)
  - Alpha-blend, color-keying
  - Memory efficient block-based rotation engine
  - Supports up to eight overlays
  - 8 × 8 and 16 × 16 programmable block size for DRAM bus-width matching resulting in optimized efficiency.

- Data Co-Processor (DCP)
  - AES 128-bit encryption/decryption
  - SHA-1 and SHA256 hashing

- High-speed memory copy
- Bit Blit

- Six Universal Asynchronous Receiver-Transmitters (UARTs)
  - Five high-speed application UARTs operating up to 3.25 Mb/s with hardware flow control, dual DMA and auto baud-rate detection
  - Debug UART operates at up to 115 Kb/s using programmed I/O

- Two I^{2}C Master/Slave Interfaces
  - Up to 400 Kb/s
  - DMA control of an entire EEPROM or other device read/write transaction without CPU intervention
  - PIO and PIO queue modes controlled by CPU. In PIO queue mode, commands and data are queued up with FIFOs
  - Both 7-bit and 10-bit device address supported in master mode. Programmable 7-bit address employed in slave mode

- Four Synchronous Serial Ports (for SPI, MMC, SDIO, Triflash)
  - Up to 52 MHz external SSP clock for SD/MMC mode
  - 1-bit, 4-bit and 8-bit MMC/SD/SDIO modes
  - Compliant with SDIO Rev. 2.0
  - Support eMMC4.3 and eMMC4.4
  - SPI with single, dual and quad modes

- Four-Channel 32-Bit Timer with Rotary Decoder
- Eight-Channel Pulse Width Modulator (PWM)
- Real-Time Clock
  - Alarm clock can turn the system on
  - Uses the existing 24-MHz XTAL for low cost or optional low power crystal (32.768 KHz or 32.0 KHz), customer-selectable through OTP

- SPDIF Transmitter
- Dual Serial Audio Interface (SAIF), Three Stereo Pairs
  - Full-duplex stereo transmit and stereo receive operations
  - Cell phone baseband processor connection and external ADCs and DACs
  - Bluetooth hands-free connection (with full-duplex voice)
  - Analog I/O for peripheral bus breakouts
  - I^2S, left-justified, right-justified, PCM, and non-standard formats
- Customer-Programmable One-Time-Programmable (OTP) ROM through Integrated eFuse Block
  - Resistorless boot mode selection
  - 128-bit boot mode crypto key
  - Boot mode specification of NAND characteristics for device that the customer is soldering to the board. This means no more costly delays waiting for new device support in the boot ROM.
  - Fully software-programmable and accessible
- Flexible I/O Pins
  - All digital pins have drive-strength controls as described in Pin Drive Strength Selection.
  - Almost all non-EMI digital pins have general-purpose input/output (GPIO) mode
- Offered in 289-Pin Ball Grid Array (BGA)

1.3 i.MX28 Product Features

The following figure shows the block diagram of a typical system based on the i.MX28.

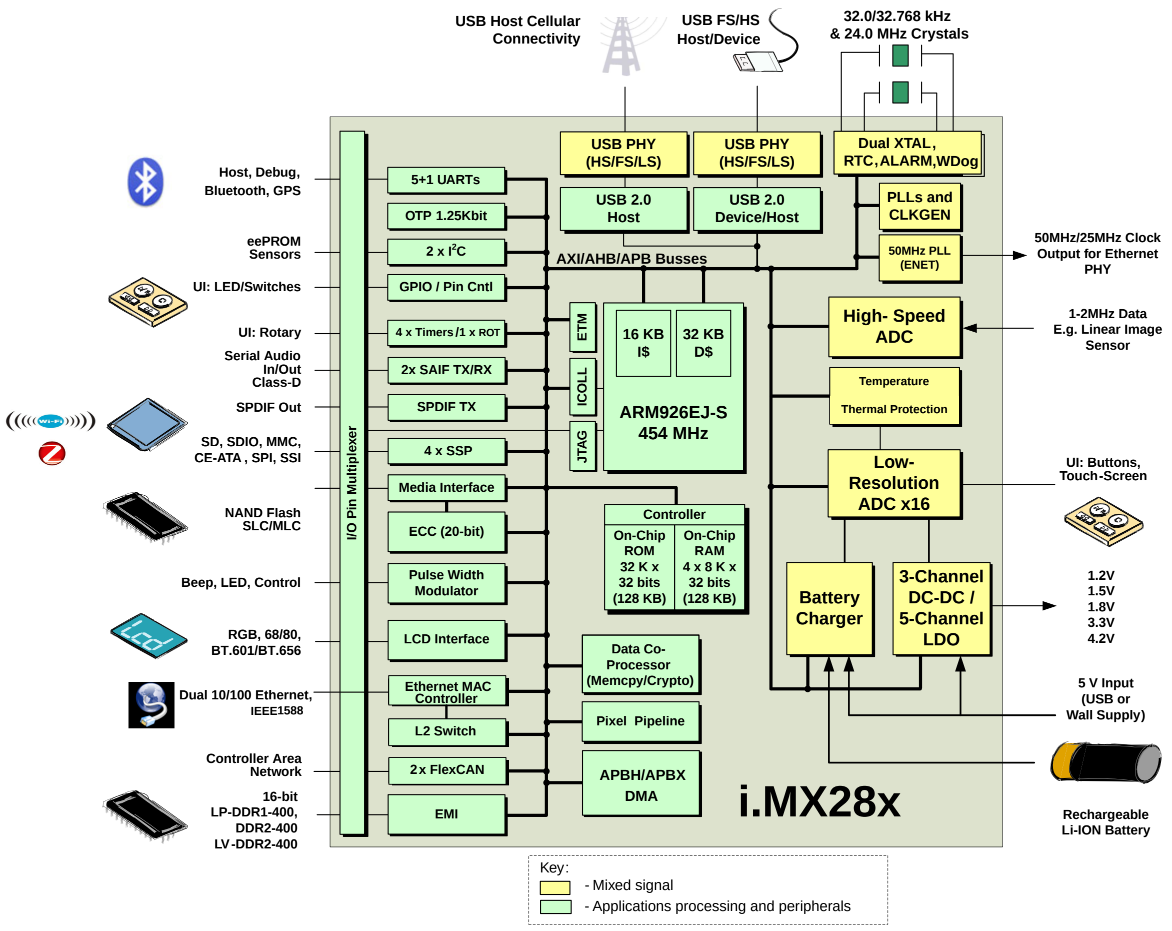


Figure 1-1. System Block Diagram

The i.MX28 features low power consumption to enable long battery life in portable applications. The integrated power management unit includes highly-efficient on-chip DC-DC converters. The power management unit also includes an intelligent battery charger and detector for Li-Ion cells and is designed to support Adaptive Voltage Control (AVC), which can reduce the system power consumption by half. AVC also allows the chip to operate at a higher peak CPU operating frequency than typical voltage control systems. The DC-DC converters and the clock generators can be reprogrammed on-the-fly to trade off power versus performance dynamically.

To provide the maximum application flexibility, the i.MX28 integrates a wide range of I/O ports. It can efficiently interface to nearly any type of flash memory, serial bus, LCD or HDMI/MHL transmitter. It is also ready for advanced connectivity applications such as Bluetooth, WiFi and Cellular Data Model through its integrated 4-bit SDIO controller, high-speed (3.25 Mb/s) UARTs and secondary USB host PHY.

An ARM926EJ-S CPU with 16K I$ & 32K D$ L1-caches and 128 Kbytes of on-chip SRAM provides the processing power needed to support advanced features such as web browsing and portable gaming (together with graphics and video processing hardware). Contact the local Freescale representative for more information on the software board support packages available for the i.MX28.

Execution always begins in on-chip ROM after reset, unless overridden by the debugger. A number of devices are programmed only at initialization or application state change, such as DC-DC converter voltages, clock generator settings and so on. Certain other devices either operate in a crystal clock domain or have significant portions that operate in a crystal clock domain, for example, ADC, PLLs and so on. These devices operate on a slower speed asynchronous peripheral bus. Write posting in the ARM core, additional write post buffering in the peripheral AHB, and set/clear operations at the device registers make these operations efficient.

1.3.1 ARM9 CPU Subsystem

The on-chip RISC processor core is an ARM926EJ-S CPU. This CPU implements the ARM v5TE instruction set architecture. This CPU has two instructions sets, a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in the Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two states. This enables optimization for both code density and performance. ARM studies indicate that Thumb code is typically 65% the size of the equivalent ARM code, while providing 160% of the effective performance in a constrained memory bandwidth applications. The ARM CPU is described in Overview.

The ARM RISC CPU is the central controller for the entire SOC, as shown in System Buses.

- AHB1—Used for ARM instruction fetches
- AHB2—Used for ARM data load/stores

1.3.2 System Buses

The i.MX28 uses buses based on ARM's Advanced Microcontroller Bus Architecture (AMBA) for the on-chip peripherals. The AMBA2 specification (http://www.arm.com/products/solutions/AMBA_Spec.html) outlines two bus types: AHB and APB. The AMBA3 specification (http://www.arm.com/products/solutions/axi_Spec.html) additionally outlines the AXI fabric. The three bus types are explained as follows:

- AXI is the highest-performance AMBA bus that supports de-coupled R/W channels, multiple outstanding transactions, and out-of-order data capability. This leads to higher performance and more efficient use of external memory.
- AHB is a higher-performance bus that supports multiple masters such as the CPU and DMA controllers.
- The APB is a lower-speed peripheral bus.

As shown below, the i.MX28 uses a three-layer AHB, one AXI 64bit bus and two APB buses (APBH and APBX).

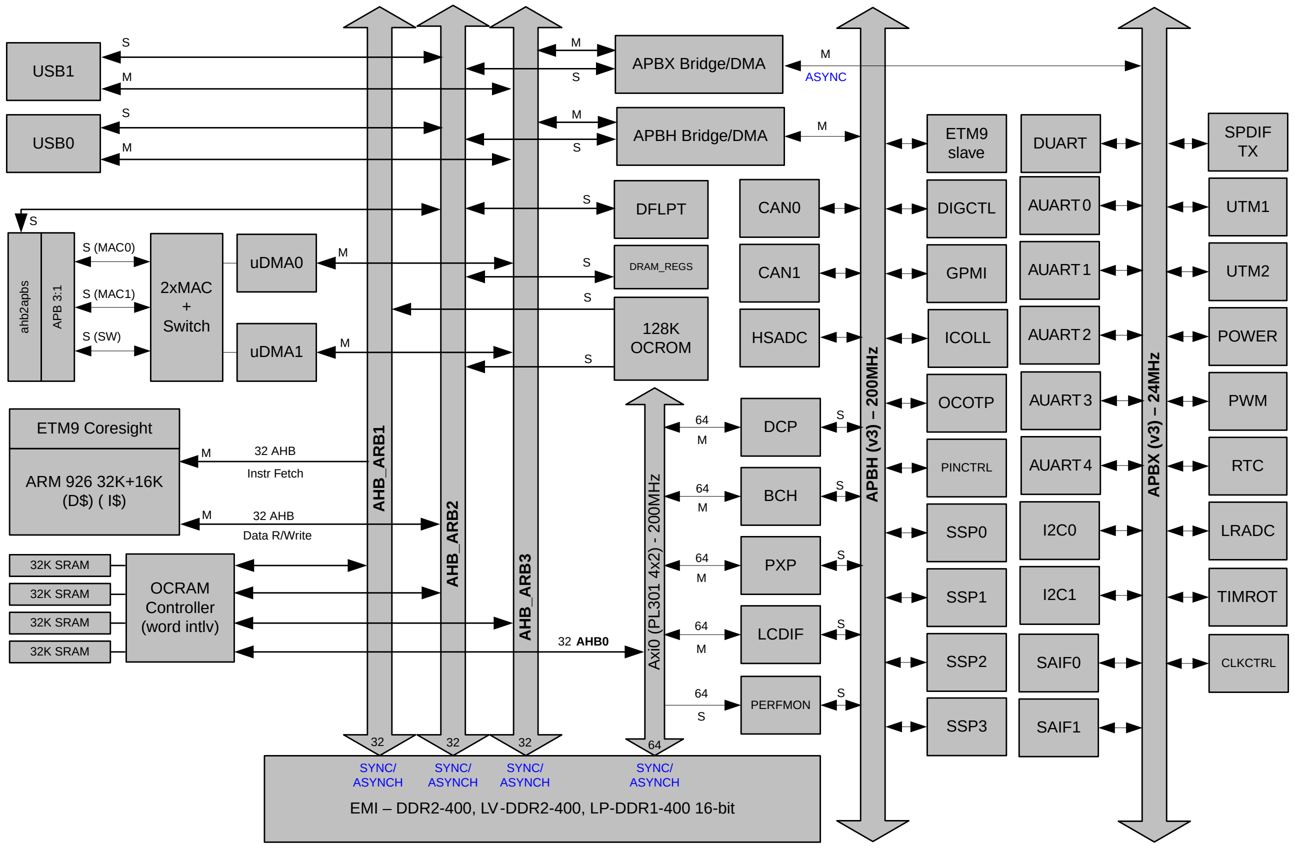


Figure 1-2. i.MX28 SOC System Buses

1.3.2.1 AXI Bus Segments

The AXI0 bus-segment on the i.MX28 provides several high-bandwidth/performance-critical peripherals, a tightly-coupled and efficient interface to Port-0 of the external memory controller. The peripherals are as follows:

- DCP (Crypto/Memcpy)

- Pixel Processing Pipeline (PXP)
- Display Controller (LCDIF)
- BCH-ECC Engine

i.MX28 also contains one bus performance monitor modules (PERFMON). There is one PERFMON per AXI segment. The key features of the PERFMON are as follows:

- Real-time performance statistics collection on three AXI buses
- Single or multiple master transactions monitoring
- Various interrupts support: address trap, latency threshold and so on
- AXI protocol and out-of-order sequence support
- Parameterized performance monitor for scalability

1.3.2.2 AHB Buses

The AHB is the main high-performance system bus and is implemented in three layers, as follows:

- Layer 1 (AHB1)—CPU instruction access to OCRAM, OCROM
- Layer 2 (AHB2)—CPU data access to OCRAM, OCROM, all bridges, Ethernet, USB and DFLPT slaves
- Layer 3 (AHB3)—APBH DMA, APBX DMA, Ethernet and USB masters

The ARM926 instruction bus (AHB1) is a single-master layer as the ARM926 data bus (AHB2). The AHB3 has multiple masters, as shown in Figure 1-2.

The ARM926 data bus connects to all the slaves in the system, including RAMs, ROMs, bridge slaves, Ethernet and USB slaves. The APB peripherals can act as AHB slaves through the AHB-APB bridge. The AHB has nine slaves:

- USB slave
- Ethernet slave
- On-chip RAM
- On-chip ROM

- Default first-level page table
- Two APB bridges

Each layer of the AHB bus allows one active transaction at a time. A transaction is initiated by a master, controlled by an arbiter, and serviced by the slave at the corresponding address. A transaction can be as short as a single byte, or as long as a CPU cache line (32 bytes). For the USB, a transaction can be much longer, up to 512 bytes.

AHB bus clock is named as the abbreviated "HCLK" or "clk_h" in the reference manual.

1.3.2.3 APB Buses

There are two APB buses on the i.MX28:

- APBH - The APBH (H DMA) is an AMBA2 APB system peripheral bus. This bus comprises of the APBH peripherals such as those accessed through the APBH DMA engine. The APBH peripherals are typically high-speed I/Os.
- APBX - The APBX (X DMA) is an AMBA2 APB system peripheral bus. This bus is asynchronous to other system buses. This bus comprises of the APBX peripherals such as those accessed through the APBX DMA engine. The APBX peripherals are typically low speed I/Os or analog control related.

The “H” in APBH denotes that the APBH is synchronous to AHB's HCLK, as compared to APBX, which runs on the crystal-derived XCLK. The APBX bus clock is named as the abbreviated "XCLK" or "clk_x" in the reference manual.

The ARM926 data bus connects to all the slaves in the system, including RAMs, ROMs, bridge slaves, USB and Ethernet slaves.

1.3.3 On-Chip RAM and ROM

The device includes 128 KB of on-chip RAM (OCRAM) implemented as four physical banks. It adopts a 4-port, word interleaved topology to maximize performance in a multi-layer bus system.

The device also includes 128 KB of on-chip mask-programmable ROM. The ROM contains initialization code written by Freescale Inc. to handle the initial boot and hardware initialization. Software in this ROM offers a large number of boot configuration options, including manufacturing boot modes for burn-in and tester operation.

Other boot modes are responsible for loading the application code from off-chip into the on-chip RAM. It supports initial program loading from a number of sources:

- NAND Flash devices
- SD2.0/MMC4.4, 4.3,4.2
- SPI from EEPROM devices or NOR Flash devices
- USB recovery mode
- I2C

At power-on time, the first instruction executed by the ARM core comes from this ROM. The reset boot vector is located at 0xFFFF0000. The on-chip boot code includes a firmware recovery mode. If the device fails to boot from NAND Flash, or hard drive, for example, the device will attempt to boot from a PC host connected to its USB port.

The ROM boot loader can be restricted to boot only properly certified load packages. This function is enabled by an OTP bit. Additional laser fuse bits select one of the 16 customer keys, which are further modified by laser fuses. A polynomial LSFR is used to decrypt the supplied boot package. A second polynomial computes an authentication code for the load package. If the decrypted package does not compute the correct authentication code, then the boot loader will enter recovery mode and attempt to boot from the USB device.

See Boot Modes for more information.

Memory Map Overview shows the memory map as seen by the processor. Any blank entries indicate that nothing is mapped at that address. No accesses should be made to these addresses because the results are indeterminate. The Decode Block column indicates the decode group to which each peripheral belongs. Most peripherals reside on the APBH, APBX and Axi0 (through AHB0).

1.3.4 On-Chip One-Time-Programmable (OCOTP) ROM

The device contains 1280 bits of One-Time-Programmable (OTP) ROM. The OTP is segmented into five distinct physical banks. Each bank is further divided logically into eight 32-bit words. The OTP serves several functions:

- Housing of hardware and software capability bits (copied into shadow registers).
- Housing of Freescale operations and unique-ID fields.
- Housing the customer-programmable cryptography key.
- Four words for customer general use.

- A 32-bit word is dedicated to controller read and write locking of the various OTP regions (copied into a shadow register).
- Storage of various ROM configuration bits.
- Storage of HAB data.

Access to the OTP is done through a memory-mapped APBH slave interface. Each of the 32 words is memory-mapped on APBH for the purpose of reading (requires a bank-opening sequence). Writing to the OTP is done through an address and data interface, where software provides the OTP word number (one of 32) and a programming mask.

See OCOTP Overview for more information.

1.3.5 External Memory Controller (EMI)

The i.MX28 contains a fully embedded DRAM controller solution including a multi-port AXI & AHB arbiter, control and PHY. The key features are as follows:

- Support for DDR2 (1.8 V), LP-DDR1 (1.8 V) and LV-DDR2 (1.5 V) up to 200 MHz clock rate (400 MHz data-rate).
- Support for up to 1024 MB with any combination of DRAMs up to two chip-enables.
- Fully pipelined command, read and write data interfaces to the memory controller.
- Advanced bank look-ahead features for high memory throughput.
- Front-end interface to three AHB ports and one AXI port with optimized buffering for high-bandwidth capability.
- A programmable register interface to control memory device parameters.
- Full initialization of memory on memory controller reset.
- Delay-Line for reliable data capture timing across process, temperature and all supported voltage ranges.
- Integrated On Device Termination (ODT) for DDR2 applications.
- Supports all levels of power modes for various device types.

See Overview for more information on the EMI.

1.3.6 Interrupt Collector

The i.MX28 contains a 128-bit vectored interrupt collector for the CPU's IRQ input and a separate non-vectored interrupt collection mechanism for the CPU's FIQ input. Each interrupt can be assigned to one of the four levels of priority. The interrupt collector supports nesting of interrupts that preempt an interrupt service routine running at a lower priority level. Each of the 128 interrupts is assigned its own 32-bit programming register and can be set for HW source IRQ, SW source IRQ or HW source FIQ.

See Interrupt Collector (ICOLL) Overview for additional information.

1.3.7 Default First-Level Page Table

The device contains a default first-level page table (DFLPT) implemented as an AHB slave. This device provides an economical way to present 16 Kbytes of L1 page table entry data to the ARM CPU's MMU. This page-table is connected to the AHB bus at base address 0x800C0000. The DFLPT provides 16 movable and scalable entries such that each entry can span up to128 locations. This allows the DFLPT to be used for systems with large external memories.

See Default First-Level Page Table (DFLPT) Overview for additional information.

1.3.8 DMA Controller

Many peripherals on the i.MX28 use direct memory access (DMA) transfers. Some peripherals, such as the USB controller, make high random accesses to the system memory for a large number of descriptor, queue heads, and packet payload transfers. This high random access nature is supported by integrating a dedicated DMA into the USB controller and connecting it directly to the high-speed AHB bus.

Similarly, the DCP (crypto/memcpy), BCH-ECC and LCD controller (legacy DMA mode) devices contain their own bus masters to allow more random accesses to the system memory.

Other peripherals have small number of high sequential transactions, for example the ADC streams, SPDIF transmitter, and so on. These devices share a centralized address generation and a data transfer function that allows them to share a single shared master on the AHB.

As mentioned previously, there are two AMBA peripheral buses on the i.MX28:

- The APBH bus runs at 200 MHz clock domain.
- The APBX bus runs in an independent XCLK clock domain that can be slowed down significantly for power reduction.

See AHB-to-APBH Bridge Overview and AHB-to-APBX Bridge Overview for more detailed information.

The two bridge DMAs are controlled through linked DMA command lists. The CPU sets up the DMA command chains before starting the DMA. The DMA command chains include set-up information for a peripheral and associated DMA channel. The DMA controller reads the DMA command, writes any peripheral set up, informs the peripheral to start running and then transfers data, all without CPU intervention. The CPU can add commands to the end of a chain to keep data moving without interventions.

The linked DMA command architecture off loads most of the real-time aspects of I/O control from the CPU to the DMA controller. This provides better system performance, while allowing longer interrupt latency tolerances for the CPU.

1.3.9 Clock Generation Subsystem

The i.MX28 uses several different clock domains to provide clocks to various subsystems, as shown in Figure 10-1. These clocks are either derived from the 24-MHz crystal or from one of the integrated high-speed PLLs. There are three PLLs in the i.MX28:

- PLL0 - 480 MHz with multiple phase-fractional and integer post-dividers used to clock the AXI and APB buses (except for APBX), I/O peripherals, multimedia/display hardware and the external memory interface. It also acts as the primary PLL for USB0.
- PLL1 - 480 MHz dedicated purely for USB1.
- PLL2 - 50 MHz dedicated to the Ethernet hardware with additional post-divider for lower frequencies.

See Clock Generation and Control (CLKCTRL) Overview for additional information about the system clock architecture.

The system also includes a real-time clock that can use either the 24-MHz system crystal or an optional low power crystal oscillator running at either 32.768 KHz or 32.0 KHz (customer-configurable through OTP). An integrated watchdog reset timer is also available for automatic recovery from an errant code execution.

See RTC Overview for more information about these features.

1.3.10 Power Management Unit

The i.MX28 contains a sophisticated Power Management Unit (PMU), including one integrated DC-DC converter, four linear regulators, one regulated 4.2 V output and battery chargers. The PMU can operate from a Li-Ion battery using the DC-DC converters or a 5-V supply using the linear regulators and can automatically switch between them without interrupting the operation. The PMU includes circuits for battery and system voltage brownout detection, as well as the on-chip temperature, digital speed, and process monitoring. In addition, there exist safety features such as thermal shut-down and battery charge current cut-off based on the external thermistor (through LRADC). The integrated PMU converter can be used to provide programmable power for the device as well as the entire application on up to five rails:

- VDD4P2 (nominal 4.2 V) -- when connected to 5 V source, can source the DC-DCs
- VDDIO (nominal 3.3 V) -- DC-DC or linear-regulator from 5 V
- VDDA (nominal 1.8 V) -- DC-DC or linear-regulator from VDDIO
- VDD1P5 (nominal 1.5 V) -- linear-regulator from VDDIO for LVDDR2
- VDDD (nominal 1.2 V) -- DC-DC or linear-regulator from VDDA

The 4.2 V regulated output also allows the programmable current limits:

- Total load plus battery charge current (5 V limit)
- Battery charge current
- Load current -- for both on-chip and off-chip circuits

The 4.2 V circuit is capable of adjusting distribution of current supply between the load and the battery-charger depending on the programmed current-limits and the load conditions. For example, when the battery charging exceeds the 5 V current limit, the 4.2 V regulator steals the current from the battery-charger circuit and diverts it to the load circuit. The converter can be configured to operate from the standard Li-Ion battery up to 4.2 V. These converters use off-chip reactive components (L/C) in a pulse-width or frequency-modulated DC-DC converter. The real-time clock includes an alarm function that can be used to wake-up the DC-DC converters, which then wakes up the rest of the system.

1.3.11 Ethernet Interfaces

The i.MX28 includes extensive Ethernet connectivity support. Ethernet controller includes two programmable 10/100 IEEE 802.3 Ethernet MACs. There is one time stamping module for each MAC to support Ethernet applications requiring precise timing references for incoming and outgoing frames to implement a distributed time synchronization protocol such as the IEEE 1588.

In addition to this, a hardware 3-port switch is supported for either redundancy (dual cables) or daisy-chaining (packet forwarding) to support automatic extension of control networks.

Two unified DMA blocks allows backward compatibility to legacy FEC interface (note that all uDMA programming is performed through the MAC0 and MAC1 register interfaces).

1.3.12 CAN Interfaces

The i.MX28 includes dual FlexCAN2 controllers which are compatible with the CAN 2.0B protocol specification [Ref. 1]. The CAN Protocol Interface (CPI) manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling. The Message Buffer Management (MBM) handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms.

1.3.13 USB Interfaces

The i.MX28 includes two high-speed Universal Serial Bus (USB) version 2.0 controllers and integrated USB Transceiver Macrocell Interface (UTMI) PHYs. The i.MX28 device interface can be attached to USB 2.0 hosts and hubs running in the USB 2.0 high-speed mode at 480 Mbits per second. It can be attached to USB 2.0 full-speed interfaces at 12 Mbits per second. Note that a dual-device configuration is not supported.

The USB controllers and integrated PHYs support high-speed Host modes for peer-to-peer file interchange. The USB controller can also be configured as a high-speed host.

The USB subsystem is designed to make efficient use of system resources within the i.MX28. It contains a random-access DMA engine that reduces the interrupt load on the system and reduces the total bus bandwidth that must be dedicated to service the five on-chip physical endpoints.

Each USB is a dynamically configured port that can support up to seven RX and seven TX endpoints besides EP0, each of which may be configured for bulk, interrupt, or isochronous transfers. The USB configuration information is read from on-chip memory through the USB controller's DMA.

See USB High-Speed OTG-capable USB Controller Overview and USB PHY Overview for more information.

1.3.14 General-Purpose Media Interface (GPMI)

The chip includes a general-purpose media interface (GPMI) controller that supports NAND devices (all packages).

The NAND Flash interface provides a state machine that provides all the logic necessary to perform DMA functions between on-chip or off-chip RAM and up to eight NAND Flash devices. The controller and DMA are sophisticated enough to manage the sharing of a single 8-bit wide data bus among eight NAND devices, without detailed CPU intervention. This allows the i.MX28 to provide unprecedented levels of NAND performance.

The general-purpose media interface can be described as two fairly independent devices in one. The three operating modes are integrated into one overall state machine that can freely intermix cycles to different device types on the media interface. There are eight chip selects on the media interface. Each chip select can be programmed to have a different type device installed.

The GPMI pin timings are based on a dedicated clock divider from PLL0, allowing the CPU clock divider to change without affecting the GPMI.

See General-Purpose Media Interface Overview for more information.

1.3.15 Hardware Acceleration for ECC for Robust External Storage

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the i.MX28. Modern high-density NAND Flash devices presume the existence of forward error-correction algorithms to correct some soft and/or hard bit errors within the device, allowing higher device yields and, therefore, lower NAND device costs.

The i.MX28 contains an Error Correction Code (ECC) hardware engine implementing the Bose Ray-Choudhury Hocquenghem algorithm for up to 20 bits of correction. The ECC engine is tightly coupled to the GPMI and has a dedicated programming model and a DMA structure.

1.3.15.1 Bose Ray-Choudhury Hocquenghem ECC Engine

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 20 single bit errors within a block of data not larger than about 900 bytes (512 bytes is typical) in applications such as protecting data and resources stored on modern NAND Flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2, 4, 8, 10, 12, 14, 16, 18, or 20 bit errors when writing into the flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as that was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF (2^{13} = 8192), which is the Galois Field consisting of 8191 one-bit symbols. BCH encoding (or encode for any block-code) can be performed by either of the two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block. These symbols are divided continuously by the generator polynomial for the GF (8192) and the resulting ***t*** parity symbols are appended to the block to create a BCH codeword (where ***t*** is the number of correctable bits).

The BCH sits on the AXI fabric with close coupling to both the GPMI and the external memory controller.

See BCH Overview for more information.

1.3.16 Data Co-Processor (DCP)—Memory Copy, Crypto

The device contains a data co-processor consisting of four virtual channels. Each channel is essentially a memory-to-memory copy engine. The linked list control structure can be used to move the byte-aligned blocks of data from a source to the destination. In the process of copying from one place to another, the DCP can be programmed to encrypt or decrypt the block using AES-128 in one of the several chaining modes. An SHA-1 or SHA256 hash can be calculated as part of the memory-copy operation.

See Data Co-Processor (DCP) Overview for more information.

1.3.17 I^{2}C Interface

The i.MX28 contains two two-wire SMB/I^{2}C bus interfaces. Each interface can act either as a slave or a master on the SMB interface. The on-chip ROM supports boot operations from I^{2}C mastered EEPROMs, as well as slave I^{2}C boot mode. I^{2}C flexibly supports three transaction modes: DMA, PIO and PIO queue modes.

See I2C Overview for more information.

1.3.18 General-Purpose Input/Output (GPIO)

The i.MX28 contains 124 GPIO pins in the 289-pin BGA package. Most digital pins (except for EMI pins) that are available for specific functions are also available as GPIO pins if they are not otherwise used in a particular application. All GPIO pins support both 1.8V and 3.3V VDDIO. See Pin Control and GPIO Overview for more information.

1.3.19 Display and Capture Processing

In order to support a variety of multimedia-rich use-cases, the i.MX28 contains a powerful and flexible display and capture sub-system comprised of the following components:

- Pixel Processing Pipeline (PXP)
- Display Controller (LCDIF)

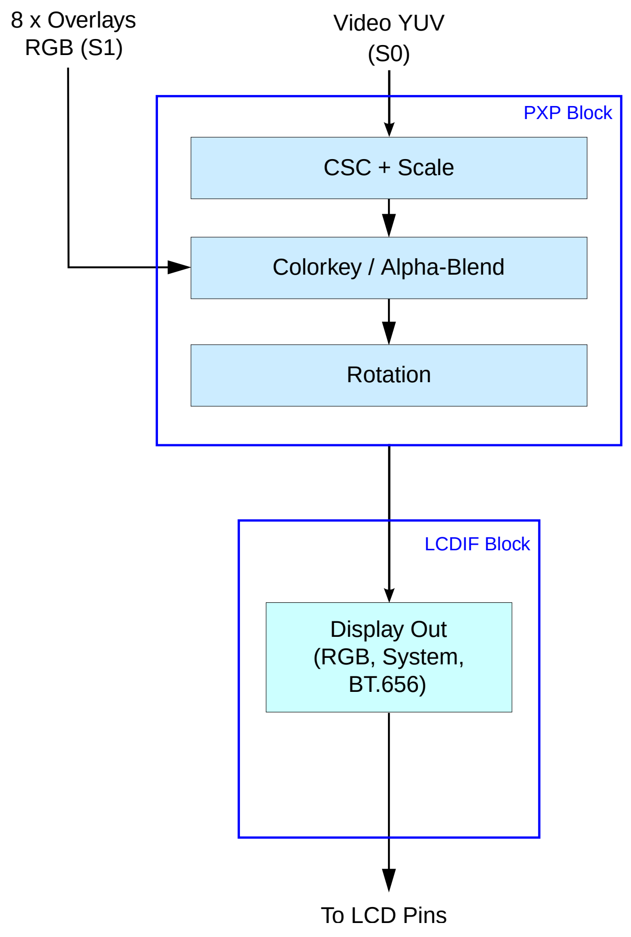


Figure 1-3. i.MX28 Display System

1.3.19.1 Display Controllers / LCD Interfaces (LCDIF)

The device contains the LCDIF display controller module which connects to the AXI bus. Some of the key features of the LCDIFs are as follows:

- Very flexible display support which allows for connection to a wide variety of display interfaces.
- AXI master on the AXI bus segment with large efficient request sizes and support for outstanding transfer requests.
- Ability to drive 24-bit RGB/DOTCK displays up to WVGA at 60 Hz. High robustness guaranteed by 512-pixel FIFO with under-run recovery.
- Support for full 24-bit system mode (8080/6080/VSYNC/WSYNC).

- ITU-R/BT.656 compliant D1 digital video output mode with on-the-fly RGB to YCbCr color-space-conversion. This output mode is suitable for driving the external TV-Encoder.
- Support for a wide variety of input and output formats that allows for conversion between input and output (for example, RGB565 input to RGB888 output). Also support for packed byte formats.

The APB DMA mode still exists for backward compatibility, but is not recommended for driving displays larger than QVGA (320 × 240).

See LCD Interface (LCDIF) Overview for more information.

1.3.19.2 Pixel Processing Pipeline (PXP)

The PXP performs all necessary post display frame pre-processing in hardware with minimal CPU overhead. The PXP operation and features can be described as follows:

- The PXP can be programmed to operate in either 8 × 8 or 16 × 16 block modes. When using 32-bit memories with the EMI, 16 ×16 block mode is recommended to allow for efficient memory access. Because the PXP accesses the blocks by requesting a line of block, a 16-pixel fetch (one line of a block) will make full use of a 16-byte burst on the EMI.
- The *background* image (for example, decoded video frames) is read from an external memory (single-plane YUV, dual-plane Y/UV and three plane Y/U/V inter-leaved source buffer formats are supported) into the internal buffers as either 8 × 8 or 16 ×16 pixel blocks. These buffers are then fed into a color-space converter (for example, YUV to RGB) followed by the scaling engine which utilizes an advanced bi-linear weighted scaling algorithm. The scaling operation is defined in terms of the output image (through programmable offsets and cropping registers). The output of the scaler is fed into yet another internal buffer called S0. If the background image is already in the RGB color-space, it is assumed to be scaled appropriately for the required output format and can thus be read directly into the internal S0 buffer. In order to maintain the efficient use of an external memory, only the relevant (visible) portion of the background image is fetched.
- The scaled RGB image (in the internal S0 buffer) can be blended with up to eight programmable *overlays*. The co-ordinates of the overlays can once again be described in terms of the resultant output image. Each overlay can have either a global programmable opacity or a per-pixel resolution if constructed with RGB color-space. In addition to this, each overlay can have a relative priority level such that when constructing the output image, the PXP only fetches the visible overlay in the current

8 × 8 or 16 ×16 block. The overlays are fetched into the internal S1 buffer. Alpha blending is performed on the S0 and S1 buffers to generate the blended output into the internal S3 buffer. Other operations such as BITBLT and color-keying can also be performed at this stage.

- The final stage of the PXP operation is the rotator which can perform flips and 90, 180 and 270 rotations. The rotator operates on the 8 × 8 or 16 ×16 pixel blocks in the S3 buffer to maximize external memory fetch efficiency. It writes 8 × 8 or 16 ×16 blocks to the external memory in this final stage.
- The PXP supports scaling down up to 4:1 in single-pass mode. This is useful for scaling 720p decoded content to a QVGA display. The PXP can also be used to further scale down images using a multi-pass approach which is enabled by the ability of the PXP to write YUV output.

See Pixel Pipeling (PXP) Overview for more information on the PXP.

1.3.20 SPDIF Transmitter

The device includes a Sony-Phillips Digital Interface Format (SPDIF) transmitter. It includes independent sample-rate conversion hardware so that the A/D, D/A and SPDIF can run simultaneously. The SPDIF has a dedicated DMA channel. The SPDIF has its own clock divider from the PLL.

See FlexCAN Introduction for more information.

1.3.21 Dual Serial Audio Interfaces

The i.MX28 SOC includes two Serial Audio Interfaces (SAIF), each with three stereo pairs. The pin multiplexing scheme for i.MX28 allows a stereo transmitter and a stereo receiver to be connected to external devices, either D/A and A/D converters or to a host processor, such as a cell phone or BlueTooth controller.

See Serial Audio Interface (SAIF) Overview for more information.

1.3.22 Rotary Decoder

An automatic rotary decoder function is integrated into the chip. Two digital inputs are monitored to determine which is leading and by how much. In addition, the hardware automatically determines the period for rotary inputs.

See Timers and Rotary Decoder (TIMROT) Overview for more information.

1.3.23 UARTs

There are six UARTs (similar to 16550 UART) provided; five for application use and one for debug. The application UARTs are high-speed devices capable of running up to 3.25 Mbits per second with 16-byte receive and transmit FIFOs. The application UARTs support DMA and flow control (RTS/CTS). The debug UART does not use DMA channels.

See Application UART Overview and Debug UART Overview for more information.

1.3.24 Low-Resolution ADC, Touch-Screen Interface and Temperature Sensor

The LRADC provides 16 physical channels of 12-bit resolution for analog-to-digital conversion. Only eight "virtual" channels can be used at one time, but those eight channels can be mapped to any of the 16 physical channels. Some physical channels have dedicated inputs:

- Channel 15—VDD5V
- Channel 14—Bandgap reference
- Channel 13—VDDD
- Channel 12—VDDA
- Channel 11—VTH
- Channel 10—Sample VDDIO
- Channel 8 and 9—Internal temperature sense input
- Channel 7—Battery

The USB_DN/DP inputs can only be sampled with the LRADC in non-USB mode (see HW_USBPHY_CTRL_DATA_ON_LRADC).

The remaining seven channels are available for other uses and can be used for resistive button sense, touch-screens or other analog input. Channels 0 and 1 have integrated drivers for external temperature monitor thermistors. Channels 2–6 have integrated drivers for resistive touch-screens. The LRADC provides typical performance of 12-bit no-missing-codes, 9-bit SNR, and 1% absolute accuracy (limited by the bandgap reference).

See LRADC Overview for more information.

1.3.25 High Speed ADC (HSADC) Controller

The high-speed ADC module is designed for driving the linear image scanner sensor (for example, TOSHIBA TCD1304DG linear image scanner sensor). It can also support some other general user cases which need to sample analog source with up to 2 Msps data rate and then move the sample data to the external memory. The high-speed ADC module integrates an 12-bit analog ADC module. This analog ADC module can support up to 2 Msps sample rate. In order to improve the flexibility, the high-speed ADC module can co-work with PWM module which can generate driving signals of external device such as the linear image scanner sensor. The PWM can also generate trigger signal which is synchronous with high-speed ADC module to start the conversion of ADC. An APBH-DMA channel is connected to the high-speed ADC module to move the sample data from the asynchronous FIFO inside the high-speed ADC module to the external memory.

1.3.26 Pulse Width Modulator (PWM) Controller

The device contains eight PWM output controllers that can be used in place of GPIO pins. Applications include LED brightness control, HSADC signal driving and high-voltage generators for electroluminescent lamp (EL) display backlights. Independent output control of each phase allows 0, 1, or high impedance to be independently selected for the active and inactive phases. Individual outputs can be run in lock step with guaranteed non-overlapping portions for differential drive applications.

See Pulse Width Modulator (PWM) Overview for more information.

Chapter 2
ARM CPU Complex

2.1 Overview

This chapter describes the ARM CPU included on the i.MX28 and includes sections on the processor core, the JTAG debugger, and the embedded trace macrocell (ETM) interface.

2.2 ARM 926 Processor Core

The on-chip Reduced Instruction Set Computer (RISC) processor core is an ARM926EJ-S CPU. This CPU implements the ARM v5TE instruction set architecture, which includes the enhanced DSP instructions.

The ARM9EJ-S has two instruction sets: a 32-bit instruction set used in the ARM state and a 16-bit instruction set used in the Thumb state. The core offers the choice of running in the ARM state or the Thumb state or a mix of the two. This enables optimization for both code density and performance. ARM studies indicate that Thumb code is typically 65% the size of an equivalent ARM code, while providing 160% of the effective performance in constrained memory bandwidth applications.

A block diagram of the ARM926EJ-S core is shown in Figure 2-1.

See http://www.arm.com/documentation/ARMProcessor_Cores/index.html to download the following ARM documentation on the ARM926EJ-S core:

- ARM926EJ-S Technical Reference Manual, DDI0198D
- ARM926EJ-S Development Chip Reference Manual, DDI0287A

The ARM9 core has a total of 37 programmer-visible registers, including 31 general-purpose 32-bit registers, six 32-bit status registers, and a 32-bit program counter, as shown in Figure 2-2. In ARM state, 16 general-purpose registers and one or two status registers are accessible at any one time. In privileged modes, mode-specific banked registers become available.

The ARM state register set contains 16 directly addressable registers, r0 through r15. An additional register, the Current Program Status Register (CPSR), contains condition code flags and the current mode bits. Registers r0–r13 are general-purpose registers used to hold data and address values, with r13 being used as a stack pointer. Register r14 is used as the subroutine link register (lr) to hold the return address. Register r15 holds the program counter (PC).

The Thumb state register set is a subset of the ARM register set. The programmer has access to eight general-purpose registers, r0–r7, the PC (ARM r15), the stack pointer (ARM r13), the link register (ARM r14), and the CPSR.

Exceptions arise whenever the normal flow of program execution has to be temporarily suspended, for example, to service an interrupt from a peripheral. Before attempting to handle an exception, the ARM core preserves the current processor state, so that the original program can resume when the handler is finished.

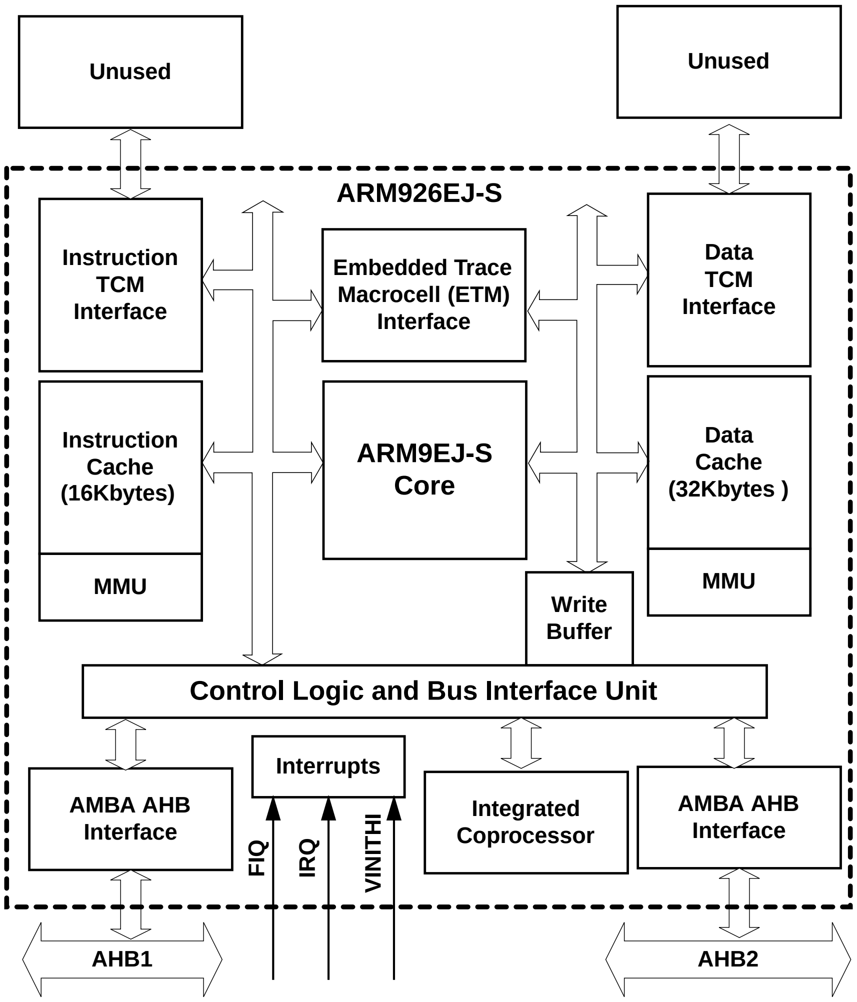


Figure 2-1. ARM926 RISC Processor Core

The following exceptions are recognized by the core:

- SWI—Software interrupt
- UNDEF—Undefined instruction
- PABT—Instruction prefetch abort
- FIQ—Fast peripheral interrupt
- IRQ—Normal peripheral interrupt
- DABT—Data abort
- RESET—Reset
- BKPT—Breakpoint

The vector table pointing to these interrupts can be located at physical address 0x00000000 or 0xFFFF0000. The i.MX28 maps its 64-Kbyte on-chip ROM to the address 0xFFFF0000 to 0xFFFFFFFF. The core is hardwired to use the high address vector table at hard reset (core port VINITHI =1).

The ARM 926 core includes a 16-Kbyte instruction cache and a 32-Kbyte data cache and has two master interfaces to the AMBA AHB, as shown below.

The i.MX28 always operates in a little-endian mode.

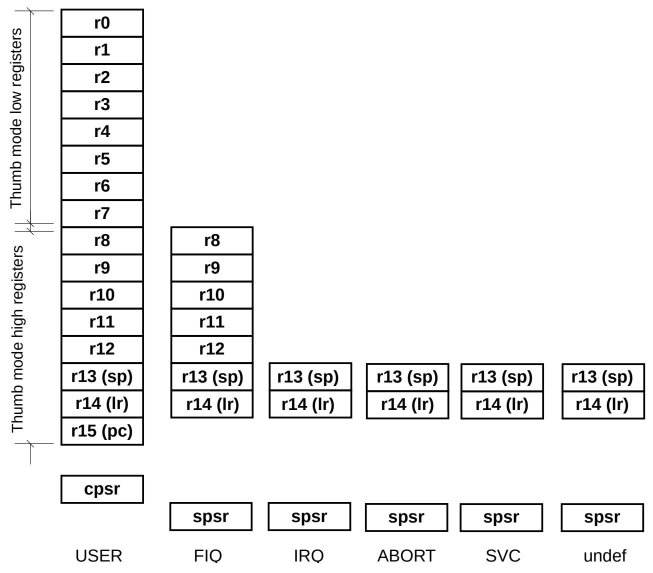


Figure 2-2. ARM Programmable Registers

2.3 JTAG Debugger

The TAP controller of the ARM core in the i.MX28 performs the standard debugger instructions.

2.3.1 JTAG READ ARM ID (TAP)

The TAP controller returns the following 32-bit data value in response to a JTAG READ ID instruction: 0x079264F3 when the pin DEBUG=1 (JTAG interface works for ARM debugging).

2.3.2 JTAG Hardware Reset

The JTAG reset instruction can be accomplished by writing 0xDEADC0DE to ETM address 0x70. The ETM is on scan chain 6. The bitstream is 0xF0DEADC0DE. The digital wide reset does not affect the DC-DC converters or the contents of the persistent registers in the analog side of the RTC.

2.3.3 JTAG Interaction with CPUCLK

Because the JTAG clock is sampled from the processor clock CPUCLK, there are cases in which the behavior of CPUCLK affects the ability to make use of JTAG. Specifically, the JTAG block will not function as expected if:

- CPUCLK is stalled due to an interrupt
- CPUCLK is less than 3x the JTAG clock
- CPUCLK is disabled for any reason

2.4 Embedded Trace Macrocell (ETM) Interface

The i.MX28 includes a stand-alone ARM CoreSight Embedded Trace Macrocell, ETM9CSSingle, which provides a instruction trace and a data trace for the ARM9 microprocessor. For more details, see the CoreSight ETM9 Technical Reference Manual. Also, see the pin list in the data sheet for pinout information.

Chapter 3
Default First-level Page Table (DFLPT)

3.1 Default First-Level Page Table (DFLPT) Overview

The DFLPT provides a unique method of implementing the ARM MMU first-level page table (L1PT) using a hardware-based approach. The ARM MMU L1PT must consist of 4096 page-table entries (PTE), each of which maps to a 1-Mbyte section of the 4-Gbyte system memory. Each PTE consists of a 32-bit descriptor, such that 16 Kbytes of memory is required to implement the L1PT. Using 16 Kbytes of system memory for the L1PT can be an issue for memory-constrained embedded systems (especially those without SDRAM).

The DFLPT implements a very sparse L1PT in hardware, as shown in Figure 3-1. This is achieved by having sixteen movable and expandable page table entries (MPTE) that are fully programmable and one semi-programmable fixed PTE. Any of the sixteen MPTEs can be bound to 4095 of the 4096 sections using sixteen locator registers in the DIGCTL block (MPTE0_LOC).

This implementation, although sparse, is very useful in low-memory applications. For small SDRAM systems (where the L1PT would typically be placed in SDRAM), the DFLPT provides significant speed and power advantages (as well as saving 16 Kbytes). For larger DRAM system, it provides a performance advantage. Large memory systems are accommodated through the spanning option (for example, in cases such as multimedia buffers, large amounts of memory are typically marked with the same attributes, and the memory is contiguous physically). Using the DFLPT, a level-one descriptor fetch takes two HCLK cycles to complete.

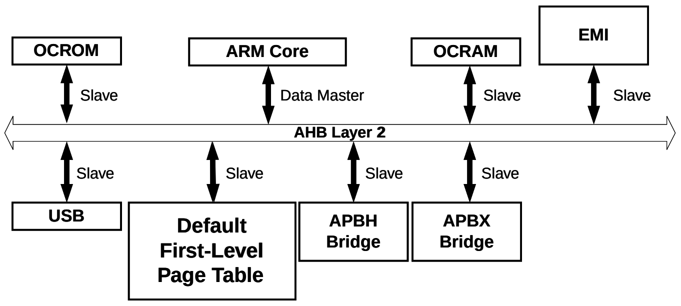


Figure 3-1. Default First-Level Page Table (DFLPT) Block Diagram

3.2 Operation

The DFLPT provides the following features as part of the memory management within the i.MX28 embedded software system:

- 16-Kbyte AHB slave located at addresses 0x800C_0000–0x800C_3FFF supports the required 4K L1 PTEs. To use the DFLPT, point the ARM TTB to 0x800C_0000.
  - Only 32-bit word accesses are supported.
  - All AHB burst types are supported.
  - Each access has a fixed 1-cycle AHB wait state.
  - Bus errors are supported when accessing an unbound PTE (that is, an address not bound to an MPTE or spanned MPTE).
- Sixteen fully programmable (value and location) page table entries. All 32-bits are programmable. The location/binding of each MPTE is determined through the HW_DIGCTL_MPTEn_LOC register fields in the DIGCTL module. The span (1-128 MB) of MPTE is determined through the HW_DIGCTL_MPTEn_SPAN register fields in the DIGCTL module. In addition, each of the sixteen entries can be disabled through HW_DIGCTL_MPTEn_DIS if less than sixteen entries are required (this avoids the need to bind an entry to some unused area of physical memory):
  - When an entry is disabled (through HW_DIGCTL_MPTEn_DIS), it cannot be bound, so programming its HW_DIGCTL_MPTEn_LOC field has no effect. This means that another entry which is not disabled (HW_DIGCTL_MPTEn_DIS=0) can have the same value for HW_DIGCTL_MPTEn_LOC as the disabled entry.
  - A read from an unbound PTE returns 0x0000_0000. When the MMU is enabled, this results in a section translation fault if the read is the result of a page-walk.
  - When using the span feature, the entire span is bound to the MPTE. This means that up to 128 32-bit word locations/addresses can be bound to any MPTE (see below for a more detailed explanation).
  - A write to an un-bound PTE shall result in a slave error, which shall in turn result in an ARM data-abort. Note that when writing to a PTE encompassed in a span, the descriptor will apply to the span of the MPTE with the exception of the base-address, which only applies to the base MPTE (at location LOC). Subsequent base addresses within a span are generated linearly from this LOC value (see below).
  - Each MPTE has a reset value of 0x0000_0000.

- One fixed PTE at location 2048 covers the i.MX28 PIO and register space. The location is fixed as virtual = real, is non-cacheable, and includes programmable bufferable, domain, and AP fields.
- Each MPTE requires a location register, such that it can be mapped to any of the 4K L1 section descriptors. In addition, each of these location registers can be programmed to be bound to up to 128 locations each (covering up to 128 MB of physical each). These location registers (HW_DIGCTL_MPTEn where n = 0...15) are located in the DIGCTL module. Refer to MPTE0_LOC for a description of the MPTEn_LOC registers.
- Each HW_DIGCTL_MPTEn_LOC must be programmed with a 12-bit value that corresponds to one of the 4K section entries. Once programmed, the AHB physical address of the MPTE is determined as:

  0x800C_0000 + (HW_DIGCTL_MPTEn_LOC << 2)
  HW_DIGCTL_MPTEn_LOC = 0x000 – 0xFFF

  - The reset state of each HW_DIGCTL_MPTEn_LOC register is n (n = 0...15), for example, HW_DIGCTL_MPTE3_LOC has reset value of 0x0000_0003.
  - None of the MPTEn_LOC registers should be programmed to 0x800 (2048). This corresponds to the fixed entry that covers the i.MX28 register space.
  - No checking/status is given for incorrect programming (for example, overlap). If multiple MPTEs are programmed to the same address in the DFLPT, the behavior is non-deterministic. The only exception is that all but one of the entries has DIS=1 set (disabled).

- The spanning feature can be enabled by adjusting the value of HW_DIGCTL_MPTEn_SPAN. This is a 3-bit value which determines the size of the span as 2^SPAN. This means that any access to the L1PT within the span will have the following properties:
  - The span allows any MPTE to bind to a range of MPTEn_LOC * 4 through to (MPTEn_LOC + (2^SPAN – 1)) * 4, where SPAN is an integer from 0 to 7.
  - Because spanning binds a single MPTE to multiple L1PT entries, the DFLPT must generate unique base addresses for each spanned entry, otherwise each 1 MB section would map to the same physical section. The DFLPT achieves this by assuming contiguous 1 MB section addressing for all sections covered within a span.

3.2.1 Top-Level Symbol and Functional Overview

The DFLPT connects to the AHB as shown in Figure 3-2. Note that due to the internal pipelining (single wait-state), the bus interfaces have little or no combinatorial logic. The read and address paths connect directly to the registers. The write-path connects to the register bank through one level 1-hot muxing. All address and locator decodes are also performed using 1-hot logic. Offset address generated from the AHB address (for purposes of generating spanned base addresses) use a registered version of the AHB address.

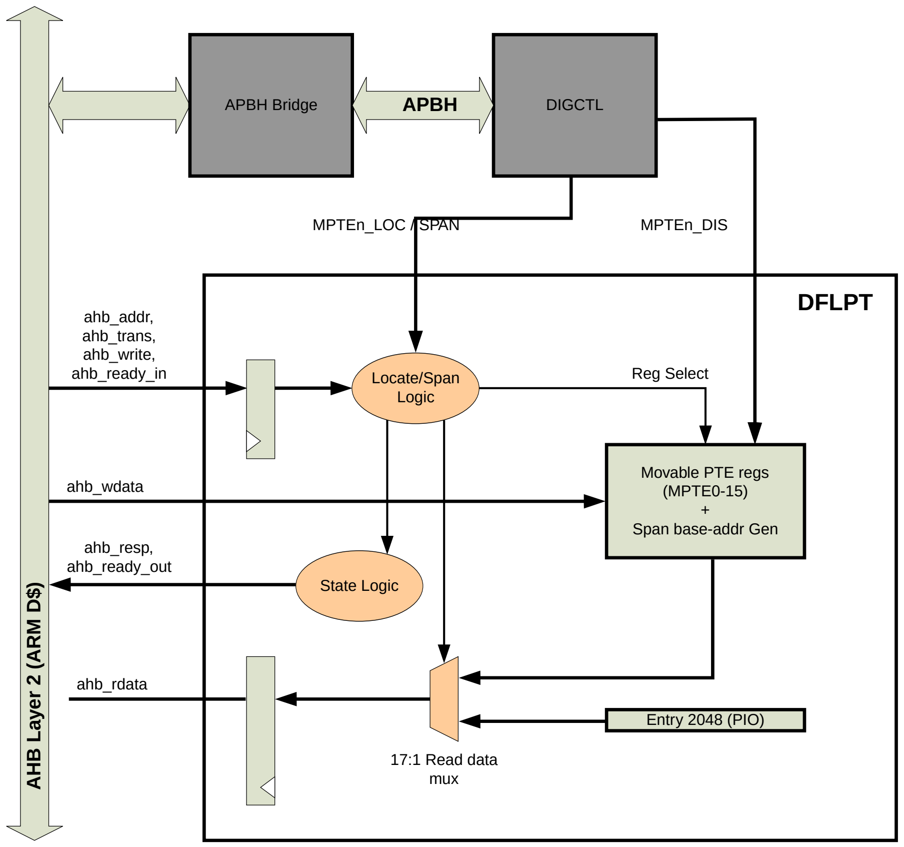


Figure 3-2. DFLPT System Level Block Diagram

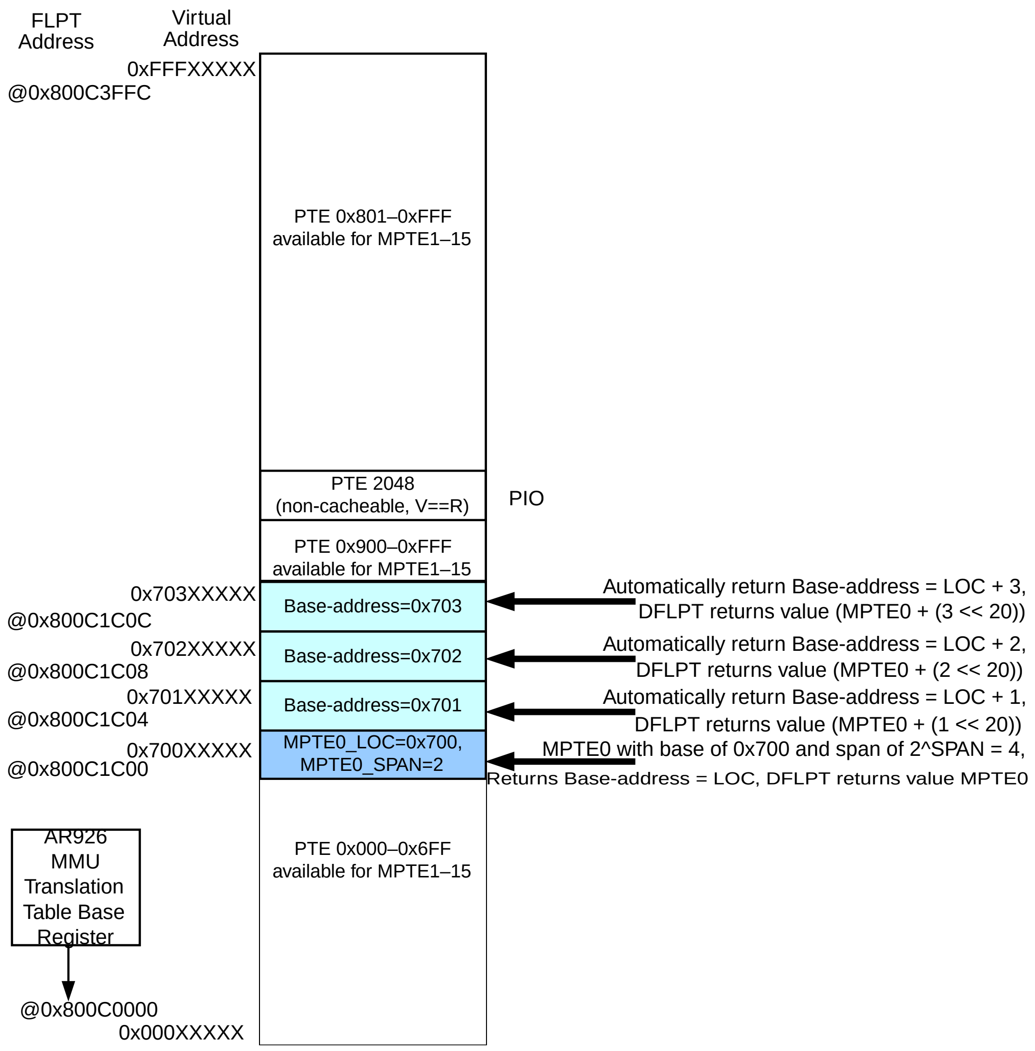
3.2.2 Memory Map

The virtual memory view of the DFLPT can be seen in the figure below. This example shows how a MPTE can be spanned and how it will appear to the ARM MMU. In this example, MPTE0 is set as follows:

- HW_DIGCTL_MPTE0_DIS = 0x0
- HW_DIGCTL_MPTE0_SPAN = 0x2

- HW_DIGCTL_MPTE0_LOC = 0x700
- TTB (0xBASE) = 0x800C_0000

Based on these settings, MPTE0 spans 2^2=4 entries in the page table. Because only one LOC value is specified, the DFLPT must generate section base-address values for each location in the span as a linear offset to the base LOC. Even though this represents some constraint in the view of virtual-to-physical memory mapping, it is quite common in an embedded system to have large chunks of virtual memory to be contiguous.



FLPT Address
Virtual Address
0xFFFXXXXX
@0x800C3FFC
PTE 0x801–0xFFF available for MPTE1–15
PTE 2048 (non-cacheable, V==R)
PIO
PTE 0x900–0xFFF available for MPTE1–15
0x703XXXXX
@0x800C1C0C
Base-address=0x703
Automatically return Base-address = LOC + 3, DFLPT returns value (MPTE0 + (3 << 20))
0x702XXXXX
@0x800C1C08
Base-address=0x702
Automatically return Base-address = LOC + 2, DFLPT returns value (MPTE0 + (2 << 20))
0x701XXXXX
@0x800C1C04
Base-address=0x701
Automatically return Base-address = LOC + 1, DFLPT returns value (MPTE0 + (1 << 20))
0x700XXXXX
@0x800C1C00
MPTE0_LOC=0x700, MPTE0_SPAN=2
MPTE0 with base of 0x700 and span of 2^SPAN = 4, Returns Base-address = LOC, DFLPT returns value MPTE0
AR926 MMU Translation Table Base Register
PTE 0x000–0x6FF available for MPTE1–15
@0x800C0000
0x000XXXXX

Figure 3-3. DFLPT Virtual Memory Map

The table below lists the page-table entries available in the DFLPT.

Table 3-1. Default First-Level Page Table

RE-GISTER NAME	ADDRESS	DESCRIPTION
MPTE15	0x800C0000 + (HW_DIGCTL_MPTE15_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE15_LOC + 2^SPAN)<<2)	Moveable PTE15 + ((Address[13:2] - HW_DIGCTL_MPTE15_LOC) << 20)
MPTE14	0x800C0000 + (HW_DIGCTL_MPTE14_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE14_LOC + 2^SPAN)<<2)	Moveable PTE14 + ((Address[13:2] - HW_DIGCTL_MPTE14_LOC) << 20)
MPTE13	0x800C0000 + (HW_DIGCTL_MPTE13_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE13_LOC + 2^SPAN)<<2)	Moveable PTE13 + ((Address[13:2] - HW_DIGCTL_MPTE13_LOC) << 20)
MPTE12	0x800C0000 + (HW_DIGCTL_MPTE12_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE12_LOC + 2^SPAN)<<2)	Moveable PTE12 + ((Address[13:2] - HW_DIGCTL_MPTE12_LOC) << 20)
MPTE11	0x800C0000 + (HW_DIGCTL_MPTE11_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE11_LOC + 2^SPAN)<<2)	Moveable PTE11 + ((Address[13:2] - HW_DIGCTL_MPTE11_LOC) << 20)
MPTE10	0x800C0000 + (HW_DIGCTL_MPTE10_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE10_LOC + 2^SPAN)<<2)	Moveable PTE10 + ((Address[13:2] - HW_DIGCTL_MPTE10_LOC) << 20)
MPTE9	0x800C0000 + (HW_DIGCTL_MPTE9_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE9_LOC + 2^SPAN)<<2)	Moveable PTE9 + ((Address[13:2] - HW_DIGCTL_MPTE9_LOC) << 20)
MPTE8	0x800C0000 + (HW_DIGCTL_MPTE8_LOC <<2) <= Address< 0x800C0000 + ((HW_DIGCTL_MPTE8_LOC + 2^SPAN)<<2)	Moveable PTE8 + ((Address[13:2] - HW_DIGCTL_MPTE8_LOC) << 20)
MPTE7	0x800C0000 + (HW_DIGCTL_MPTE7_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE7_LOC + 2^SPAN)<<2)	Moveable PTE7 + ((Address[13:2] - HW_DIGCTL_MPTE7_LOC) << 20)
MPTE6	0x800C0000 + (HW_DIGCTL_MPTE6_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE6_LOC + 2^SPAN)<<2)	Moveable PTE6 + ((Address[13:2] - HW_DIGCTL_MPTE6_LOC) << 20)
MPTE5	0x800C0000 + (HW_DIGCTL_MPTE5_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE5_LOC + 2^SPAN)<<2)	Moveable PTE5 + ((Address[13:2] - HW_DIGCTL_MPTE5_LOC) << 20)
MPTE4	0x800C0000 + (HW_DIGCTL_MPTE4_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE4_LOC + 2^SPAN)<<2)	Moveable PTE4 + ((Address[13:2] - HW_DIGCTL_MPTE4_LOC) << 20)
MPTE3	0x800C0000 + (HW_DIGCTL_MPTE3_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE3_LOC + 2^SPAN)<<2)	Moveable PTE3 + ((Address[13:2] - HW_DIGCTL_MPTE3_LOC) << 20)
MPTE2	0x800C0000 + (HW_DIGCTL_MPTE2_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE2_LOC + 2^SPAN)<<2)	Moveable PTE2 + ((Address[13:2] - HW_DIGCTL_MPTE2_LOC) << 20)
MPTE1	0x800C0000 + (HW_DIGCTL_MPTE1_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE1_LOC + 2^SPAN)<<2)	Moveable PTE1 + ((Address[13:2] - HW_DIGCTL_MPTE1_LOC) << 20)
MPTE0	0x800C0000 + (HW_DIGCTL_MPTE0_LOC <<2) <= Address < 0x800C0000 + ((HW_DIGCTL_MPTE0_LOC + 2^SPAN)<<2)	Moveable PTE0 + ((Address[13:2] - HW_DIGCTL_MPTE0_LOC) << 20)
PTE_2048	0x800C2000	PTE 2048 is semi-programmable, as shown in Default First-Level Page Table PIO Register Map Entry 2048.

3.2.3 Default First-Level Page Table PIO Register Map Entry 2048

The 1-Mbyte PIO region at physical address 0x800XXXXX is mapped as "virtual equal real" by the default first-level page table PTE_2048, as shown below.

DFLPT_PTE_2048 0x800C2000

Table 3-2. First-Level Page Table Entry 2048 (0x80000000–0x800FFFFF) at 0x800C2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Virtual == Real Section, i.e., 0x80000																				AP		0	DOMAIN				1	C	B	10	

Table 3-3. PTE 2048 Bit Field Descriptions

BITS	Label	RW	reset	Definition
31:12	**POINTER**	RO	0x80000	This section points to 0x80000000 and is always available.
11:10	**AP**	RW	0x3	Initially set to 0x3 for allowing ALL accesses. Set to other values as desired.
9	**ALWAYS_ZERO**	RO	0x0	Always reads back a 0.
8:5	**DOMAIN**	RW	0x0	Set as desired.
4	**ALWAYS_ONE**	RO	0x1	Always reads back a 1, as required in ARM926 Technical Reference Manual.
3	**CACHEABLE**	RO	0x0	Always reads back a 0 (uncached).
2	**BUFFERABLE**	RW	0x0	Always reads back a 0 (unbuffered).
1:0	**FIRST_LEVEL**	RO	0x2	Always reads back 0x2 for section descriptor.

Chapter 4
Memory Map

4.1 Memory Map Overview

The following table shows the memory map as seen by the processor. Any blank entries indicate that nothing is mapped at that address. No accesses should be made to these addresses since the results are indeterminate. The Decode Block column indicates the decode group to which each peripheral belongs. Most peripherals reside on the APBH or APBX peripheral buses.

Table 4-1. Address Map for i.MX28

DECODE BLOCK	DEVICE	MNEMONIC	START ADDRESS	END ADDRESS	SIZE
AHB	On-Chip RAM	OCRAM	0x00000000	0x0001FFFF	128KB
	On-Chip RAM alias	OCRAM	0x00020000	0x7FFFFFFF	
	External Memory	EMI	0x40000000	0x7FFFFFFF	1GB
APBH	Interrupt Controller	ICOLL	0x80000000	0x80001FFF	8KB
	High Resolution ADC	HSADC	0x80002000	0x80003FFF	8KB
	APBH DMA	APBH	0x80004000	0x80005FFF	8KB
	Performance Monitor	PERFMON	0x80006000	0x800067FF	2KB
			0x80006800	0x80006FFF	2KB
			0x80007000	0x800077FF	2KB
			0x80007800	0x80007FFF	2KB
			0x80008000	0x80009FFF	8KB
	BCH ECC	BCH	0x8000A000	0x8000BFFF	8KB
	General Purpose Media Interface	GPMI	0x8000C000	0x8000DFFF	8KB
			0x8000E000	0x8000FFFF	8KB
	Sync Serial Port 0	SSP0	0x80010000	0x80011FFF	8KB
	Sync Serial Port 1	SSP1	0x80012000	0x80013FFF	8KB
	Sync Serial Port 2	SSP2	0x80014000	0x80015FFF	8KB
	Sync Serial Port 3	SSP3	0x80016000	0x80017FFF	8KB
	Pin Control	PINCTRL	0x80018000	0x80019FFF	8KB

DECODE BLOCK	DEVICE	MNEMONIC	START ADDRESS	END ADDRESS	SIZE
			0x8001A000	0x8001BFFF	8KB
	Digital Control	DIGCTL	0x8001C000	0x8001DFFF	8KB
			0x8001E000	0x8001FFFF	8KB
			0x80020000	0x80021FFF	8KB
	ETM	ETM	0x80022000	0x80023FFF	8KB
	APBX DMA	APBX	0x80024000	0x80025FFF	8KB
			0x80026000	0x80027FFF	8KB
	Data CoProcessor	DCP	0x80028000	0x80029FFF	8KB
	Pixel Pipeline	PXP	0x8002A000	0x8002BFFF	8KB
	One Time Prog Controller	OCOTP	0x8002C000	0x8002DFFF	8KB
	AXI Control	AXI_AHB0	0x8002E000	0x8002FFFF	8KB
	LCD Interface	LCDIF	0x80030000	0x80031FFF	8KB
	CAN0	CAN0	0x80032000	0x80033FFF	8KB
	CAN1	CAN1	0x80034000	0x80035FFF	8KB
			0x80036000	0x80037FFF	8KB
			0x80038000	0x80039FFF	8KB
			0x8003A000	0x8003BFFF	8KB
	SIMDBG	SIMDBG	0x8003C000	0x8003C1ff	
	SIMGPMISEL	SIMGPMISEL	0x8003C200	0x8003C2ff	
	SIMSSPSEL	SIMSSPSEL	0x8003C300	0x8003C3ff	
	SIMMEMSEL	SIMMEMSEL	0x8003C400	0x8003C4ff	
	GPIOMON	GPIOMON	0x8003C500	0x8003C5ff	
	SIMENET	SIMENET	0x8003C700	0x8003C7ff	
	ARMJTAG	ARMJTAG	0x8003C800	0x8003C8ff	
APBX	Clock Controller	CLKCTRL	0x80040000	0x80041FFF	8KB
	Serial Audio Interface 0	SAIF0	0x80042000	0x80043FFF	8KB
	Power Control	POWER	0x80044000	0x80045FFF	8KB
	Serial Audio Interface 1	SAIF1	0x80046000	0x80047FFF	8KB
			0x80048000	0x80049FFF	8KB
			0x8004A000	0x8004BFFF	8KB
			0x8004C000	0x8004DFFF	8KB
			0x8004E000	0x8004FFFF	8KB
	Low Resolution ADC	LRADC	0x80050000	0x80051FFF	8KB
			0x80052000	0x80053FFF	8KB
	Sony/Phillips Digital Interface	SPDIF	0x80054000	0x80055FFF	8KB

DECODE BLOCK	DEVICE	MNEMONIC	START ADDRESS	END ADDRESS	SIZE
	Real Time Clock	RTC	0x80056000	0x80057FFF	8KB
	Inter-Integrated Circuit 0	I2C0	0x80058000	0x80059FFF	8KB
	Inter-Integrated Circuit 1	I2C1	0x8005A000	0x8005BFFF	8KB
			0x8005C000	0x8005DFFF	8KB
			0x8005E000	0x8005FFFF	8KB
			0x80060000	0x80061FFF	8KB
			0x80062000	0x80063FFF	8KB
	Pulse Width Modulation	PWM	0x80064000	0x80065FFF	8KB
			0x80066000	0x80067FFF	8KB
	Timers/Rotary	TIMROT	0x80068000	0x80069FFF	8KB
	Application UART 0	UARTAPP0	0x8006A000	0x8006BFFF	8KB
	Application UART 1	UARTAPP1	0x8006C000	0x8006DFFF	8KB
	Application UART 2	UARTAPP2	0x8006E000	0x8006FFFF	8KB
	Application UART 3	UARTAPP3	0x80070000	0x80071FFF	8KB
	Application UART 4	UARTAPP4	0x80072000	0x80073FFF	8KB
	Debug Uart	UARTDBG	0x80074000	0x80075FFF	8KB
			0x80076000	0x80077FFF	8KB
			0x80078000	0x80079FFF	8KB
			0x8007A000	0x8007BFFF	8KB
	Univeral Serial Bus Physical IF	USBPHY0	0x8007C000	0x8007DFFF	8KB
	Univeral Serial Bus Physical IF	USBPHY1	0x8007E000	0x8007FFFF	8KB
AHB	USB Controller 0	USBCTRL0	0x80080000	0x8008FFFF	64KB
	USB Controller 1	USBCTRL1	0x80090000	0x8009FFFF	64KB
			0x800A0000	0x800AFFFF	64KB
			0x800B0000	0x800BFFFF	64KB
	Default First-Level Page Table	DFLPT	0x800C0000	0x800CFFFF	64KB
			0x800D0000	0x800DFFFF	64KB
	External Memory Interface (REG)	DRAM	0x800E0000	0x800EFFFF	64KB
	ENET MAC0	ENET	0x800F0000	0x800F3FFF	16KB
	ENET MAC1	ENET	0x800F4000	0x800F7FFF	16KB
	ENT Switch	SWITCH	0x800F8000	0x800FFFFF	32KB
AHB	On-Chip ROM	OCROM	0xC0000000	0xFFFFFFFF	128KB

Chapter 5
Interrupt Collector (ICOLL)

5.1 Interrupt Collector (ICOLL) Overview

The ARM9 CPU core has two interrupt input lines, IRQ and FIQ. As shown in the following figure, the Interrupt Collector (ICOLL) can steer any of the 128 interrupt sources to either the FIQn or IRQn lines of the ARM9 CPU.

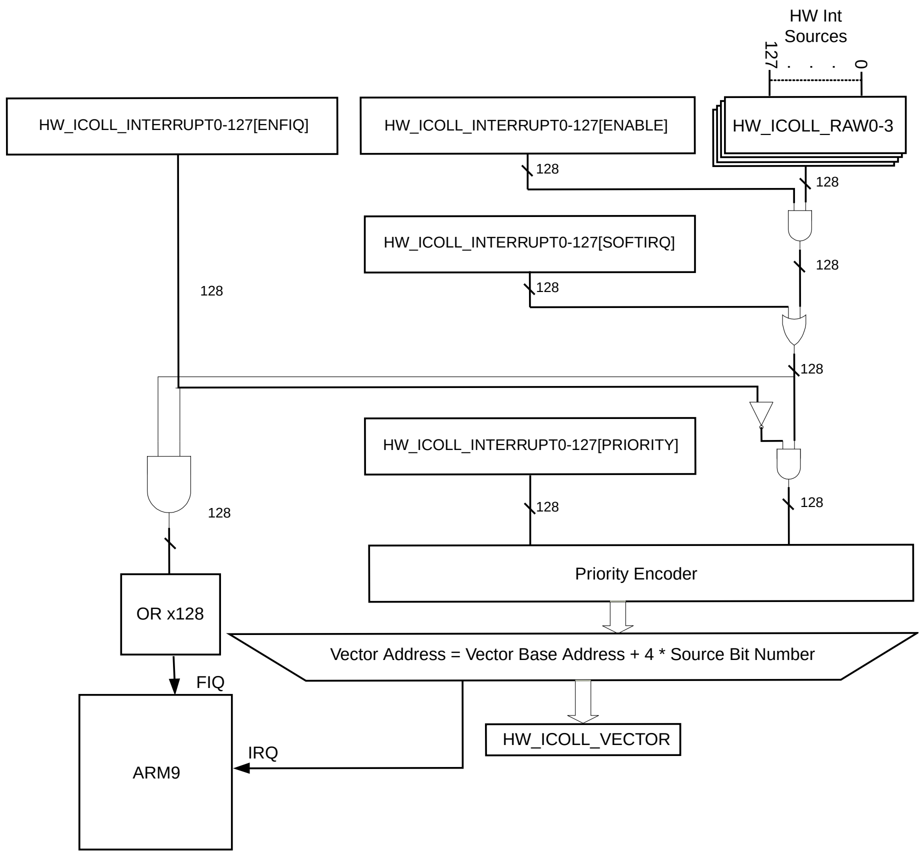

HW Int Sources
127
0
HW_ICOLL_INTERRUPT0-127[ENFIQ]
HW_ICOLL_INTERRUPT0-127[ENABLE]
HW_ICOLL_RAW0-3
HW_ICOLL_INTERRUPT0-127[SOFTIRQ]
HW_ICOLL_INTERRUPT0-127[PRIORITY]
128
Priority Encoder
OR x128
Vector Address = Vector Base Address + 4 * Source Bit Number
FIQ
IRQ
ARM9
HW_ICOLL_VECTOR

Figure 5-1. Interrupt Collector System Diagram

5.2 Operation

Within an individual interrupt request line (IRQ only), the ICOLL offers four-level priority (above base level) for each of its interrupt sources. Preemption of a lower priority interrupt by a higher priority is supported (interrupt nesting). Interrupts assigned to the same level are serviced in a strict linear priority order within level from lowest to highest interrupt source bit number. FIQ interrupts are neither prioritized nor vectorized. All the interrupt lines can be configured as a FIQ. If more than one is routed to the FIQ, then they must be discriminated by a software. It is highly recommended to reserve FIQ assignment to time critical events such as voltage brownouts or timers.

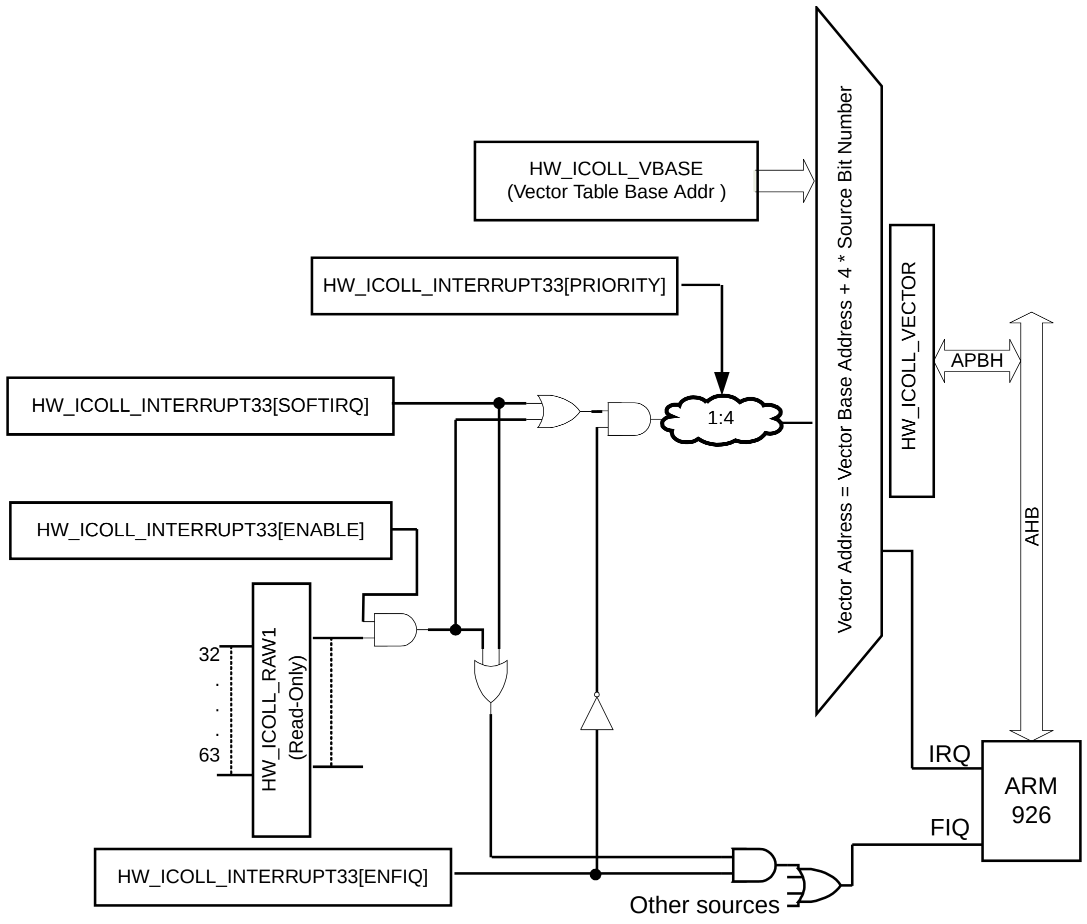

HW_ICOLL_VBASE
(Vector Table Base Addr)
HW_ICOLL_INTERRUPT33[PRIORITY]
HW_ICOLL_INTERRUPT33[SOFTIRQ]
1:4
HW_ICOLL_INTERRUPT33[ENABLE]
32
63
HW_ICOLL_RAW1
(Read-Only)
HW_ICOLL_INTERRUPT33[ENFIQ]
Other sources
Vector Address = Vector Base Address + 4 * Source Bit Number
HW_ICOLL_VECTOR
APBH
AHB
IRQ
FIQ
ARM
926

Figure 5-2. Interrupt Collector IRQ/FIQ Logic for Source 33

For a single interrupt source bit, there is an enable bit that gates it to the priority logic (HW_ICOLL_INTERRUPTn[ENABLE]). A software interrupt bit per source bit can be used to force an interrupt at the appropriate priority level directed to the corresponding vector address. Each source can be applied to one of the four interrupt levels.

The enable bit, FIQ-enable, the software interrupt bit, and the two-bit priority level specification for each interrupt source bit are contained with a single programmable register for each interrupt. The path from any interrupt source to the FIQ or IRQ logic is shown in Figure 5-2 using HW_ICOLL_INTERRUPT33 as an example.

The data path for generating the vector address (readable by software) for the IRQ generation portion of the interrupt collector is implemented as a multicycle path, as shown in Figure 5-3. The interrupt sources are continuously sampled in the holding register until one or more arrive. The FSM causes the holding register to stop sampling while a vector address is computed. Each interrupt source bit is applied to one of the four levels based on the two-bit priority specification of each source bit. When the holding register closes, there can be more than one newly arrived source bit. Thus, the source bits could be assigned such that more than one interrupt level is requesting an interrupt. The pipeline first determines the highest level requesting interrupt service. All interrupt requests on that level are presented to the linear priority encoder. The result of this stage is a six-bit number corresponding to the source bit number of the highest priority requesting an interrupt. This six-bit source number is used to compute the vector address as follows:

VectorAddress = VectorBase + (Pitch * SourceBitNumber)
Pitch = 4,8, 12,16,20,24, or 28 as desired, see HW_CTRL_VECTOR_PITCH.

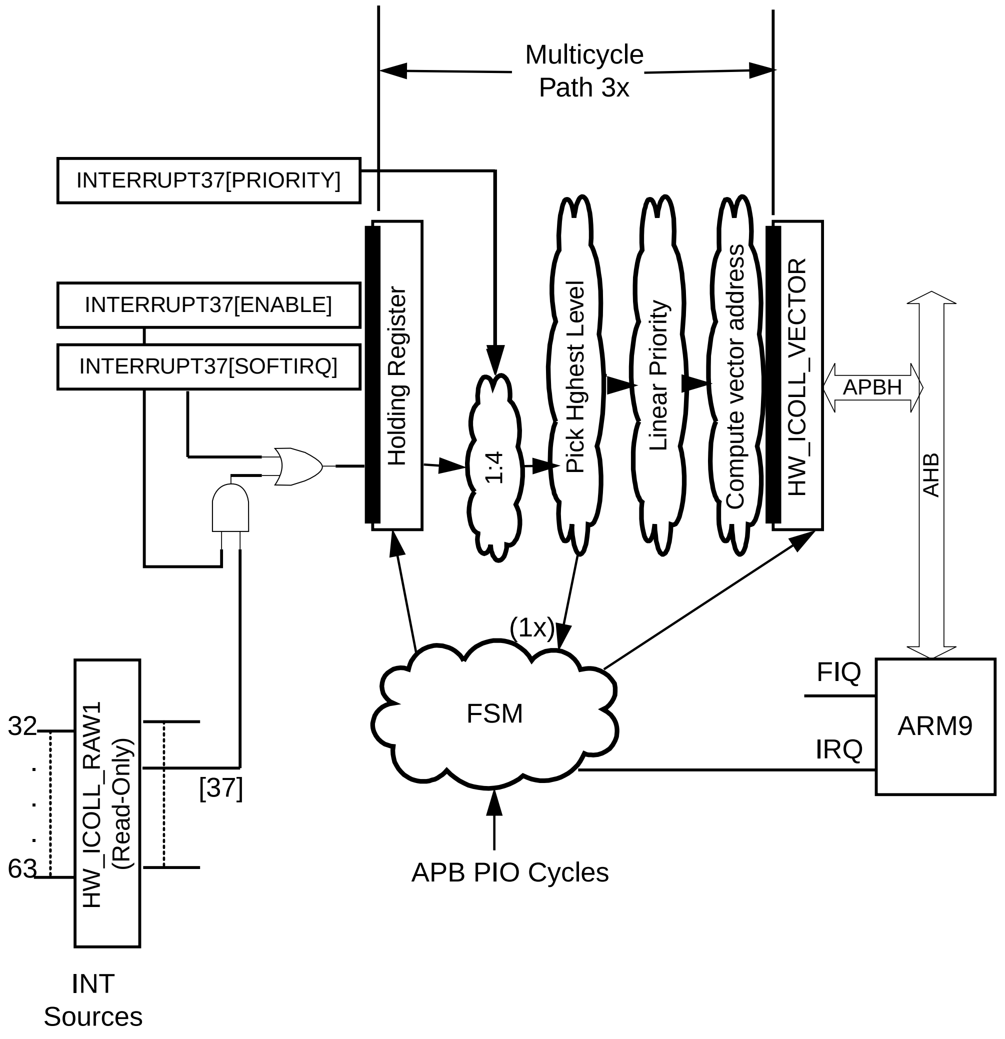


Figure 5-3. IRQ Control Flow

5.2.1 Nesting of Multi-Level IRQ Interrupts

There are a number of very important interactions between the interrupt collector's FSM and the interrupt service routine (ISR) running on the CPU. See Figure 5-4.

As soon as the interrupt source is recognized in the holding register, the FSM delays two clocks, then grabs the vector address and asserts IRQ to the CPU. After the CPU enters the interrupt service routine, it must notify the interrupt collector as soon as possible. Software indicates the in-service state by writing to the HW_ICOLL_VECTOR register. The contents of the data bus on this write do not matter. Optionally, firmware can enable the ARM read side-effect mode. In this case, the in-service state is indicated as a side effect of having read the HW_ICOLL_VECTOR register at the exception vector (0xFFFF0018). At this point, the FSM reopens the holding register and scans for new interrupt sources. Any such IRQ sources are presented to the CPU, provided that they are at a level higher than any currently in-service level.

Whenever the ARM CPU takes an IRQ exception, it turns off the IRQ enable in the CPU status register (CSR), as shown in Figure 5-4. If a higher priority interrupt is pending at this point, then another IRQ exception is taken.

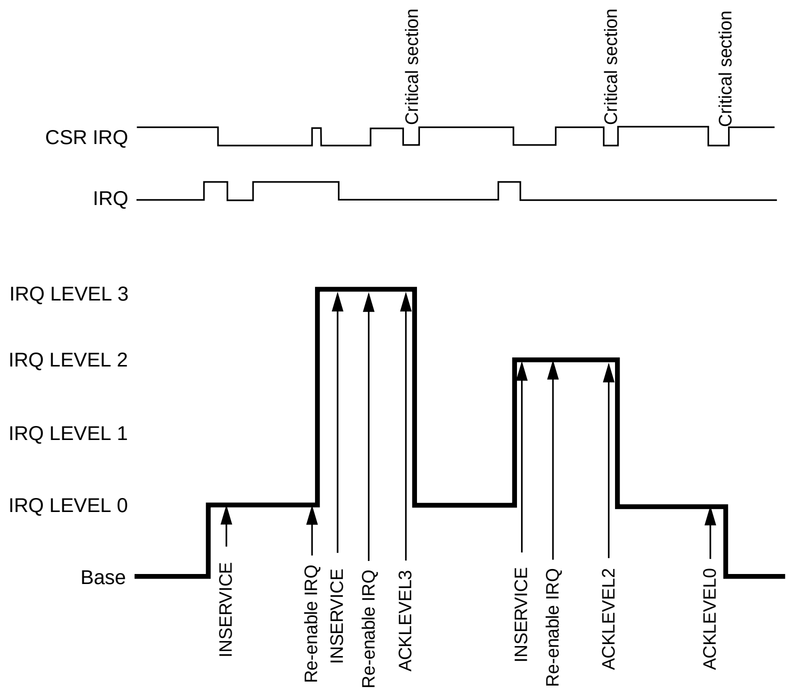


Figure 5-4. Nesting of Multi-Level IRQ Interrupts

The example in shows going from the base to a level 0 ISR. When the ISR at level 0 was ready, it enabled IRQ interrupts. At this point, it nests IRQ interrupts up to a level 3 interrupt. The level 3 ISR marks its in-service state, which causes the interrupt collector to open the holding register to search for new interrupt sources. In this example, none comes in, so the level 3 ISR completes. As part of the return process, the ISR disables IRQ interrupts, then acknowledges the level 3 service state. This is accomplished by writing the level number (3 in this case) to the interrupt collector's Level Acknowledge register. The interrupt collector resets the in-service bit for level 3. If this enables an IRQ at level 3, then it asserts IRQ and goes through the nesting process again. Since IRQ exceptions are masked in the level 3 ISR, this nesting does not take place until the level 3 ISR returns from interrupt. This return automatically re-enables IRQ exceptions. At this point, another exception could occur.

Figure 5-4 shows a second nesting of the IRQ interrupt by the arrival of a level 2 interrupt source bit. Finally, the figure shows the point at which the level 0 ISR enters its critical section (masks IRQ) and acknowledges level 0 to the interrupt collector and returns from the interrupt.

The FSM reverts to its BASE level state waiting for an interrupt request to arrive in the holding register. The waveform for the IRQ mask in the CPU status register (CSR) and the waveform for the IRQ input to the CPU as they relate to the interrupt collector action are shown in Figure 5-4.

NOTE

There is an inherent race condition between notifying the interrupt collector that an ISR has been entered and having that ISR re-enable IRQ exceptions in the CSR. The in-service notification can take a number of cycles to percolate through the write buffer, through the AHB and APB bridge and into the interrupt collector where it removes the IRQ assertion to the CPU. This ICOLL IRQ must be deasserted before the CSR IRQ on the CPU is re-enabled or the CPU will see a phantom interrupt. This is why the ARM vectored interrupt controller provides this in-service notification as a read side effect of the vector address read. Alternatively, the ISR can read the interrupt collector's CSR. The value received is unimportant, but the time required to do the read ensures that the write data has arrived at the interrupt collector. If firmware uses this method, it should allow clocks after the read for the FSM and for the CPU to recognize that the IRQ has been deasserted.

5.2.2 FIQ Generation

On this device, all interrupt sources can be configured as FIQ. This is controlled through the HW_ICOLL_INTERRUPTn[ENFIQ] register bit as shown in Figure 5-2. When enabled to the FIQ, the software interrupt associated with these bits can be used to generate the FIQ from these sources for test purposes. When an interrupt source is programmed as an FIQ, and IRQ cannot be generated from that source.

5.2.3 Interrupt Sources

The following table lists all of the interrupt sources on the device. Use hw_irq.h to access these bits.

Table 5-1. i.MX28 Interrupt Sources

Source Number	Interrupt	Vector	Description
0	batt_brownout_irq	0x0000	Power module battery brownout detect IRQ, recommend to set as FIQ.

Source Number	Interrupt	Vector	Description
1	vddd_brownout_irq	0x0004	Power module VDDD brownout detect IRQ, recommend to set as FIQ.
2	vddio_brownout_irq	0x0008	Power module VDDIO brownout detect IRQ, recommend to set as FIQ.
3	vdda_brownout_irq	0x000C	Power module VDDA brownout detect IRQ, recommend to set as FIQ.
4	vdd5v_droop_irq	0x0010	5V Droop IRQ, recommend to be set as FIQ.
5	dcdc4p2_brownout_irq	0x0014	4.2V regulated supply brown-out IRQ, recommend to be set as FIQ.
6	vdd5v_irq	0x0018	IRQ on 5V connect or disconnect also OTG 4.2V
7		0x001C	Reserved
8	can0_irq	0x0020	CAN0 IRQ.
9	can1_irq	0x0024	CAN1 IRQ.
10	lradc_touch_irq	0x0028	(Touch Screen) Touch detection IRQ.
11		0x002C	Reserved
12		0x0030	Reserved
13	hsadc_irq	0x0034	HSADC IRQ.
14	lradc_thresh0_irq	0x0038	LRADC0 Threshold IRQ.
15	lradc_thresh1_irq	0x003C	LRADC1 Threshold IRQ.
16	lradc_ch0_irq	0x0040	LRADC Channel 0 conversion complete IRQ.
17	lradc_ch1_irq	0x0044	LRADC Channel 1 conversion complete IRQ.
18	lradc_ch2_irq	0x0048	LRADC Channel 2 conversion complete IRQ.
19	lradc_ch3_irq	0x004C	LRADC Channel 3 conversion complete IRQ.
20	lradc_ch4_irq	0x0050	LRADC Channel 4 conversion complete IRQ.
21	lradc_ch5_irq	0x0054	LRADC Channel 5 conversion complete IRQ.
22	lradc_ch6_irq	0x0058	LRADC Channel 6 conversion complete IRQ.
23	lradc_ch7_irq	0x005C	LRADC Channel 7 conversion complete IRQ.
24	lradc_button0_irq	0x0060	LRADC Channel 0 button detection IRQ.
25	lradc_button1_irq	0x0064	LRADC Channel 1 button detection IRQ.
26		0x0068	Reserved

Source Number	Interrupt	Vector	Description
27	perfmon_irq	0x006C	Performance monitor IRQ.
28	rtc_1msec_irq	0x0070	RTC 1ms event IRQ.
29	rtc_alarm_irq	0x0074	RTC alarm event IRQ.
30		0x0078	Reserved
31	comms_irq	0x007C	JTAG debug communications port IRQ.
32	emi_error_irq	0x0080	External memory controller IRQ.
33		0x0084	Reserved
34		0x0088	Reserved
35		0x008C	Reserved
36		0x0090	Reserved
37	reserved	0x0094	Reserved
38	lcdif_irq	0x0098	LCDIF IRQ.
39	pxp_irq	0x009C	PXP IRQ.
40		0x00A0	Reserved
41	bch_irq	0x00A4	BCH consolidated IRQ.
42	gpmi_irq	0x00A8	GPMI internal error and status IRQ.
43		0x00AC	Reserved
44		0x00B0	Reserved
45	spdif_error_irq	0x00B4	SPDIF FIFO error IRQ.
46		0x00B8	Reserved
47	duart_irq	0x00BC	Debug UART IRQ.
48	timer0_irq	0x00C0	Timer0 IRQ, recommend to set as FIQ.
49	timer1_irq	0x00C4	Timer1 IRQ, recommend to set as FIQ.
50	timer2_irq	0x00C8	Timer2 IRQ, recommend to set as FIQ.
51	timer3_irq	0x00CC	Timer3 IRQ, recommend to set as FIQ.
52	dcp_vmi_irq	0x00D0	DCP Channel 0 virtual memory page copy IRQ.
53	dcp_irq	0x00D4	DCP (per channel and CSC) IRQ.

Source Number	Interrupt	Vector	Description
54	dcp_secure_irq	0x00D8	DCP secure IRQ.
55		0x00DC	Reserved
56		0x00E0	Reserved
57		0x00E4	Reserved
58	saif1_irq	0x00E8	SAIF1 FIFO & Service error IRQ.
59	saif0_irq	0x00EC	SAIF0 FIFO & Service error IRQ.
60		0x00F0	Reserved
61		0x00F4	Reserved
62		0x00F8	Reserved
63		0x00FC	Reserved
64		0x0100	Reserved
65		0x0104	Reserved
66	spdif_dma_irq	0x0108	SPDIF DMA channel IRQ.
67		0x010C	Reserved
68	i2c0_dma_irq	0x0110	I2C0 DMA channel IRQ.
69	i2c1_dma_irq	0x0114	I2C1 DMA channel IRQ.
70	auart0_rx_dma_irq	0x0118	Application UART0 receiver DMA channel IRQ.
71	auart0_tx_dma_irq	0x011C	Application UART0 transmitter DMA channel IRQ.
72	auart1_rx_dma_irq	0x0120	Application UART1 receiver DMA channel IRQ.
73	auart1_tx_dma_irq	0x0124	Application UART1 transmitter DMA
74	auart2_rx_dma_irq	0x0128	Application UART2 receiver DMA channel IRQ.
75	auart2_tx_dma_irq	0x012C	Application UART2 transmitter DMA channel IRQ.
76	auart3_rx_dma_irq	0x0130	Application UART3 receiver DMA channel IRQ.
77	auart3_tx_dma_irq	0x0134	Application UART3 transmitter DMA channel IRQ.
78	auart4_rx_dma_irq	0x0138	Application UART4 receiver DMA channel IRQ.
79	auart4_tx_dma_irq	0x013C	Application UART4 transmitter DMA channel IRQ.
80	saif0_dma_irq	0x0140	SAIF0 DMA channel IRQ.

Source Number	Interrupt	Vector	Description
81	saif1_dma_irq	0x0144	SAIF1 DMA channel IRQ.
82	ssp0_dma_irq	0x0148	SSP0 DMA channel IRQ.
83	ssp1_dma_irq	0x014C	SSP1 DMA channel IRQ.
84	ssp2_dma_irq	0x0150	SSP2 DMA channel IRQ.
85	ssp3_dma_irq	0x0154	SSP3 DMA channel IRQ.
86	lcdif_dma_irq	0x0158	LCDIF DMA channel IRQ.
87	hsadc_dma_irq	0x015C	HSADC DMA channel IRQ.
88	gpmi_dma_irq	0x0160	GPMI DMA channel IRQ.
89	digctl_debug_trap_irq	0x0164	Layer 0 or Layer 3 AHB address access trap IRQ.
90		0x0168	Reserved
91		0x016C	Reserved
92	usb1_irq	0x0170	USB1 IRQ.
93	usb0_irq	0x0174	USB0 IRQ.
94	usb1_wakeup_irq	0x0178	UTM1 IRQ.
95	usb0_wakeup_irq	0x017C	UTM0 IRQ.
96	ssp0_error_irq	0x0180	SSP0 device-level error and status IRQ.
97	ssp1_error_irq	0x0184	SSP1 device-level error and status IRQ.
98	ssp2_error_irq	0x0188	SSP2 device-level error and status IRQ.
99	ssp3_error_irq	0x018C	SSP3 device-level error and status IRQ.
100	enet_swi_irq	0x0190	Switch IRQ.
101	enet_mac0_irq	0x0194	MAC0 IRQ.
102	enet_mac1_irq	0x0198	MAC1 IRQ.
103	enet_mac0_1588_irq	0x019C	1588 of MAC0 IRQ.
104	enet_mac1_1588_irq	0x01A0	1588 of MAC1 IRQ.
105		0x01A4	Reserved
106		0x01A8	Reserved
107		0x01AC	Reserved

Source Number	Interrupt	Vector	Description
108		0x01B0	Reserved
109		0x01B4	Reserved
110	i2c1_error_irq	0x01B8	I2C1 device detected errors and line conditions IRQ.
111	i2c0_error_irq	0x01BC	I2C0 device detected errors and line conditions IRQ.
112	auart0_irq	0x01C0	Application UART0 internal error IRQ.
113	auart1_irq	0x01C4	Application UART1 internal error IRQ.
114	auart2_irq	0x01C8	Application UART2 internal error IRQ.
115	auart3_irq	0x01CC	Application UART3 internal error IRQ.
116	auart4_irq	0x01D0	Application UART4 internal error IRQ.
117		0x01D4	Reserved
118		0x01D8	Reserved
119		0x01DC	Reserved
120		0x01E0	Reserved
121		0x01E4	Reserved
122	pinctrl5_irq	0x01E8	GPIO bank 5 interrupt IRQ.
123	pinctrl4_irq	0x01EC	GPIO bank 4 interrupt IRQ.
124	pinctrl3_irq	0x01F0	GPIO bank 3 interrupt IRQ.
125	pinctrl2_irq	0x01F4	GPIO bank 2 interrupt IRQ.
126	pinctrl1_irq	0x01F8	GPIO bank 1 interrupt IRQ.
127	pinctrl0_irq	0x01FC	GPIO bank 0 interrupt IRQ.

5.2.4 CPU Wait-for-Interrupt Mode

To enable wait-for-interrupt mode, two distinct actions are required by the programmer.

1. Set the INTERRUPT_WAIT bit in the HW_CLKCTRL_CPUCLKCTRL register. This must be done through a RMW operation. For example:

```
uclkctrl  = HW_CLKCTRL_CPUCLKCTRL_RD();
```

```
                uclkctrl |= BM_CLKCTRL_CPUCLKCTRL_INTERRUPT_WAIT;
                HW_CLKCTRL_CPUCLKCTRL_WR(uclkctrl);
```

2. After setting the INTERRUPT_WAIT bit, a coprocessor instruction is required.

```
    asm (
        // Note: R0 is used in the following example, but any usual <Rd> register may be used.

        "mov R0, 0;"             // Rd SBZ (should be zero)
        "mcr p15,0,r0,c7,c0,4;" //Drain write buffers, idle CPU clock & processor, and stop
                                 //processor at this instruction
        "nop");                  // The lr sent to handler points here after RTI
```

The coprocessor instruction sequence above enables an internal gating signal. This internal signal guarantees that write buffers are drained and ensures that the processor is in an idle state. On execution of the MCR coprocessor instruction, the CPU clock is stopped and the processor halts on the instruction—waiting for an interrupt to occur.

The INTERRUPT_WAIT bit can be thought of as a Wait-for-Interrupt enable bit. Therefore, it must be set prior to the execution of the MCR instruction. It is recommended that, when the Wait-for-Interrupt mode is to be used, the INTERRUPT_WAIT bit be set at initialization time and left on.

With the INTERRUPT_WAIT bit set, after the execution of the MCR WFI command, the processor halts on the MCR instruction. When an interrupt or FIQ occurs, the MCR instruction completes and the IRQ or FIQ handler is entered normally. The return link that is passed to the handler is automatically adjusted by the above MCR instruction, such that a normal return from an interrupt results in a continuous execution of the instruction immediately following the MCR. That is, the LR will contain the address of the MCR instruction plus eight, such that a typical return from an interrupt instruction (for example, subs pc, LR, 4) will return to the instruction immediately following the MCR (the NOP in the example above).

Whenever the CPU is stopped because the clock control HW_CLKCTRL_ CPUCLKCTRL_INTERRUPT_WAIT bit is set and the MCR WFI instruction is executed, the CPU stops until an interrupt occurs. The actual condition that wakes up the CPU is determined by ORing together all enabled interrupt requests including those that are directed to the FIQ CPU input. The ICOLL_BUSY output signal from the ICOLL communicates this information to the clock control. This function does not pass through the normal ICOLL state machine. It starts the CPU clock as soon as an enabled interrupt arrives.

5.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block, for additional information on using the SFTRST and CLKGATE bit fields.

5.4 Programmable Registers

ICOLL Hardware Register Format Summary

HW_ICOLL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_0000	Interrupt Collector Interrupt Vector Address Register (HW_ICOLL_VECTOR)	32	R/W	0000_0000h	5.4.1/148
8000_0010	Interrupt Collector Level Acknowledge Register (HW_ICOLL_LEVELACK)	32	R/W	0000_0000h	5.4.2/149
8000_0020	Interrupt Collector Control Register (HW_ICOLL_CTRL)	32	R/W	C003_0000h	5.4.3/150
8000_0040	Interrupt Collector Interrupt Vector Base Address Register (HW_ICOLL_VBASE)	32	R/W	0000_0000h	5.4.4/152
8000_0070	Interrupt Collector Status Register (HW_ICOLL_STAT)	32	R	0000_007Fh	5.4.5/153
8000_00A0	Interrupt Collector Raw Interrupt Input Register 0 (HW_ICOLL_RAW0)	32	R	0000_0000h	5.4.6/154
8000_00B0	Interrupt Collector Raw Interrupt Input Register 1 (HW_ICOLL_RAW1)	32	R	0000_0000h	5.4.7/154
8000_00C0	Interrupt Collector Raw Interrupt Input Register 2 (HW_ICOLL_RAW2)	32	R	0000_0000h	5.4.8/155
8000_00D0	Interrupt Collector Raw Interrupt Input Register 3 (HW_ICOLL_RAW3)	32	R	0000_0000h	5.4.9/156
8000_0120	Interrupt Collector Interrupt Register 0 (HW_ICOLL_INTERRUPT0)	32	R/W	0000_0000h	5.4.10/157
8000_0130	Interrupt Collector Interrupt Register 1 (HW_ICOLL_INTERRUPT1)	32	R/W	0000_0000h	5.4.11/158
8000_0140	Interrupt Collector Interrupt Register 2 (HW_ICOLL_INTERRUPT2)	32	R/W	0000_0000h	5.4.12/159
8000_0150	Interrupt Collector Interrupt Register 3 (HW_ICOLL_INTERRUPT3)	32	R/W	0000_0000h	5.4.13/160
8000_0160	Interrupt Collector Interrupt Register 4 (HW_ICOLL_INTERRUPT4)	32	R/W	0000_0000h	5.4.14/162
8000_0170	Interrupt Collector Interrupt Register 5 (HW_ICOLL_INTERRUPT5)	32	R/W	0000_0000h	5.4.15/163

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_0180	Interrupt Collector Interrupt Register 6 (HW_ICOLL_INTERRUPT6)	32	R/W	0000_0000h	5.4.16/164
8000_0190	Interrupt Collector Interrupt Register 7 (HW_ICOLL_INTERRUPT7)	32	R/W	0000_0000h	5.4.17/166
8000_01A0	Interrupt Collector Interrupt Register 8 (HW_ICOLL_INTERRUPT8)	32	R/W	0000_0000h	5.4.18/167
8000_01B0	Interrupt Collector Interrupt Register 9 (HW_ICOLL_INTERRUPT9)	32	R/W	0000_0000h	5.4.19/168
8000_01C0	Interrupt Collector Interrupt Register 10 (HW_ICOLL_INTERRUPT10)	32	R/W	0000_0000h	5.4.20/170
8000_01D0	Interrupt Collector Interrupt Register 11 (HW_ICOLL_INTERRUPT11)	32	R/W	0000_0000h	5.4.21/171
8000_01E0	Interrupt Collector Interrupt Register 12 (HW_ICOLL_INTERRUPT12)	32	R/W	0000_0000h	5.4.22/172
8000_01F0	Interrupt Collector Interrupt Register 13 (HW_ICOLL_INTERRUPT13)	32	R/W	0000_0000h	5.4.23/174
8000_0200	Interrupt Collector Interrupt Register 14 (HW_ICOLL_INTERRUPT14)	32	R/W	0000_0000h	5.4.24/175
8000_0210	Interrupt Collector Interrupt Register 15 (HW_ICOLL_INTERRUPT15)	32	R/W	0000_0000h	5.4.25/177
8000_0220	Interrupt Collector Interrupt Register 16 (HW_ICOLL_INTERRUPT16)	32	R/W	0000_0000h	5.4.26/178
8000_0230	Interrupt Collector Interrupt Register 17 (HW_ICOLL_INTERRUPT17)	32	R/W	0000_0000h	5.4.27/179
8000_0240	Interrupt Collector Interrupt Register 18 (HW_ICOLL_INTERRUPT18)	32	R/W	0000_0000h	5.4.28/181
8000_0250	Interrupt Collector Interrupt Register 19 (HW_ICOLL_INTERRUPT19)	32	R/W	0000_0000h	5.4.29/182
8000_0260	Interrupt Collector Interrupt Register 20 (HW_ICOLL_INTERRUPT20)	32	R/W	0000_0000h	5.4.30/184
8000_0270	Interrupt Collector Interrupt Register 21 (HW_ICOLL_INTERRUPT21)	32	R/W	0000_0000h	5.4.31/185
8000_0280	Interrupt Collector Interrupt Register 22 (HW_ICOLL_INTERRUPT22)	32	R/W	0000_0000h	5.4.32/186
8000_0290	Interrupt Collector Interrupt Register 23 (HW_ICOLL_INTERRUPT23)	32	R/W	0000_0000h	5.4.33/188
8000_02A0	Interrupt Collector Interrupt Register 24 (HW_ICOLL_INTERRUPT24)	32	R/W	0000_0000h	5.4.34/189
8000_02B0	Interrupt Collector Interrupt Register 25 (HW_ICOLL_INTERRUPT25)	32	R/W	0000_0000h	5.4.35/191
8000_02C0	Interrupt Collector Interrupt Register 26 (HW_ICOLL_INTERRUPT26)	32	R/W	0000_0000h	5.4.36/192

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_02D0	Interrupt Collector Interrupt Register 27 (HW_ICOLL_INTERRUPT27)	32	R/W	0000_0000h	5.4.37/193
8000_02E0	Interrupt Collector Interrupt Register 28 (HW_ICOLL_INTERRUPT28)	32	R/W	0000_0000h	5.4.38/195
8000_02F0	Interrupt Collector Interrupt Register 29 (HW_ICOLL_INTERRUPT29)	32	R/W	0000_0000h	5.4.39/196
8000_0300	Interrupt Collector Interrupt Register 30 (HW_ICOLL_INTERRUPT30)	32	R/W	0000_0000h	5.4.40/198
8000_0310	Interrupt Collector Interrupt Register 31 (HW_ICOLL_INTERRUPT31)	32	R/W	0000_0000h	5.4.41/199
8000_0320	Interrupt Collector Interrupt Register 32 (HW_ICOLL_INTERRUPT32)	32	R/W	0000_0000h	5.4.42/200
8000_0330	Interrupt Collector Interrupt Register 33 (HW_ICOLL_INTERRUPT33)	32	R/W	0000_0000h	5.4.43/202
8000_0340	Interrupt Collector Interrupt Register 34 (HW_ICOLL_INTERRUPT34)	32	R/W	0000_0000h	5.4.44/203
8000_0350	Interrupt Collector Interrupt Register 35 (HW_ICOLL_INTERRUPT35)	32	R/W	0000_0000h	5.4.45/205
8000_0360	Interrupt Collector Interrupt Register 36 (HW_ICOLL_INTERRUPT36)	32	R/W	0000_0000h	5.4.46/206
8000_0370	Interrupt Collector Interrupt Register 37 (HW_ICOLL_INTERRUPT37)	32	R/W	0000_0000h	5.4.47/207
8000_0380	Interrupt Collector Interrupt Register 38 (HW_ICOLL_INTERRUPT38)	32	R/W	0000_0000h	5.4.48/209
8000_0390	Interrupt Collector Interrupt Register 39 (HW_ICOLL_INTERRUPT39)	32	R/W	0000_0000h	5.4.49/210
8000_03A0	Interrupt Collector Interrupt Register 40 (HW_ICOLL_INTERRUPT40)	32	R/W	0000_0000h	5.4.50/212
8000_03B0	Interrupt Collector Interrupt Register 41 (HW_ICOLL_INTERRUPT41)	32	R/W	0000_0000h	5.4.51/213
8000_03C0	Interrupt Collector Interrupt Register 42 (HW_ICOLL_INTERRUPT42)	32	R/W	0000_0000h	5.4.52/214
8000_03D0	Interrupt Collector Interrupt Register 43 (HW_ICOLL_INTERRUPT43)	32	R/W	0000_0000h	5.4.53/216
8000_03E0	Interrupt Collector Interrupt Register 44 (HW_ICOLL_INTERRUPT44)	32	R/W	0000_0000h	5.4.54/217
8000_03F0	Interrupt Collector Interrupt Register 45 (HW_ICOLL_INTERRUPT45)	32	R/W	0000_0000h	5.4.55/219
8000_0400	Interrupt Collector Interrupt Register 46 (HW_ICOLL_INTERRUPT46)	32	R/W	0000_0000h	5.4.56/220
8000_0410	Interrupt Collector Interrupt Register 47 (HW_ICOLL_INTERRUPT47)	32	R/W	0000_0000h	5.4.57/221

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_0420	Interrupt Collector Interrupt Register 48 (HW_ICOLL_INTERRUPT48)	32	R/W	0000_0000h	5.4.58/223
8000_0430	Interrupt Collector Interrupt Register 49 (HW_ICOLL_INTERRUPT49)	32	R/W	0000_0000h	5.4.59/224
8000_0440	Interrupt Collector Interrupt Register 50 (HW_ICOLL_INTERRUPT50)	32	R/W	0000_0000h	5.4.60/226
8000_0450	Interrupt Collector Interrupt Register 51 (HW_ICOLL_INTERRUPT51)	32	R/W	0000_0000h	5.4.61/227
8000_0460	Interrupt Collector Interrupt Register 52 (HW_ICOLL_INTERRUPT52)	32	R/W	0000_0000h	5.4.62/228
8000_0470	Interrupt Collector Interrupt Register 53 (HW_ICOLL_INTERRUPT53)	32	R/W	0000_0000h	5.4.63/230
8000_0480	Interrupt Collector Interrupt Register 54 (HW_ICOLL_INTERRUPT54)	32	R/W	0000_0000h	5.4.64/231
8000_0490	Interrupt Collector Interrupt Register 55 (HW_ICOLL_INTERRUPT55)	32	R/W	0000_0000h	5.4.65/233
8000_04A0	Interrupt Collector Interrupt Register 56 (HW_ICOLL_INTERRUPT56)	32	R/W	0000_0000h	5.4.66/234
8000_04B0	Interrupt Collector Interrupt Register 57 (HW_ICOLL_INTERRUPT57)	32	R/W	0000_0000h	5.4.67/235
8000_04C0	Interrupt Collector Interrupt Register 58 (HW_ICOLL_INTERRUPT58)	32	R/W	0000_0000h	5.4.68/237
8000_04D0	Interrupt Collector Interrupt Register 59 (HW_ICOLL_INTERRUPT59)	32	R/W	0000_0000h	5.4.69/238
8000_04E0	Interrupt Collector Interrupt Register 60 (HW_ICOLL_INTERRUPT60)	32	R/W	0000_0000h	5.4.70/240
8000_04F0	Interrupt Collector Interrupt Register 61 (HW_ICOLL_INTERRUPT61)	32	R/W	0000_0000h	5.4.71/241
8000_0500	Interrupt Collector Interrupt Register 62 (HW_ICOLL_INTERRUPT62)	32	R/W	0000_0000h	5.4.72/242
8000_0510	Interrupt Collector Interrupt Register 63 (HW_ICOLL_INTERRUPT63)	32	R/W	0000_0000h	5.4.73/244
8000_0520	Interrupt Collector Interrupt Register 64 (HW_ICOLL_INTERRUPT64)	32	R/W	0000_0000h	5.4.74/245
8000_0530	Interrupt Collector Interrupt Register 65 (HW_ICOLL_INTERRUPT65)	32	R/W	0000_0000h	5.4.75/247
8000_0540	Interrupt Collector Interrupt Register 66 (HW_ICOLL_INTERRUPT66)	32	R/W	0000_0000h	5.4.76/248
8000_0550	Interrupt Collector Interrupt Register 67 (HW_ICOLL_INTERRUPT67)	32	R/W	0000_0000h	5.4.77/249
8000_0560	Interrupt Collector Interrupt Register 68 (HW_ICOLL_INTERRUPT68)	32	R/W	0000_0000h	5.4.78/251

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_0570	Interrupt Collector Interrupt Register 69 (HW_ICOLL_INTERRUPT69)	32	R/W	0000_0000h	5.4.79/252
8000_0580	Interrupt Collector Interrupt Register 70 (HW_ICOLL_INTERRUPT70)	32	R/W	0000_0000h	5.4.80/254
8000_0590	Interrupt Collector Interrupt Register 71 (HW_ICOLL_INTERRUPT71)	32	R/W	0000_0000h	5.4.81/255
8000_05A0	Interrupt Collector Interrupt Register 72 (HW_ICOLL_INTERRUPT72)	32	R/W	0000_0000h	5.4.82/256
8000_05B0	Interrupt Collector Interrupt Register 73 (HW_ICOLL_INTERRUPT73)	32	R/W	0000_0000h	5.4.83/258
8000_05C0	Interrupt Collector Interrupt Register 74 (HW_ICOLL_INTERRUPT74)	32	R/W	0000_0000h	5.4.84/259
8000_05D0	Interrupt Collector Interrupt Register 75 (HW_ICOLL_INTERRUPT75)	32	R/W	0000_0000h	5.4.85/261
8000_05E0	Interrupt Collector Interrupt Register 76 (HW_ICOLL_INTERRUPT76)	32	R/W	0000_0000h	5.4.86/262
8000_05F0	Interrupt Collector Interrupt Register 77 (HW_ICOLL_INTERRUPT77)	32	R/W	0000_0000h	5.4.87/263
8000_0600	Interrupt Collector Interrupt Register 78 (HW_ICOLL_INTERRUPT78)	32	R/W	0000_0000h	5.4.88/265
8000_0610	Interrupt Collector Interrupt Register 79 (HW_ICOLL_INTERRUPT79)	32	R/W	0000_0000h	5.4.89/266
8000_0620	Interrupt Collector Interrupt Register 80 (HW_ICOLL_INTERRUPT80)	32	R/W	0000_0000h	5.4.90/268
8000_0630	Interrupt Collector Interrupt Register 81 (HW_ICOLL_INTERRUPT81)	32	R/W	0000_0000h	5.4.91/269
8000_0640	Interrupt Collector Interrupt Register 82 (HW_ICOLL_INTERRUPT82)	32	R/W	0000_0000h	5.4.92/270
8000_0650	Interrupt Collector Interrupt Register 83 (HW_ICOLL_INTERRUPT83)	32	R/W	0000_0000h	5.4.93/272
8000_0660	Interrupt Collector Interrupt Register 84 (HW_ICOLL_INTERRUPT84)	32	R/W	0000_0000h	5.4.94/273
8000_0670	Interrupt Collector Interrupt Register 85 (HW_ICOLL_INTERRUPT85)	32	R/W	0000_0000h	5.4.95/275
8000_0680	Interrupt Collector Interrupt Register 86 (HW_ICOLL_INTERRUPT86)	32	R/W	0000_0000h	5.4.96/276
8000_0690	Interrupt Collector Interrupt Register 87 (HW_ICOLL_INTERRUPT87)	32	R/W	0000_0000h	5.4.97/277
8000_06A0	Interrupt Collector Interrupt Register 88 (HW_ICOLL_INTERRUPT88)	32	R/W	0000_0000h	5.4.98/279
8000_06B0	Interrupt Collector Interrupt Register 89 (HW_ICOLL_INTERRUPT89)	32	R/W	0000_0000h	5.4.99/280

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_06C0	Interrupt Collector Interrupt Register 90 (HW_ICOLL_INTERRUPT90)	32	R/W	0000_0000h	5.4.100/282
8000_06D0	Interrupt Collector Interrupt Register 91 (HW_ICOLL_INTERRUPT91)	32	R/W	0000_0000h	5.4.101/283
8000_06E0	Interrupt Collector Interrupt Register 92 (HW_ICOLL_INTERRUPT92)	32	R/W	0000_0000h	5.4.102/284
8000_06F0	Interrupt Collector Interrupt Register 93 (HW_ICOLL_INTERRUPT93)	32	R/W	0000_0000h	5.4.103/286
8000_0700	Interrupt Collector Interrupt Register 94 (HW_ICOLL_INTERRUPT94)	32	R/W	0000_0000h	5.4.104/287
8000_0710	Interrupt Collector Interrupt Register 95 (HW_ICOLL_INTERRUPT95)	32	R/W	0000_0000h	5.4.105/289
8000_0720	Interrupt Collector Interrupt Register 96 (HW_ICOLL_INTERRUPT96)	32	R/W	0000_0000h	5.4.106/290
8000_0730	Interrupt Collector Interrupt Register 97 (HW_ICOLL_INTERRUPT97)	32	R/W	0000_0000h	5.4.107/291
8000_0740	Interrupt Collector Interrupt Register 98 (HW_ICOLL_INTERRUPT98)	32	R/W	0000_0000h	5.4.108/293
8000_0750	Interrupt Collector Interrupt Register 99 (HW_ICOLL_INTERRUPT99)	32	R/W	0000_0000h	5.4.109/294
8000_0760	Interrupt Collector Interrupt Register 100 (HW_ICOLL_INTERRUPT100)	32	R/W	0000_0000h	5.4.110/296
8000_0770	Interrupt Collector Interrupt Register 101 (HW_ICOLL_INTERRUPT101)	32	R/W	0000_0000h	5.4.111/297
8000_0780	Interrupt Collector Interrupt Register 102 (HW_ICOLL_INTERRUPT102)	32	R/W	0000_0000h	5.4.112/298
8000_0790	Interrupt Collector Interrupt Register 103 (HW_ICOLL_INTERRUPT103)	32	R/W	0000_0000h	5.4.113/300
8000_07A0	Interrupt Collector Interrupt Register 104 (HW_ICOLL_INTERRUPT104)	32	R/W	0000_0000h	5.4.114/301
8000_07B0	Interrupt Collector Interrupt Register 105 (HW_ICOLL_INTERRUPT105)	32	R/W	0000_0000h	5.4.115/303
8000_07C0	Interrupt Collector Interrupt Register 106 (HW_ICOLL_INTERRUPT106)	32	R/W	0000_0000h	5.4.116/304
8000_07D0	Interrupt Collector Interrupt Register 107 (HW_ICOLL_INTERRUPT107)	32	R/W	0000_0000h	5.4.117/305
8000_07E0	Interrupt Collector Interrupt Register 108 (HW_ICOLL_INTERRUPT108)	32	R/W	0000_0000h	5.4.118/307
8000_07F0	Interrupt Collector Interrupt Register 109 (HW_ICOLL_INTERRUPT109)	32	R/W	0000_0000h	5.4.119/308
8000_0800	Interrupt Collector Interrupt Register 110 (HW_ICOLL_INTERRUPT110)	32	R/W	0000_0000h	5.4.120/310

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_0810	Interrupt Collector Interrupt Register 111 (HW_ICOLL_INTERRUPT111)	32	R/W	0000_0000h	5.4.121/311
8000_0820	Interrupt Collector Interrupt Register 112 (HW_ICOLL_INTERRUPT112)	32	R/W	0000_0000h	5.4.122/312
8000_0830	Interrupt Collector Interrupt Register 113 (HW_ICOLL_INTERRUPT113)	32	R/W	0000_0000h	5.4.123/314
8000_0840	Interrupt Collector Interrupt Register 114 (HW_ICOLL_INTERRUPT114)	32	R/W	0000_0000h	5.4.124/315
8000_0850	Interrupt Collector Interrupt Register 115 (HW_ICOLL_INTERRUPT115)	32	R/W	0000_0000h	5.4.125/317
8000_0860	Interrupt Collector Interrupt Register 116 (HW_ICOLL_INTERRUPT116)	32	R/W	0000_0000h	5.4.126/318
8000_0870	Interrupt Collector Interrupt Register 117 (HW_ICOLL_INTERRUPT117)	32	R/W	0000_0000h	5.4.127/319
8000_0880	Interrupt Collector Interrupt Register 118 (HW_ICOLL_INTERRUPT118)	32	R/W	0000_0000h	5.4.128/321
8000_0890	Interrupt Collector Interrupt Register 119 (HW_ICOLL_INTERRUPT119)	32	R/W	0000_0000h	5.4.129/322
8000_08A0	Interrupt Collector Interrupt Register 120 (HW_ICOLL_INTERRUPT120)	32	R/W	0000_0000h	5.4.130/324
8000_08B0	Interrupt Collector Interrupt Register 121 (HW_ICOLL_INTERRUPT121)	32	R/W	0000_0000h	5.4.131/325
8000_08C0	Interrupt Collector Interrupt Register 122 (HW_ICOLL_INTERRUPT122)	32	R/W	0000_0000h	5.4.132/326
8000_08D0	Interrupt Collector Interrupt Register 123 (HW_ICOLL_INTERRUPT123)	32	R/W	0000_0000h	5.4.133/328
8000_08E0	Interrupt Collector Interrupt Register 124 (HW_ICOLL_INTERRUPT124)	32	R/W	0000_0000h	5.4.134/329
8000_08F0	Interrupt Collector Interrupt Register 125 (HW_ICOLL_INTERRUPT125)	32	R/W	0000_0000h	5.4.135/331
8000_0900	Interrupt Collector Interrupt Register 126 (HW_ICOLL_INTERRUPT126)	32	R/W	0000_0000h	5.4.136/332
8000_0910	Interrupt Collector Interrupt Register 127 (HW_ICOLL_INTERRUPT127)	32	R/W	0000_0000h	5.4.137/333
8000_1120	Interrupt Collector Debug Register 0 (HW_ICOLL_DEBUG)	32	R	0000_0000h	5.4.138/335
8000_1130	Interrupt Collector Debug Read Register 0 (HW_ICOLL_DBGREAD0)	32	R	ECA9_4567h	5.4.139/337
8000_1140	Interrupt Collector Debug Read Register 1 (HW_ICOLL_DBGREAD1)	32	R	1356_DA98h	5.4.140/337
8000_1150	Interrupt Collector Debug Flag Register (HW_ICOLL_DBGFLAG)	32	R/W	0000_0000h	5.4.141/338

HW_ICOLL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_1160	Interrupt Collector Debug Read Request Register 0 (HW_ICOLL_DBGREQUEST0)	32	R	0000_0000h	5.4.142/339
8000_1170	Interrupt Collector Debug Read Request Register 1 (HW_ICOLL_DBGREQUEST1)	32	R	0000_0000h	5.4.143/340
8000_1180	Interrupt Collector Debug Read Request Register 2 (HW_ICOLL_DBGREQUEST2)	32	R	0000_0000h	5.4.144/340
8000_1190	Interrupt Collector Debug Read Request Register 3 (HW_ICOLL_DBGREQUEST3)	32	R	0000_0000h	5.4.145/341
8000_11E0	Interrupt Collector Version Register (HW_ICOLL_VERSION)	32	R	0301_0000h	5.4.146/342

5.4.1 Interrupt Collector Interrupt Vector Address Register (HW_ICOLL_VECTOR)

This register is read by the Interrupt Service Routine using a load PC instruction. The priority logic presents the vector address of the next IRQ interrupt to be processed by the CPU. The vector address is held until a new ISR is entered..

HW_ICOLL_VECTOR: 0x000

HW_ICOLL_VECTOR_SET: 0x004

HW_ICOLL_VECTOR_CLR: 0x008

HW_ICOLL_VECTOR_TOG: 0x00C

This register mediates the vectored interrupt collectors interface with the CPU when it enters the IRQ exception trap. The exception trap should have a LDPC instruction from this address.

EXAMPLE

```
LDPC   HW_ICOLL_VECTOR_ADDR; IRQ exception at 0xffff0018
```

Address: HW_ICOLL_VECTOR – 8000_0000h base + 0h offset = 8000_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	IRQVECTOR[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQVECTOR[15:2]														RSRVD1	
W															▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_VECTOR field descriptions

Field	Description
31–2 IRQVECTOR	This register presents the vector address for the interrupt currently active on the CPU IRQ input. Writing to this register notifies the interrupt collector that the interrupt service routine for the current interrupt has been entered (alternatively when ARM_RSE_MODE is set, reading this register is required).
1–0 RSRVD1	Always write zeroes to this field.

5.4.2 Interrupt Collector Level Acknowledge Register (HW_ICOLL_LEVELACK)

The Interrupt Collector Level Acknowledge Register is used by software to indicate the completion of an interrupt on a specific level.

This register is written to advance the ICOLL internal IRQ state machine. It advances from an in-service on a level state to the next pending interrupt level or to the idle state. This register is written at the very end of an interrupt service routine. If nesting is used then the CPU IRQ must be turned on before writing to this register to avoid a race condition in the CPU interrupt hardware. WARNING: the value written to the levelack register is decoded not binary, i.e. 8, 4, 2, 1.

EXAMPLE

```
HW_ICOLL_LEVELACK_WR(HW_ICOLL_LEVELACK__LEVEL3);
```

Address: HW_ICOLL_LEVELACK – 8000_0000h base + 10h offset = 8000_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1																												IRQLEVELACK			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_LEVELACK field descriptions

Field	Description
31–4 RSRVD1	Any value can be written to this bitfield. Writes are ignored.

HW_ICOLL_LEVELACK field descriptions (continued)

Field	Description
3–0 IRQLEVELACK	This bitfield is written by the processor to acknowledge the completion of an interrupt. The value written must correspond to the priority level of the completed interrupt 0x1 **LEVEL0** — level 0 0x2 **LEVEL1** — level 1 0x4 **LEVEL2** — level 2 0x8 **LEVEL3** — level 3

5.4.3 Interrupt Collector Control Register (HW_ICOLL_CTRL)

The Interrupt Collector Control Register provides overall control of interrupts being routed to the CPU. This register is not at offset zero from the block base because that location is needed for single 32 bit instructions to be placed in the exception vector location.

HW_ICOLL_CTRL: 0x020

HW_ICOLL_CTRL_SET: 0x024

HW_ICOLL_CTRL_CLR: 0x028

HW_ICOLL_CTRL_TOG: 0x02C

This register handles the overall control of the interrupt collector, including soft reset and clock gate. In addition, it handles state machine variations like NO_NESTING and ARM read side effect processing on the vector address register.

EXAMPLE

```
HW_ICOLL_CTRL_CLR(BM_ICOLL_CTRL_SFTRST | BM_ICOLL_CTRL_SFTRST );
```

Address: HW_ICOLL_CTRL – 8000_0000h base + 20h offset = 8000_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSRVD3						VECTOR_PITCH			BYPASS_FSM	NO_NESTING	ARM_RSE_MODE	FIQ_FINAL_ENABLE	IRQ_FINAL_ENABLE
W			▨	▨	▨	▨	▨	▨								
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_CTRL field descriptions

Field	Description
31 SFTRST	When set to one, this bit causes a soft reset to the entire interrupt collector. This bit must be turned off for normal operation. 0x0 **RUN** — Allow the interrupt collector to operate normally. 0x1 **IN_RESET** — Hold the interrupt collector in its reset state.
30 CLKGATE	When set to one, this bit causes all clocks within the interrupt collector to be gated off. NOTE: Do not set this bit at the same time as SFTRST. Doing so, causes the soft reset to have no effect. Setting SFTRST will cause the CLKGATE bit to set automatically four clocks later. 0x0 **RUN** — Enable clocks for normal operation of interrupt collector. 0x1 **NO_CLOCKS** — disable clocking within the interrupt collector.
29–24 RSRVD3	Always write zeroes to this bitfield.
23–21 VECTOR_PITCH	When an interrupt occurs one of the 128 input requests becomes the winning bit number, i.e. 127. This bit field selects one of eight constant multiplier values to multiply the winning bit number. The multiplied bit number is added to the vector table base to become the vector address. 0x0 and 0x1 yield a multiplier of 4 bytes. 0x2 yields a multiplier of 8 bytes while 0x3 yields a multiplier of 12 bytes, that is, (8 + 4) bytes per step. 0x0 **DEFAULT_BY4** — one word pitch 0x1 **BY4** — one word pitch 0x2 **BY8** — two word pitch 0x3 **BY12** — three word pitch 0x4 **BY16** — four word pitch 0x5 **BY20** — five word pitch 0x6 **BY24** — six word pitch 0x7 **BY28** — seven word pitch
20 BYPASS_FSM	Set this bit to one to bypass the FSM control of the request holding register and the vector address. With this bit set to one, the vector address register is continuously updated as interrupt requests come in. Turn off all enable bits and walk once through the software interrupts, observing the vector address changes. Set to zero for normal operation. This control is included as a test mode, and is not intended for use by a real application. 0x0 **NORMAL** — Normal 0x1 **BYPASS** — no FSM handshake with CPU
19 NO_NESTING	Set this bit to one disable interrupt level nesting, that is, higher priority interrupt interrupting lower priority. For normal operation, set this bit to zero. 0x0 **NORMAL** — Normal 0x1 **NO_NEST** — no support for interrupt nesting
18 ARM_RSE_ MODE	Set this bit to one enable the ARM-style read side effect associated with the vector address register. In this mode, interrupt inservice is signalled by the read of the HW_ICOLL_VECTOR register to acquire the interrupt vector address. Set this bit to zero for normal operation, in which the ISR signals inservice explicitly by means of a write to the HW_ICOLL_VECTOR register.

HW_ICOLL_CTRL field descriptions (continued)

Field	Description
17 FIQ_FINAL_ ENABLE	Set this bit to one to enable the final FIQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
16 IRQ_FINAL_ ENABLE	Set this bit to one to enable the final IRQ output to the CPU. Set this bit to zero for testing the interrupt collector without causing actual CPU interrupts. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
15–0 RSRVD1	Always write zeroes to this bitfield.

5.4.4 Interrupt Collector Interrupt Vector Base Address Register (HW_ICOLL_VBASE)

This register is used by the priority logic to generate a unique vector address for each of the 80 interrupt request lines coming into the interrupt collector. The vector address is formed by multiplying the interrupt bit number by 4 and adding it to the vector base address.

HW_ICOLL_VBASE: 0x040

HW_ICOLL_VBASE_SET: 0x044

HW_ICOLL_VBASE_CLR: 0x048

HW_ICOLL_VBASE_TOG: 0x04C

This register provides a mechanism to specify the base address of the interrupt vector table. It is used in the computation of the value supplied in HW_ICOLL_VECTOR register.

EXAMPLE

```
HW_ICOLL_VBASE_WR(pInterruptVectorTable);
```

Address: HW_ICOLL_VBASE – 8000_0000h base + 40h offset = 8000_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TABLE_ADDRESS[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TABLE_ADDRESS[15:2]														RSRVD1	
W															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_VBASE field descriptions

Field	Description
31–2 TABLE_ ADDRESS	This bitfield holds the upper 30 bits of the base address of the vector table.
1–0 RSRVD1	Always write zeroes to this bitfield.

5.4.5 Interrupt Collector Status Register (HW_ICOLL_STAT)

Read only view into various internal states, including the Vector number of the current interupt.

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE

```
                if(HW_ICOLL_STAT_VECTOR_NUMBER_READ() == 0x00000017) ISR_vector_23(); //  ISR
 for vector 23 decimal, 17 hex
```

Address: HW_ICOLL_STAT – 8000_0000h base + 70h offset = 8000_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1																									VECTOR_NUMBER						
W	▨																									▨						
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

HW_ICOLL_STAT field descriptions

Field	Description
31–7 RSRVD1	Always write zeroes to this bitfield.
6–0 VECTOR_ NUMBER	Vector number of current interrupt. Multiply by 4 * HW_ICOLL_CTRL[VECTOR_PITCH] and add to vector base address to obtain the value in HW_ICOLL_VECTOR.

5.4.6 Interrupt Collector Raw Interrupt Input Register 0 (HW_ICOLL_RAW0)

The lower 32 interrupt hardware-source states are visible in this read-only register.

HW_ICOLL_RAW0: 0x0A0

HW_ICOLL_RAW0_SET: 0x0A4

HW_ICOLL_RAW0_CLR: 0x0A8

HW_ICOLL_RAW0_TOG: 0x0AC

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. Its purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

Address: HW_ICOLL_RAW0 – 8000_0000h base + A0h offset = 8000_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAW_IRQS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_RAW0 field descriptions

Field	Description
31–0 RAW_IRQS	read-only view of the lower 32 hardware interrupt request bits.

5.4.7 Interrupt Collector Raw Interrupt Input Register 1 (HW_ICOLL_RAW1)

Interrupt hardware-source states 32-63 are visible in this read-only register.

HW_ICOLL_RAW1: 0x0B0

HW_ICOLL_RAW1_SET: 0x0B4

HW_ICOLL_RAW1_CLR: 0x0B8

HW_ICOLL_RAW1_TOG: 0x0BC

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

Address: HW_ICOLL_RAW1 – 8000_0000h base + B0h offset = 8000_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAW_IRQS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_RAW1 field descriptions

Field	Description
31–0 RAW_IRQS	read-only view of hardware interrupt request bits 32-63.

5.4.8 Interrupt Collector Raw Interrupt Input Register 2 (HW_ICOLL_RAW2)

Interrupt hardware-source states 64-95 are visible in this read-only register.

HW_ICOLL_RAW2: 0x0C0

HW_ICOLL_RAW2_SET: 0x0C4

HW_ICOLL_RAW2_CLR: 0x0C8

HW_ICOLL_RAW2_TOG: 0x0CC

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

Address: HW_ICOLL_RAW2 – 8000_0000h base + C0h offset = 8000_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAW_IRQS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_RAW2 field descriptions

Field	Description
31–0 RAW_IRQS	read-only view of hardware interrupt request bits 64-95.

5.4.9 Interrupt Collector Raw Interrupt Input Register 3 (HW_ICOLL_RAW3)

Interrupt hardware-source states 96-127 are visible in this read-only register.

HW_ICOLL_RAW3: 0x0D0

HW_ICOLL_RAW3_SET: 0x0D4

HW_ICOLL_RAW3_CLR: 0x0D8

HW_ICOLL_RAW3_TOG: 0x0DC

This register provides a read-only view of the raw interrupt request lines coming from various parts of the chip. The purpose is to improve diagnostic observability. Note that these only capture the state of hardware interrupt sources.

EXAMPLE

```
ulTest = HW_ICOLL_RAW0.RAW_IRQS;
```

Address: HW_ICOLL_RAW3 – 8000_0000h base + D0h offset = 8000_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RAW_IRQS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_RAW3 field descriptions

Field	Description
31–0 RAW_IRQS	read-only view of hardware interrupt request bits 96-127.

5.4.10 Interrupt Collector Interrupt Register 0 (HW_ICOLL_INTERRUPT0)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT0: 0x120

HW_ICOLL_INTERRUPT0_SET: 0x124

HW_ICOLL_INTERRUPT0_CLR: 0x128

HW_ICOLL_INTERRUPT0_TOG: 0x12C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT0_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT0 – 8000_0000h base + 120h offset = 8000_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT0 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT0 field descriptions (continued)

Field	Description
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.11 Interrupt Collector Interrupt Register 1 (HW_ICOLL_INTERRUPT1)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT1: 0x130

HW_ICOLL_INTERRUPT1_SET: 0x134

HW_ICOLL_INTERRUPT1_CLR: 0x138

HW_ICOLL_INTERRUPT1_TOG: 0x13C

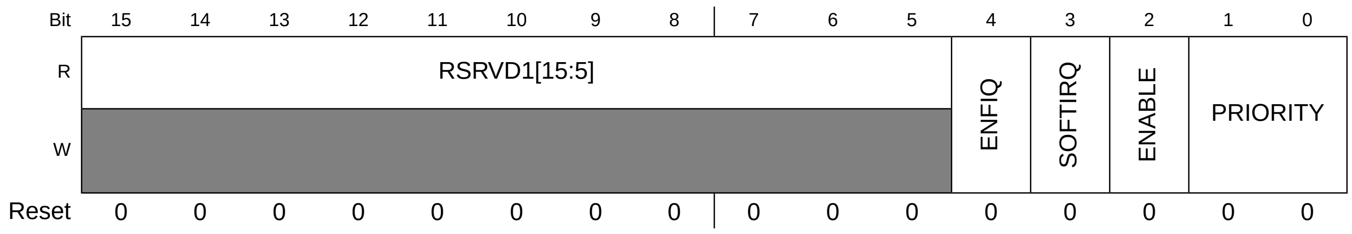
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT1_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT1 – 8000_0000h base + 130h offset = 8000_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT1 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.12 Interrupt Collector Interrupt Register 2 (HW_ICOLL_INTERRUPT2)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT2: 0x140

HW_ICOLL_INTERRUPT2_SET: 0x144

HW_ICOLL_INTERRUPT2_CLR: 0x148

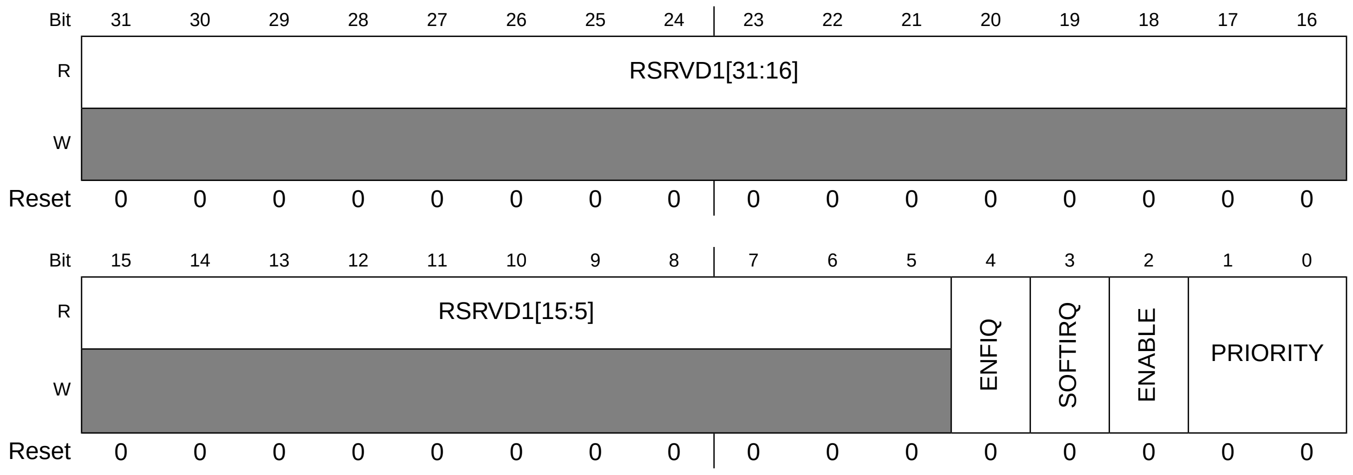
HW_ICOLL_INTERRUPT2_TOG: 0x14C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT2_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT2 – 8000_0000h base + 140h offset = 8000_0140h



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT2 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.13 Interrupt Collector Interrupt Register 3 (HW_ICOLL_INTERRUPT3)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT3: 0x150

HW_ICOLL_INTERRUPT3_SET: 0x154

HW_ICOLL_INTERRUPT3_CLR: 0x158

HW_ICOLL_INTERRUPT3_TOG: 0x15C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT3_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT3 – 8000_0000h base + 150h offset = 8000_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT3 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT3 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.14 Interrupt Collector Interrupt Register 4 (HW_ICOLL_INTERRUPT4)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT4: 0x160

HW_ICOLL_INTERRUPT4_SET: 0x164

HW_ICOLL_INTERRUPT4_CLR: 0x168

HW_ICOLL_INTERRUPT4_TOG: 0x16C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT4_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT4 – 8000_0000h base + 160h offset = 8000_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT4 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.15 Interrupt Collector Interrupt Register 5 (HW_ICOLL_INTERRUPT5)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT5: 0x170

HW_ICOLL_INTERRUPT5_SET: 0x174

HW_ICOLL_INTERRUPT5_CLR: 0x178

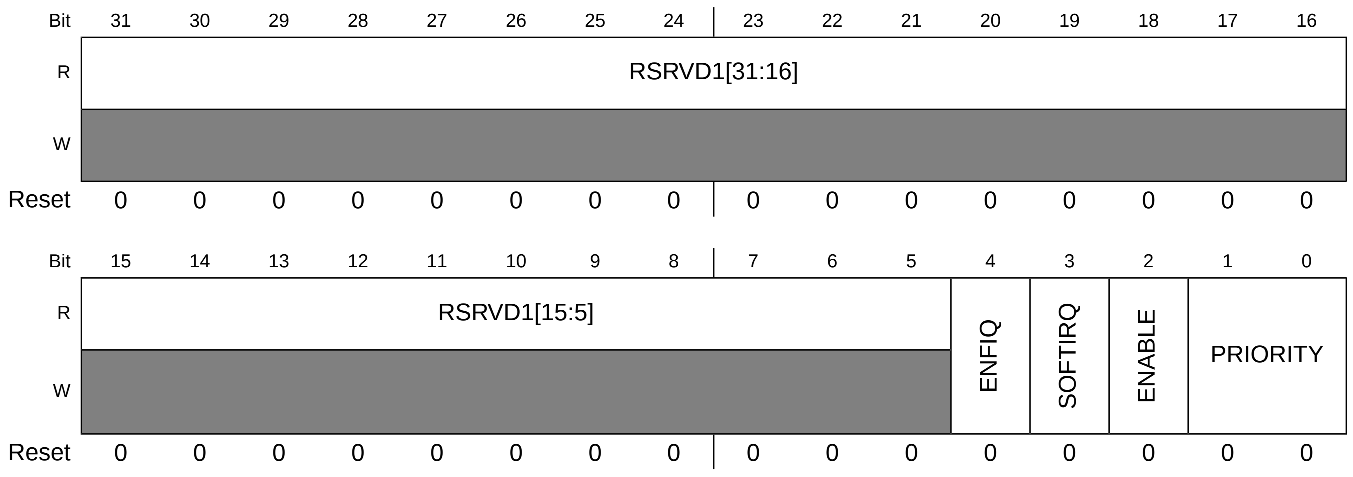
HW_ICOLL_INTERRUPT5_TOG: 0x17C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT5_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT5 – 8000_0000h base + 170h offset = 8000_0170h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT5 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.16 Interrupt Collector Interrupt Register 6 (HW_ICOLL_INTERRUPT6)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT6: 0x180

HW_ICOLL_INTERRUPT6_SET: 0x184

HW_ICOLL_INTERRUPT6_CLR: 0x188

HW_ICOLL_INTERRUPT6_TOG: 0x18C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT6_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT6 – 8000_0000h base + 180h offset = 8000_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT6 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1

HW_ICOLL_INTERRUPT6 field descriptions (continued)

Field	Description
	0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.17 Interrupt Collector Interrupt Register 7 (HW_ICOLL_INTERRUPT7)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT7: 0x190

HW_ICOLL_INTERRUPT7_SET: 0x194

HW_ICOLL_INTERRUPT7_CLR: 0x198

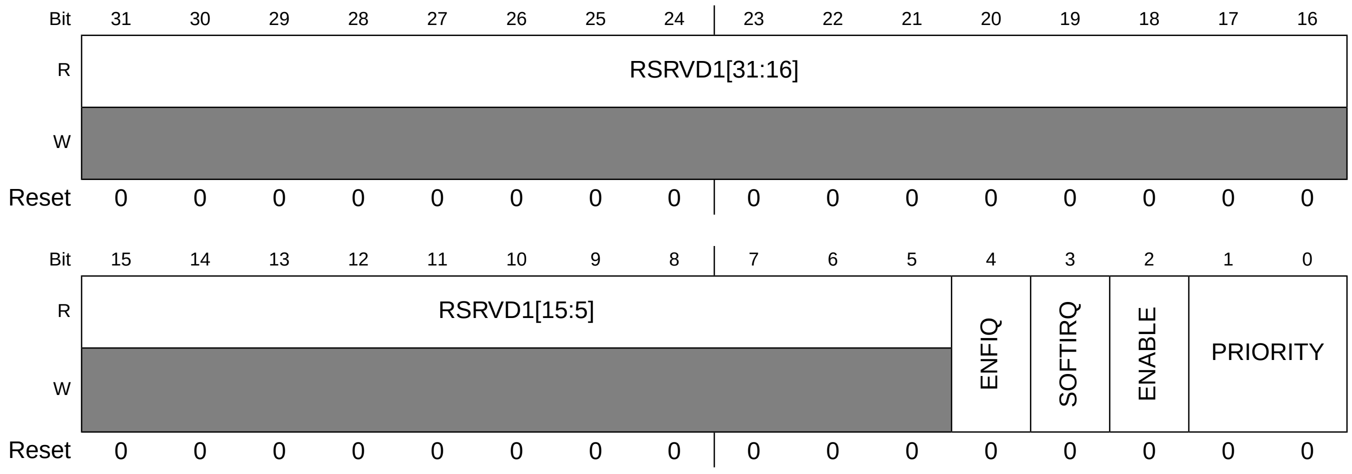
HW_ICOLL_INTERRUPT7_TOG: 0x19C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT7_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT7 – 8000_0000h base + 190h offset = 8000_0190h



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT7 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.18 Interrupt Collector Interrupt Register 8 (HW_ICOLL_INTERRUPT8)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT8: 0x1A0

HW_ICOLL_INTERRUPT8_SET: 0x1A4

HW_ICOLL_INTERRUPT8_CLR: 0x1A8

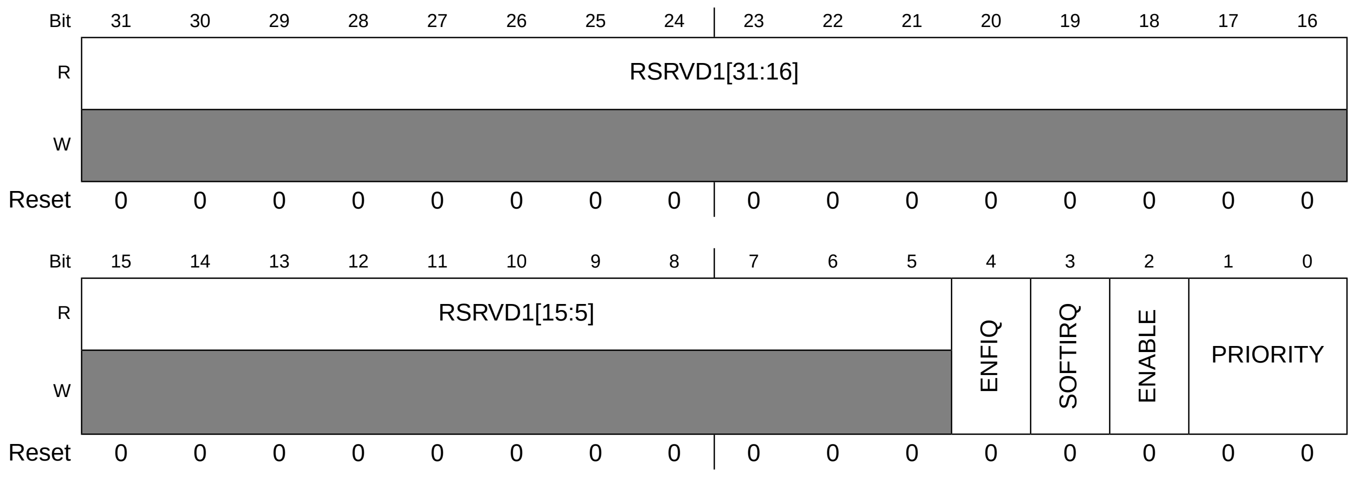
HW_ICOLL_INTERRUPT8_TOG: 0x1AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT8_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT8 – 8000_0000h base + 1A0h offset = 8000_01A0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT8 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.19 Interrupt Collector Interrupt Register 9 (HW_ICOLL_INTERRUPT9)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT9: 0x1B0

HW_ICOLL_INTERRUPT9_SET: 0x1B4

HW_ICOLL_INTERRUPT9_CLR: 0x1B8

HW_ICOLL_INTERRUPT9_TOG: 0x1BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT9_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT9 – 8000_0000h base + 1B0h offset = 8000_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT9 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1

HW_ICOLL_INTERRUPT9 field descriptions (continued)

Field	Description
	0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.20 Interrupt Collector Interrupt Register 10 (HW_ICOLL_INTERRUPT10)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT10: 0x1C0

HW_ICOLL_INTERRUPT10_SET: 0x1C4

HW_ICOLL_INTERRUPT10_CLR: 0x1C8

HW_ICOLL_INTERRUPT10_TOG: 0x1CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT10_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT10 – 8000_0000h base + 1C0h offset = 8000_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT10 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.21 Interrupt Collector Interrupt Register 11 (HW_ICOLL_INTERRUPT11)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT11: 0x1D0

HW_ICOLL_INTERRUPT11_SET: 0x1D4

HW_ICOLL_INTERRUPT11_CLR: 0x1D8

HW_ICOLL_INTERRUPT11_TOG: 0x1DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT11_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT11 – 8000_0000h base + 1D0h offset = 8000_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT11 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.22 Interrupt Collector Interrupt Register 12 (HW_ICOLL_INTERRUPT12)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT12: 0x1E0

HW_ICOLL_INTERRUPT12_SET: 0x1E4

HW_ICOLL_INTERRUPT12_CLR: 0x1E8

HW_ICOLL_INTERRUPT12_TOG: 0x1EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT12_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT12 – 8000_0000h base + 1E0h offset = 8000_01E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT12 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT12 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.23 Interrupt Collector Interrupt Register 13 (HW_ICOLL_INTERRUPT13)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT13: 0x1F0

HW_ICOLL_INTERRUPT13_SET: 0x1F4

HW_ICOLL_INTERRUPT13_CLR: 0x1F8

HW_ICOLL_INTERRUPT13_TOG: 0x1FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT13_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT13 – 8000_0000h base + 1F0h offset = 8000_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT13 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.24 Interrupt Collector Interrupt Register 14 (HW_ICOLL_INTERRUPT14)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT14: 0x200

HW_ICOLL_INTERRUPT14_SET: 0x204

HW_ICOLL_INTERRUPT14_CLR: 0x208

HW_ICOLL_INTERRUPT14_TOG: 0x20C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT14_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT14 – 8000_0000h base + 200h offset = 8000_0200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT14 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.25 Interrupt Collector Interrupt Register 15 (HW_ICOLL_INTERRUPT15)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT15: 0x210

HW_ICOLL_INTERRUPT15_SET: 0x214

HW_ICOLL_INTERRUPT15_CLR: 0x218

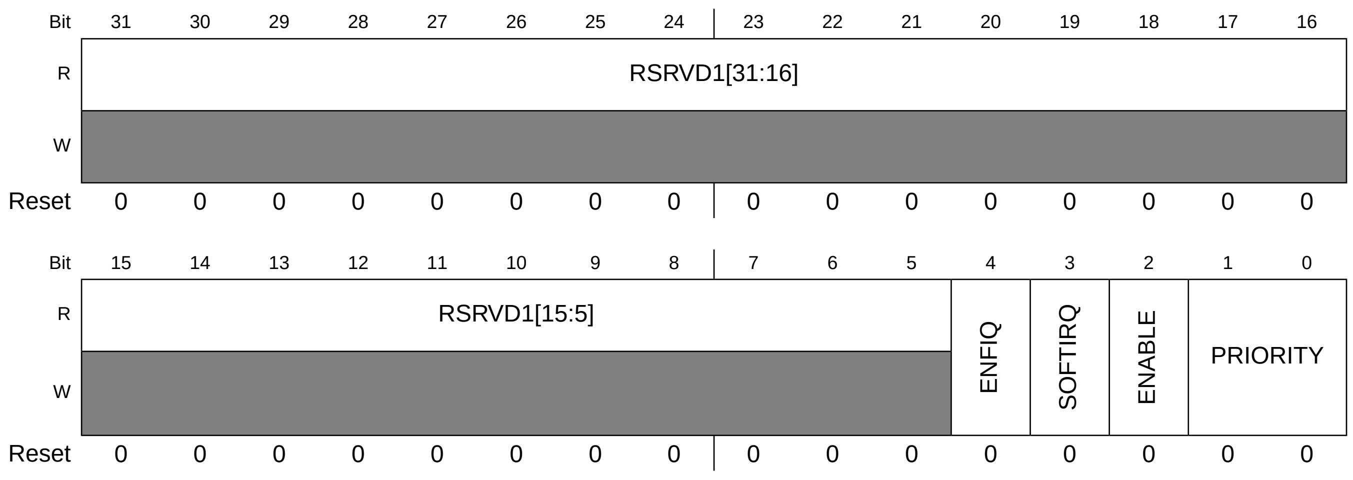
HW_ICOLL_INTERRUPT15_TOG: 0x21C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT15_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT15 – 8000_0000h base + 210h offset = 8000_0210h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT15 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT15 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.26 Interrupt Collector Interrupt Register 16 (HW_ICOLL_INTERRUPT16)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT16: 0x220

HW_ICOLL_INTERRUPT16_SET: 0x224

HW_ICOLL_INTERRUPT16_CLR: 0x228

HW_ICOLL_INTERRUPT16_TOG: 0x22C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT16_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT16 – 8000_0000h base + 220h offset = 8000_0220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT16 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.27 Interrupt Collector Interrupt Register 17 (HW_ICOLL_INTERRUPT17)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT17: 0x230

HW_ICOLL_INTERRUPT17_SET: 0x234

HW_ICOLL_INTERRUPT17_CLR: 0x238

HW_ICOLL_INTERRUPT17_TOG: 0x23C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT17_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT17 – 8000_0000h base + 230h offset = 8000_0230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT17 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT17 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.28 Interrupt Collector Interrupt Register 18 (HW_ICOLL_INTERRUPT18)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT18: 0x240

HW_ICOLL_INTERRUPT18_SET: 0x244

HW_ICOLL_INTERRUPT18_CLR: 0x248

HW_ICOLL_INTERRUPT18_TOG: 0x24C

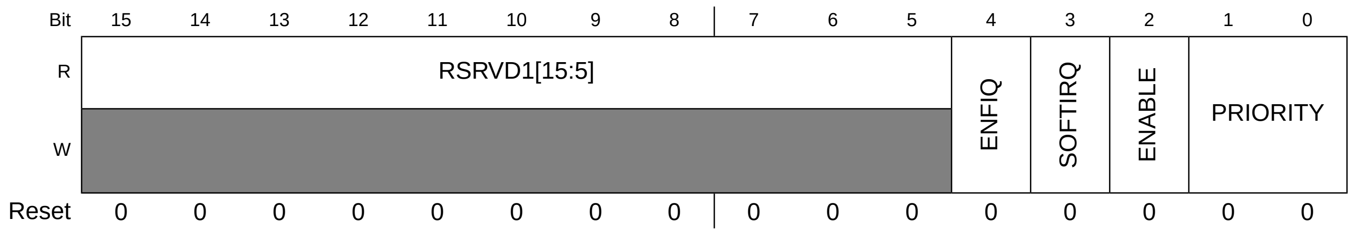
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT18_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT18 – 8000_0000h base + 240h offset = 8000_0240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT18 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.29 Interrupt Collector Interrupt Register 19 (HW_ICOLL_INTERRUPT19)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT19: 0x250

HW_ICOLL_INTERRUPT19_SET: 0x254

HW_ICOLL_INTERRUPT19_CLR: 0x258

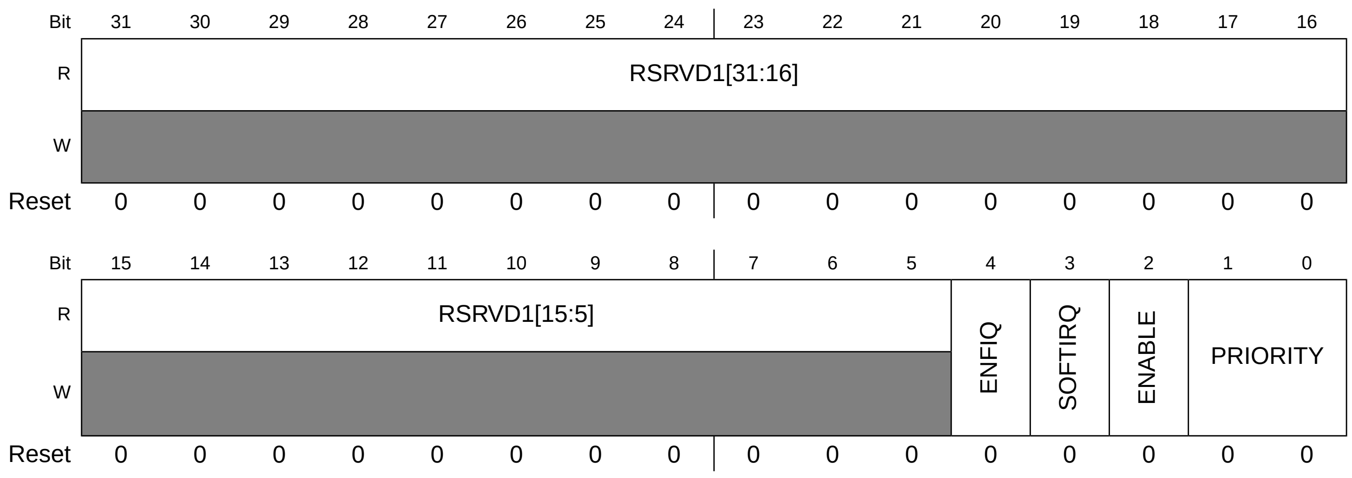
HW_ICOLL_INTERRUPT19_TOG: 0x25C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT19_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT19 – 8000_0000h base + 250h offset = 8000_0250h



HW_ICOLL_INTERRUPT19 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.30 Interrupt Collector Interrupt Register 20 (HW_ICOLL_INTERRUPT20)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT20: 0x260

HW_ICOLL_INTERRUPT20_SET: 0x264

HW_ICOLL_INTERRUPT20_CLR: 0x268

HW_ICOLL_INTERRUPT20_TOG: 0x26C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT20_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT20 – 8000_0000h base + 260h offset = 8000_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT20 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT20 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.31 Interrupt Collector Interrupt Register 21 (HW_ICOLL_INTERRUPT21)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT21: 0x270

HW_ICOLL_INTERRUPT21_SET: 0x274

HW_ICOLL_INTERRUPT21_CLR: 0x278

HW_ICOLL_INTERRUPT21_TOG: 0x27C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT21_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT21 – 8000_0000h base + 270h offset = 8000_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT21 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.32 Interrupt Collector Interrupt Register 22 (HW_ICOLL_INTERRUPT22)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT22: 0x280

HW_ICOLL_INTERRUPT22_SET: 0x284

HW_ICOLL_INTERRUPT22_CLR: 0x288

HW_ICOLL_INTERRUPT22_TOG: 0x28C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT22_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT22 – 8000_0000h base + 280h offset = 8000_0280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT22 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT22 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.33 Interrupt Collector Interrupt Register 23 (HW_ICOLL_INTERRUPT23)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT23: 0x290

HW_ICOLL_INTERRUPT23_SET: 0x294

HW_ICOLL_INTERRUPT23_CLR: 0x298

HW_ICOLL_INTERRUPT23_TOG: 0x29C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT23_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT23 – 8000_0000h base + 290h offset = 8000_0290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT23 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.34 Interrupt Collector Interrupt Register 24 (HW_ICOLL_INTERRUPT24)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT24: 0x2A0

HW_ICOLL_INTERRUPT24_SET: 0x2A4

HW_ICOLL_INTERRUPT24_CLR: 0x2A8

HW_ICOLL_INTERRUPT24_TOG: 0x2AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT24_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT24 – 8000_0000h base + 2A0h offset = 8000_02A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT24 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.35 Interrupt Collector Interrupt Register 25 (HW_ICOLL_INTERRUPT25)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT25: 0x2B0

HW_ICOLL_INTERRUPT25_SET: 0x2B4

HW_ICOLL_INTERRUPT25_CLR: 0x2B8

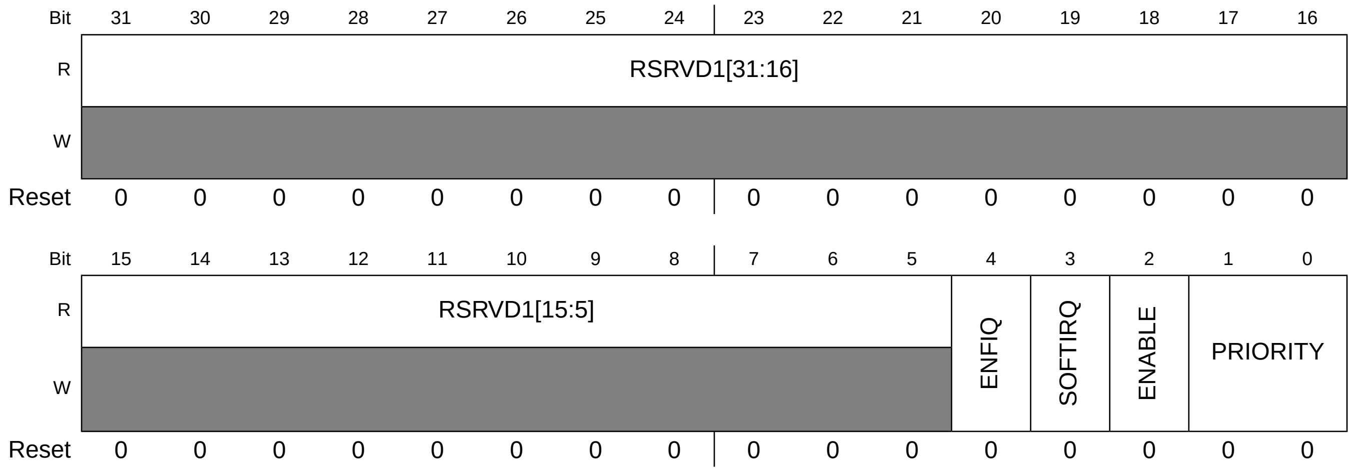
HW_ICOLL_INTERRUPT25_TOG: 0x2BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT25_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT25 – 8000_0000h base + 2B0h offset = 8000_02B0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT25 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT25 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.36 Interrupt Collector Interrupt Register 26 (HW_ICOLL_INTERRUPT26)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT26: 0x2C0

HW_ICOLL_INTERRUPT26_SET: 0x2C4

HW_ICOLL_INTERRUPT26_CLR: 0x2C8

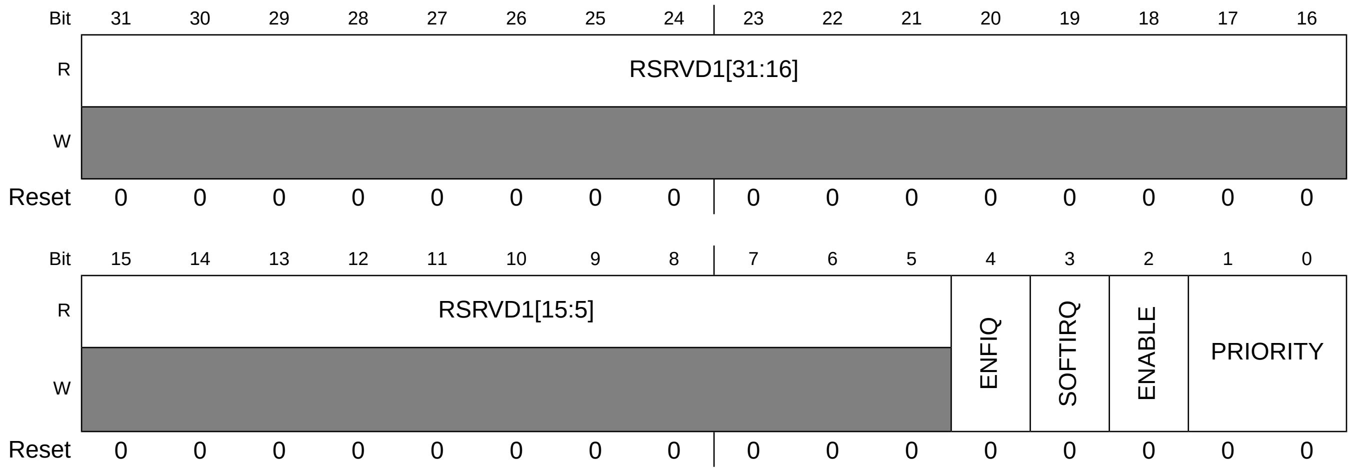
HW_ICOLL_INTERRUPT26_TOG: 0x2CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT26_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT26 – 8000_0000h base + 2C0h offset = 8000_02C0h



HW_ICOLL_INTERRUPT26 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.37 Interrupt Collector Interrupt Register 27 (HW_ICOLL_INTERRUPT27)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT27: 0x2D0

HW_ICOLL_INTERRUPT27_SET: 0x2D4

HW_ICOLL_INTERRUPT27_CLR: 0x2D8

HW_ICOLL_INTERRUPT27_TOG: 0x2DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT27_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT27 – 8000_0000h base + 2D0h offset = 8000_02D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT27 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT27 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.38 Interrupt Collector Interrupt Register 28 (HW_ICOLL_INTERRUPT28)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT28: 0x2E0

HW_ICOLL_INTERRUPT28_SET: 0x2E4

HW_ICOLL_INTERRUPT28_CLR: 0x2E8

HW_ICOLL_INTERRUPT28_TOG: 0x2EC

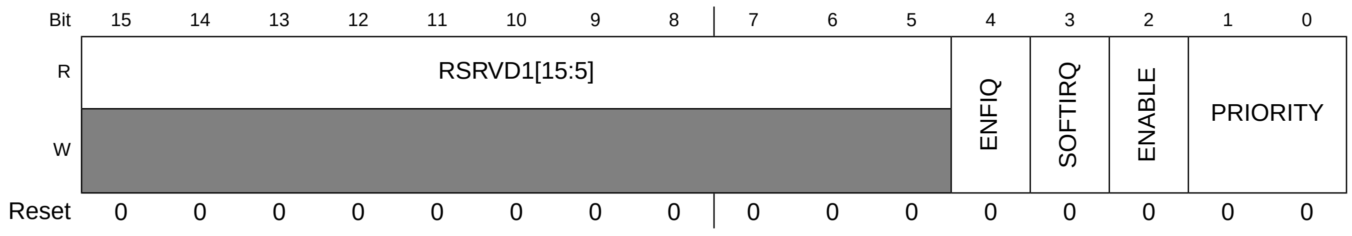
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT28_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT28 – 8000_0000h base + 2E0h offset = 8000_02E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT28 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.39 Interrupt Collector Interrupt Register 29 (HW_ICOLL_INTERRUPT29)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT29: 0x2F0

HW_ICOLL_INTERRUPT29_SET: 0x2F4

HW_ICOLL_INTERRUPT29_CLR: 0x2F8

HW_ICOLL_INTERRUPT29_TOG: 0x2FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT29_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT29 – 8000_0000h base + 2F0h offset = 8000_02F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT29 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.40 Interrupt Collector Interrupt Register 30 (HW_ICOLL_INTERRUPT30)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT30: 0x300

HW_ICOLL_INTERRUPT30_SET: 0x304

HW_ICOLL_INTERRUPT30_CLR: 0x308

HW_ICOLL_INTERRUPT30_TOG: 0x30C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT30_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT30 – 8000_0000h base + 300h offset = 8000_0300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT30 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT30 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.41 Interrupt Collector Interrupt Register 31 (HW_ICOLL_INTERRUPT31)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT31: 0x310

HW_ICOLL_INTERRUPT31_SET: 0x314

HW_ICOLL_INTERRUPT31_CLR: 0x318

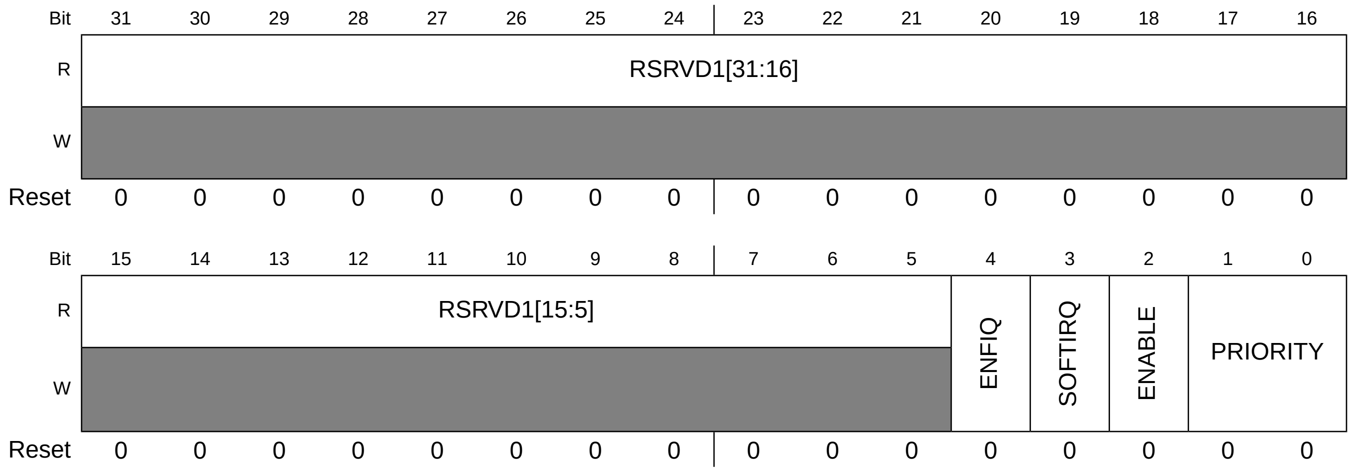
HW_ICOLL_INTERRUPT31_TOG: 0x31C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT31_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT31 – 8000_0000h base + 310h offset = 8000_0310h



HW_ICOLL_INTERRUPT31 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.42 Interrupt Collector Interrupt Register 32 (HW_ICOLL_INTERRUPT32)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT32: 0x320

HW_ICOLL_INTERRUPT32_SET: 0x324

HW_ICOLL_INTERRUPT32_CLR: 0x328

HW_ICOLL_INTERRUPT32_TOG: 0x32C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT32_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT32 – 8000_0000h base + 320h offset = 8000_0320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT32 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT32 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.43 Interrupt Collector Interrupt Register 33 (HW_ICOLL_INTERRUPT33)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT33: 0x330

HW_ICOLL_INTERRUPT33_SET: 0x334

HW_ICOLL_INTERRUPT33_CLR: 0x338

HW_ICOLL_INTERRUPT33_TOG: 0x33C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT33_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT33 – 8000_0000h base + 330h offset = 8000_0330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT33 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.44 Interrupt Collector Interrupt Register 34 (HW_ICOLL_INTERRUPT34)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT34: 0x340

HW_ICOLL_INTERRUPT34_SET: 0x344

HW_ICOLL_INTERRUPT34_CLR: 0x348

HW_ICOLL_INTERRUPT34_TOG: 0x34C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT34_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT34 – 8000_0000h base + 340h offset = 8000_0340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT34 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.45 Interrupt Collector Interrupt Register 35 (HW_ICOLL_INTERRUPT35)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT35: 0x350

HW_ICOLL_INTERRUPT35_SET: 0x354

HW_ICOLL_INTERRUPT35_CLR: 0x358

HW_ICOLL_INTERRUPT35_TOG: 0x35C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT35_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT35 – 8000_0000h base + 350h offset = 8000_0350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT35 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT35 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.46 Interrupt Collector Interrupt Register 36 (HW_ICOLL_INTERRUPT36)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT36: 0x360

HW_ICOLL_INTERRUPT36_SET: 0x364

HW_ICOLL_INTERRUPT36_CLR: 0x368

HW_ICOLL_INTERRUPT36_TOG: 0x36C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT36_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT36 – 8000_0000h base + 360h offset = 8000_0360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT36 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.47 Interrupt Collector Interrupt Register 37 (HW_ICOLL_INTERRUPT37)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT37: 0x370

HW_ICOLL_INTERRUPT37_SET: 0x374

HW_ICOLL_INTERRUPT37_CLR: 0x378

HW_ICOLL_INTERRUPT37_TOG: 0x37C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT37_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT37 – 8000_0000h base + 370h offset = 8000_0370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT37 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT37 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.48 Interrupt Collector Interrupt Register 38 (HW_ICOLL_INTERRUPT38)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT38: 0x380

HW_ICOLL_INTERRUPT38_SET: 0x384

HW_ICOLL_INTERRUPT38_CLR: 0x388

HW_ICOLL_INTERRUPT38_TOG: 0x38C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT38_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT38 – 8000_0000h base + 380h offset = 8000_0380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT38 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.49 Interrupt Collector Interrupt Register 39 (HW_ICOLL_INTERRUPT39)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT39: 0x390

HW_ICOLL_INTERRUPT39_SET: 0x394

HW_ICOLL_INTERRUPT39_CLR: 0x398

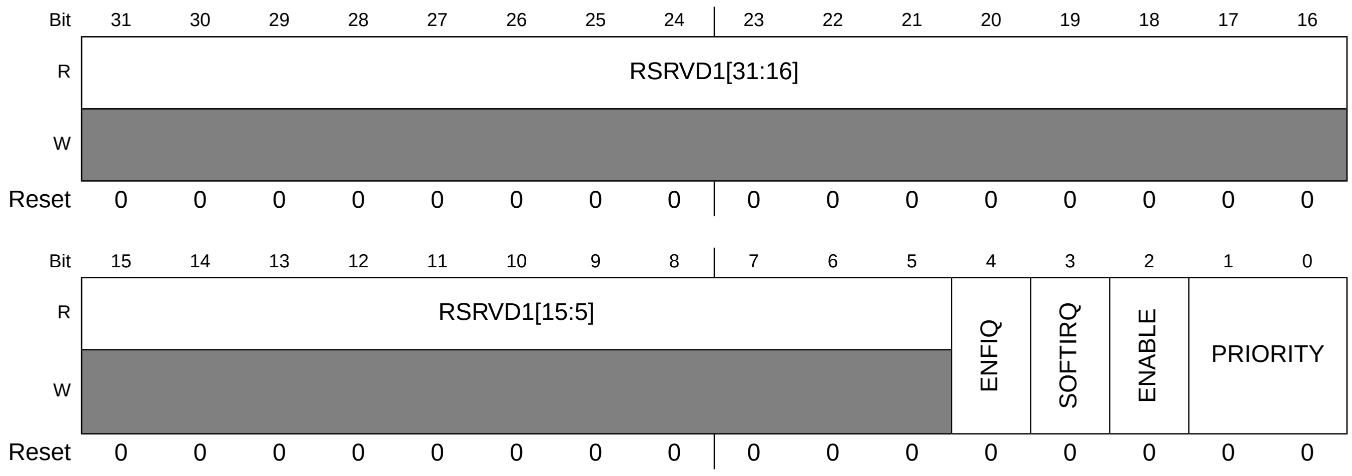
HW_ICOLL_INTERRUPT39_TOG: 0x39C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT39_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT39 – 8000_0000h base + 390h offset = 8000_0390h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT39 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.50 Interrupt Collector Interrupt Register 40 (HW_ICOLL_INTERRUPT40)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT40: 0x3A0

HW_ICOLL_INTERRUPT40_SET: 0x3A4

HW_ICOLL_INTERRUPT40_CLR: 0x3A8

HW_ICOLL_INTERRUPT40_TOG: 0x3AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT40_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT40 – 8000_0000h base + 3A0h offset = 8000_03A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT40 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT40 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.51 Interrupt Collector Interrupt Register 41 (HW_ICOLL_INTERRUPT41)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT41: 0x3B0

HW_ICOLL_INTERRUPT41_SET: 0x3B4

HW_ICOLL_INTERRUPT41_CLR: 0x3B8

HW_ICOLL_INTERRUPT41_TOG: 0x3BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT41_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT41 – 8000_0000h base + 3B0h offset = 8000_03B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT41 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.52 Interrupt Collector Interrupt Register 42 (HW_ICOLL_INTERRUPT42)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT42: 0x3C0

HW_ICOLL_INTERRUPT42_SET: 0x3C4

HW_ICOLL_INTERRUPT42_CLR: 0x3C8

HW_ICOLL_INTERRUPT42_TOG: 0x3CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT42_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT42 – 8000_0000h base + 3C0h offset = 8000_03C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT42 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT42 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.53 Interrupt Collector Interrupt Register 43 (HW_ICOLL_INTERRUPT43)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT43: 0x3D0

HW_ICOLL_INTERRUPT43_SET: 0x3D4

HW_ICOLL_INTERRUPT43_CLR: 0x3D8

HW_ICOLL_INTERRUPT43_TOG: 0x3DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT43_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT43 – 8000_0000h base + 3D0h offset = 8000_03D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT43 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.54 Interrupt Collector Interrupt Register 44 (HW_ICOLL_INTERRUPT44)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT44: 0x3E0

HW_ICOLL_INTERRUPT44_SET: 0x3E4

HW_ICOLL_INTERRUPT44_CLR: 0x3E8

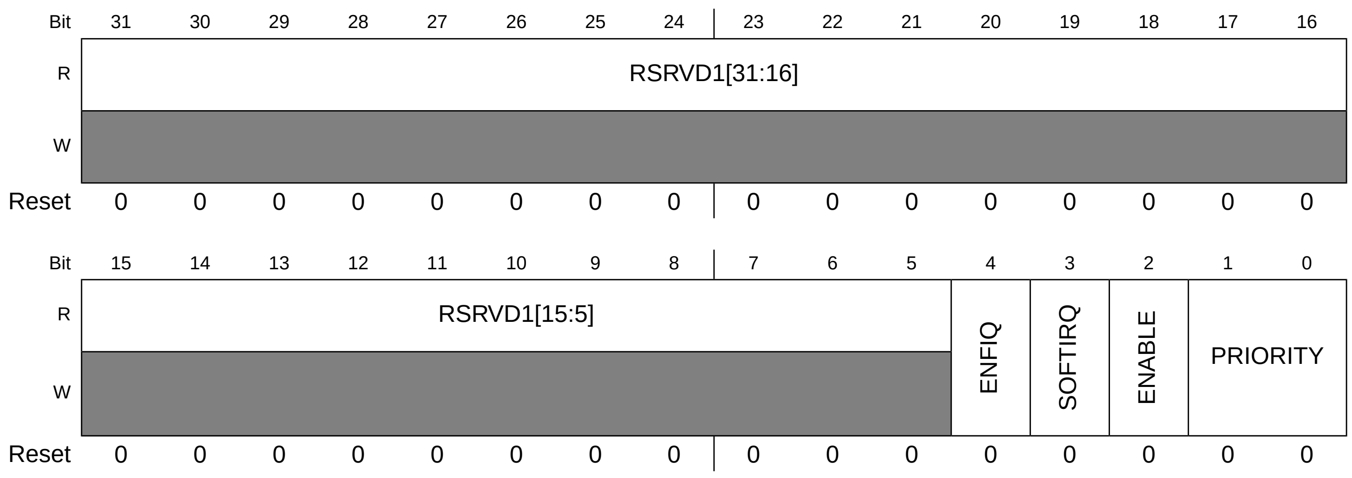
HW_ICOLL_INTERRUPT44_TOG: 0x3EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT44_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT44 – 8000_0000h base + 3E0h offset = 8000_03E0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT44 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.55 Interrupt Collector Interrupt Register 45 (HW_ICOLL_INTERRUPT45)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT45: 0x3F0

HW_ICOLL_INTERRUPT45_SET: 0x3F4

HW_ICOLL_INTERRUPT45_CLR: 0x3F8

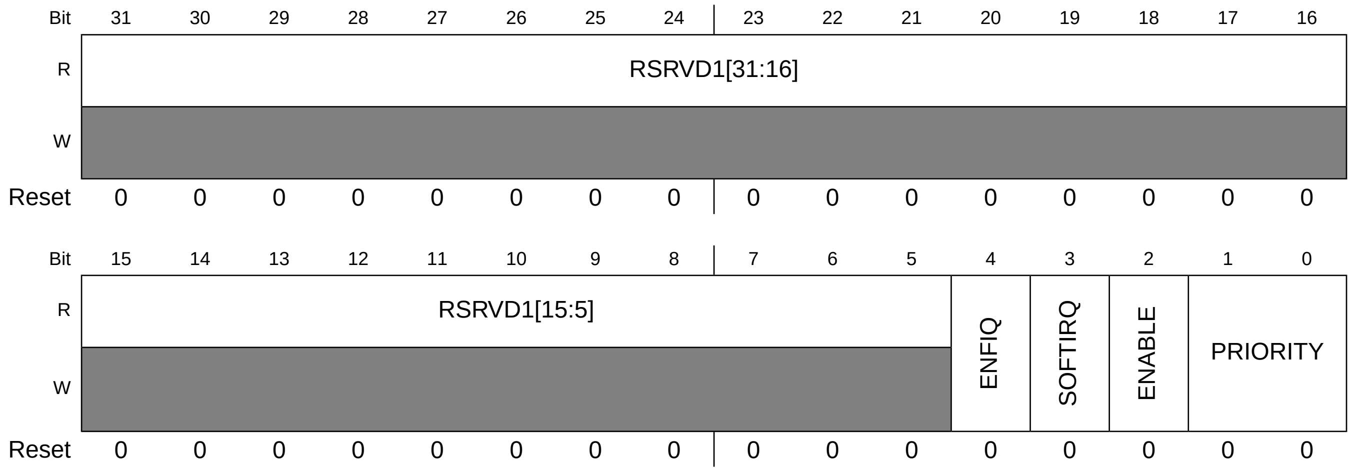
HW_ICOLL_INTERRUPT45_TOG: 0x3FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT45_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT45 – 8000_0000h base + 3F0h offset = 8000_03F0h



HW_ICOLL_INTERRUPT45 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT45 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.56 Interrupt Collector Interrupt Register 46 (HW_ICOLL_INTERRUPT46)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT46: 0x400

HW_ICOLL_INTERRUPT46_SET: 0x404

HW_ICOLL_INTERRUPT46_CLR: 0x408

HW_ICOLL_INTERRUPT46_TOG: 0x40C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT46_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT46 – 8000_0000h base + 400h offset = 8000_ 0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT46 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.57 Interrupt Collector Interrupt Register 47 (HW_ICOLL_INTERRUPT47)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT47: 0x410

HW_ICOLL_INTERRUPT47_SET: 0x414

HW_ICOLL_INTERRUPT47_CLR: 0x418

HW_ICOLL_INTERRUPT47_TOG: 0x41C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT47_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT47 – 8000_0000h base + 410h offset = 8000_0410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT47 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT47 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.58 Interrupt Collector Interrupt Register 48 (HW_ICOLL_INTERRUPT48)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT48: 0x420

HW_ICOLL_INTERRUPT48_SET: 0x424

HW_ICOLL_INTERRUPT48_CLR: 0x428

HW_ICOLL_INTERRUPT48_TOG: 0x42C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT48_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT48 – 8000_0000h base + 420h offset = 8000_0420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT48 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.59 Interrupt Collector Interrupt Register 49 (HW_ICOLL_INTERRUPT49)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT49: 0x430

HW_ICOLL_INTERRUPT49_SET: 0x434

HW_ICOLL_INTERRUPT49_CLR: 0x438

HW_ICOLL_INTERRUPT49_TOG: 0x43C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT49_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT49 – 8000_0000h base + 430h offset = 8000_0430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT49 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.60 Interrupt Collector Interrupt Register 50 (HW_ICOLL_INTERRUPT50)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT50: 0x440

HW_ICOLL_INTERRUPT50_SET: 0x444

HW_ICOLL_INTERRUPT50_CLR: 0x448

HW_ICOLL_INTERRUPT50_TOG: 0x44C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT50_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT50 – 8000_0000h base + 440h offset = 8000_0440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT50 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT50 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.61 Interrupt Collector Interrupt Register 51 (HW_ICOLL_INTERRUPT51)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT51: 0x450

HW_ICOLL_INTERRUPT51_SET: 0x454

HW_ICOLL_INTERRUPT51_CLR: 0x458

HW_ICOLL_INTERRUPT51_TOG: 0x45C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT51_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT51 – 8000_0000h base + 450h offset = 8000_0450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT51 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.62 Interrupt Collector Interrupt Register 52 (HW_ICOLL_INTERRUPT52)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT52: 0x460

HW_ICOLL_INTERRUPT52_SET: 0x464

HW_ICOLL_INTERRUPT52_CLR: 0x468

HW_ICOLL_INTERRUPT52_TOG: 0x46C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT52_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT52 – 8000_0000h base + 460h offset = 8000_0460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT52 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT52 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.63 Interrupt Collector Interrupt Register 53 (HW_ICOLL_INTERRUPT53)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT53: 0x470

HW_ICOLL_INTERRUPT53_SET: 0x474

HW_ICOLL_INTERRUPT53_CLR: 0x478

HW_ICOLL_INTERRUPT53_TOG: 0x47C

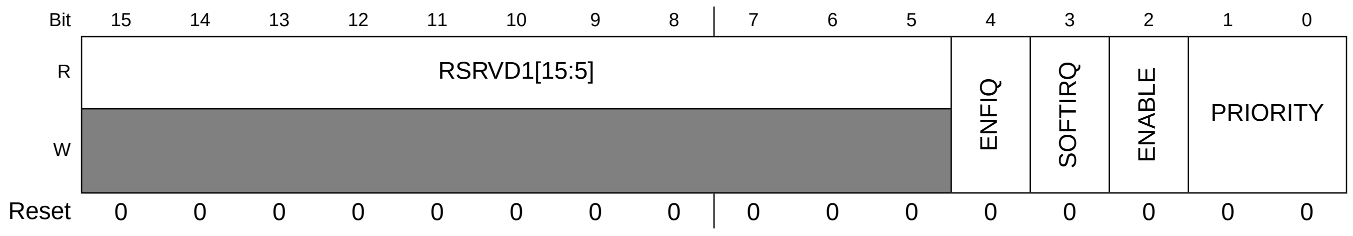
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT53_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT53 – 8000_0000h base + 470h offset = 8000_0470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT53 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.64 Interrupt Collector Interrupt Register 54 (HW_ICOLL_INTERRUPT54)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT54: 0x480

HW_ICOLL_INTERRUPT54_SET: 0x484

HW_ICOLL_INTERRUPT54_CLR: 0x488

HW_ICOLL_INTERRUPT54_TOG: 0x48C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT54_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT54 – 8000_0000h base + 480h offset = 8000_0480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT54 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.65 Interrupt Collector Interrupt Register 55 (HW_ICOLL_INTERRUPT55)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT55: 0x490

HW_ICOLL_INTERRUPT55_SET: 0x494

HW_ICOLL_INTERRUPT55_CLR: 0x498

HW_ICOLL_INTERRUPT55_TOG: 0x49C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT55_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT55 – 8000_0000h base + 490h offset = 8000_0490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT55 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT55 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.66 Interrupt Collector Interrupt Register 56 (HW_ICOLL_INTERRUPT56)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT56: 0x4A0

HW_ICOLL_INTERRUPT56_SET: 0x4A4

HW_ICOLL_INTERRUPT56_CLR: 0x4A8

HW_ICOLL_INTERRUPT56_TOG: 0x4AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT56_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT56 – 8000_0000h base + 4A0h offset = 8000_04A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT56 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.67 Interrupt Collector Interrupt Register 57 (HW_ICOLL_INTERRUPT57)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT57: 0x4B0

HW_ICOLL_INTERRUPT57_SET: 0x4B4

HW_ICOLL_INTERRUPT57_CLR: 0x4B8

HW_ICOLL_INTERRUPT57_TOG: 0x4BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT57_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT57 – 8000_0000h base + 4B0h offset = 8000_04B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT57 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT57 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.68 Interrupt Collector Interrupt Register 58 (HW_ICOLL_INTERRUPT58)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT58: 0x4C0

HW_ICOLL_INTERRUPT58_SET: 0x4C4

HW_ICOLL_INTERRUPT58_CLR: 0x4C8

HW_ICOLL_INTERRUPT58_TOG: 0x4CC

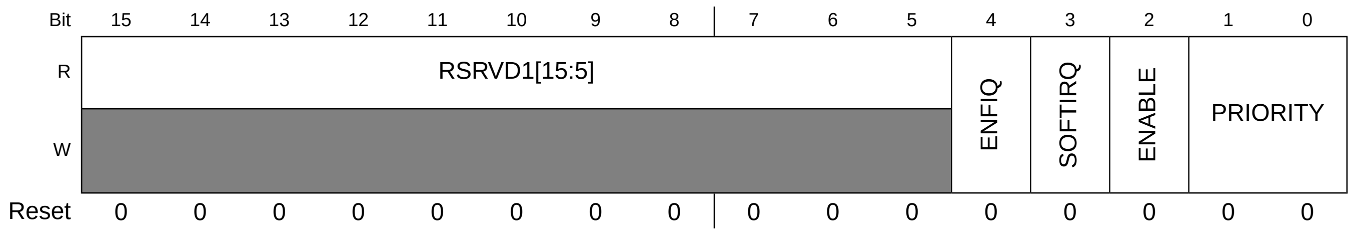
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT58_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT58 – 8000_0000h base + 4C0h offset = 8000_04C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT58 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.69 Interrupt Collector Interrupt Register 59 (HW_ICOLL_INTERRUPT59)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT59: 0x4D0

HW_ICOLL_INTERRUPT59_SET: 0x4D4

HW_ICOLL_INTERRUPT59_CLR: 0x4D8

HW_ICOLL_INTERRUPT59_TOG: 0x4DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT59_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT59 – 8000_0000h base + 4D0h offset = 8000_04D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT59 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.70 Interrupt Collector Interrupt Register 60 (HW_ICOLL_INTERRUPT60)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT60: 0x4E0

HW_ICOLL_INTERRUPT60_SET: 0x4E4

HW_ICOLL_INTERRUPT60_CLR: 0x4E8

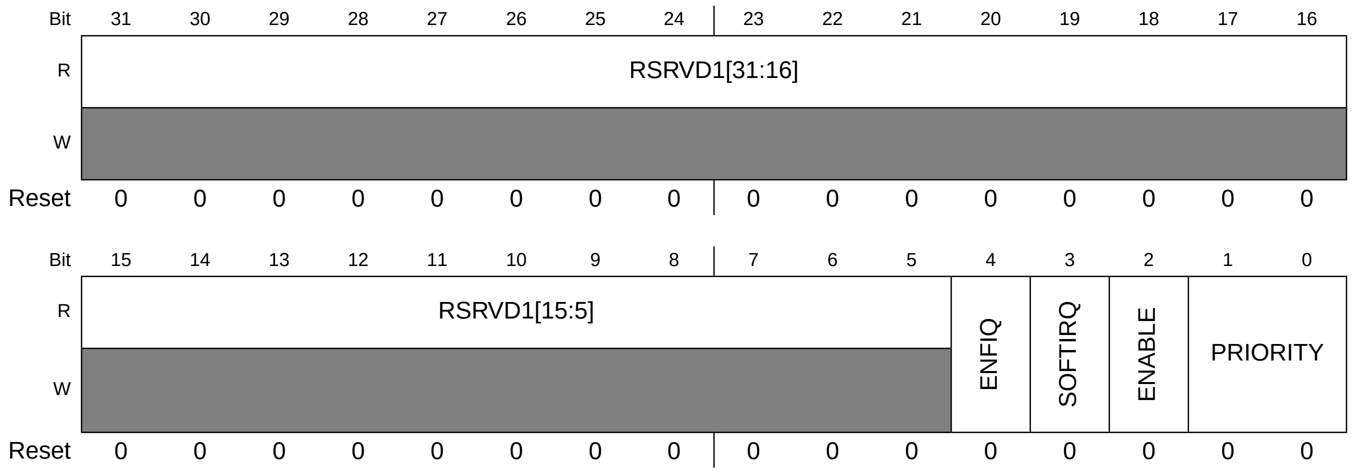
HW_ICOLL_INTERRUPT60_TOG: 0x4EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT60_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT60 – 8000_0000h base + 4E0h offset = 8000_04E0h



HW_ICOLL_INTERRUPT60 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT60 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.71 Interrupt Collector Interrupt Register 61 (HW_ICOLL_INTERRUPT61)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT61: 0x4F0

HW_ICOLL_INTERRUPT61_SET: 0x4F4

HW_ICOLL_INTERRUPT61_CLR: 0x4F8

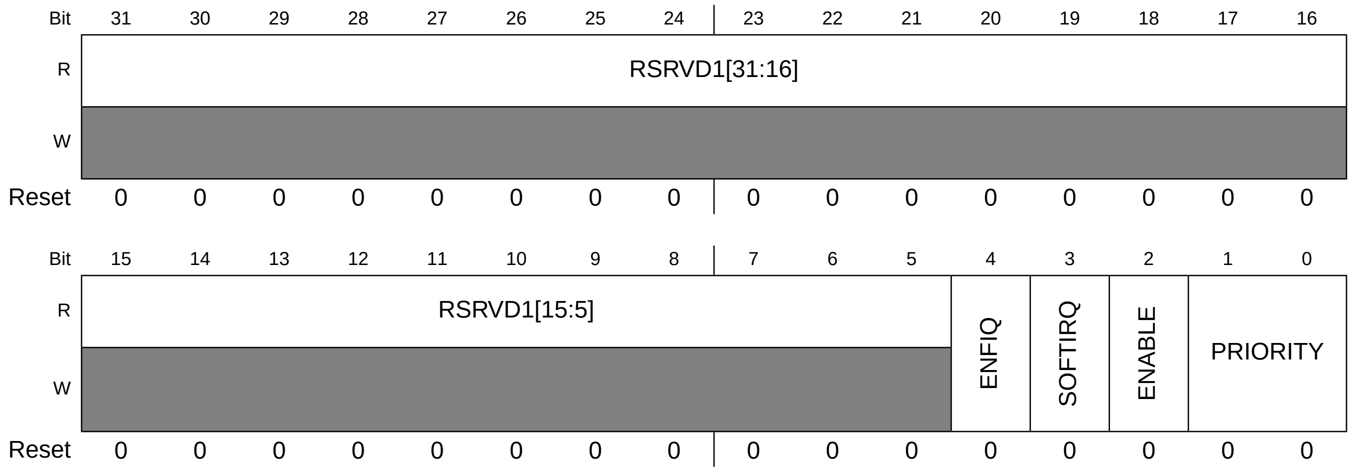
HW_ICOLL_INTERRUPT61_TOG: 0x4FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT61_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT61 – 8000_0000h base + 4F0h offset = 8000_04F0h



HW_ICOLL_INTERRUPT61 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.72 Interrupt Collector Interrupt Register 62 (HW_ICOLL_INTERRUPT62)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT62: 0x500

HW_ICOLL_INTERRUPT62_SET: 0x504

HW_ICOLL_INTERRUPT62_CLR: 0x508

HW_ICOLL_INTERRUPT62_TOG: 0x50C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT62_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT62 – 8000_0000h base + 500h offset = 8000_0500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT62 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT62 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.73 Interrupt Collector Interrupt Register 63 (HW_ICOLL_INTERRUPT63)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT63: 0x510

HW_ICOLL_INTERRUPT63_SET: 0x514

HW_ICOLL_INTERRUPT63_CLR: 0x518

HW_ICOLL_INTERRUPT63_TOG: 0x51C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT63_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT63 – 8000_0000h base + 510h offset = 8000_0510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT63 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.74 Interrupt Collector Interrupt Register 64 (HW_ICOLL_INTERRUPT64)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT64: 0x520

HW_ICOLL_INTERRUPT64_SET: 0x524

HW_ICOLL_INTERRUPT64_CLR: 0x528

HW_ICOLL_INTERRUPT64_TOG: 0x52C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT64_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT64 – 8000_0000h base + 520h offset = 8000_0520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT64 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.75 Interrupt Collector Interrupt Register 65 (HW_ICOLL_INTERRUPT65)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT65: 0x530

HW_ICOLL_INTERRUPT65_SET: 0x534

HW_ICOLL_INTERRUPT65_CLR: 0x538

HW_ICOLL_INTERRUPT65_TOG: 0x53C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT65_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT65 – 8000_0000h base + 530h offset = 8000_0530h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT65 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT65 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.76 Interrupt Collector Interrupt Register 66 (HW_ICOLL_INTERRUPT66)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT66: 0x540

HW_ICOLL_INTERRUPT66_SET: 0x544

HW_ICOLL_INTERRUPT66_CLR: 0x548

HW_ICOLL_INTERRUPT66_TOG: 0x54C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT66_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT66 – 8000_0000h base + 540h offset = 8000_0540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT66 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.77 Interrupt Collector Interrupt Register 67 (HW_ICOLL_INTERRUPT67)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT67: 0x550

HW_ICOLL_INTERRUPT67_SET: 0x554

HW_ICOLL_INTERRUPT67_CLR: 0x558

HW_ICOLL_INTERRUPT67_TOG: 0x55C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT67_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT67 – 8000_0000h base + 550h offset = 8000_0550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT67 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT67 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.78 Interrupt Collector Interrupt Register 68 (HW_ICOLL_INTERRUPT68)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT68: 0x560

HW_ICOLL_INTERRUPT68_SET: 0x564

HW_ICOLL_INTERRUPT68_CLR: 0x568

HW_ICOLL_INTERRUPT68_TOG: 0x56C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT68_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT68 – 8000_0000h base + 560h offset = 8000_0560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT68 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.79 Interrupt Collector Interrupt Register 69 (HW_ICOLL_INTERRUPT69)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT69: 0x570

HW_ICOLL_INTERRUPT69_SET: 0x574

HW_ICOLL_INTERRUPT69_CLR: 0x578

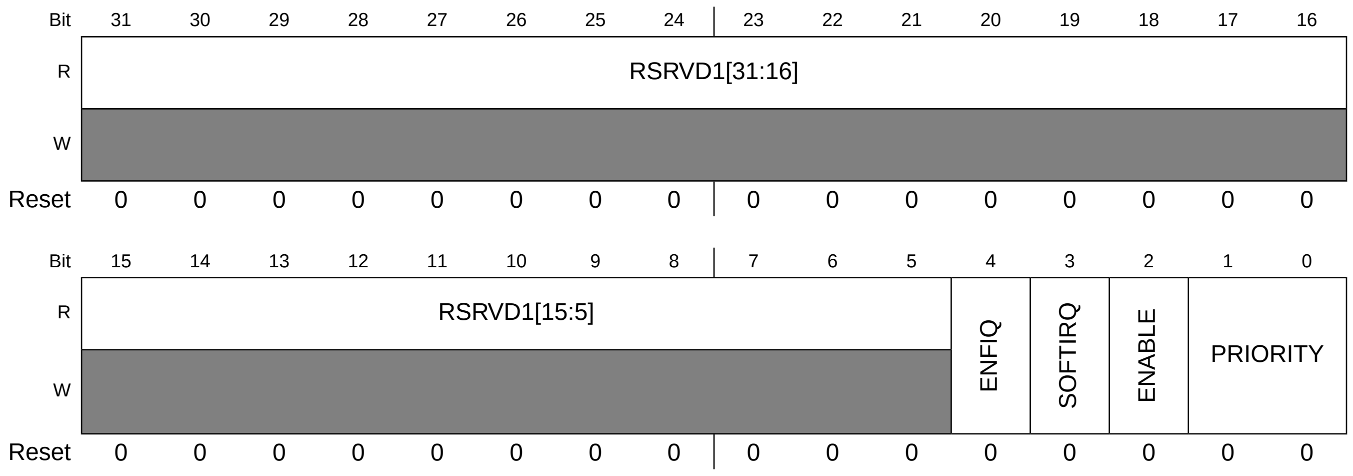
HW_ICOLL_INTERRUPT69_TOG: 0x57C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT69_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT69 – 8000_0000h base + 570h offset = 8000_0570h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT69 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.80 Interrupt Collector Interrupt Register 70 (HW_ICOLL_INTERRUPT70)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT70: 0x580

HW_ICOLL_INTERRUPT70_SET: 0x584

HW_ICOLL_INTERRUPT70_CLR: 0x588

HW_ICOLL_INTERRUPT70_TOG: 0x58C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT70_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT70 – 8000_0000h base + 580h offset = 8000_0580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT70 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT70 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.81 Interrupt Collector Interrupt Register 71 (HW_ICOLL_INTERRUPT71)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT71: 0x590

HW_ICOLL_INTERRUPT71_SET: 0x594

HW_ICOLL_INTERRUPT71_CLR: 0x598

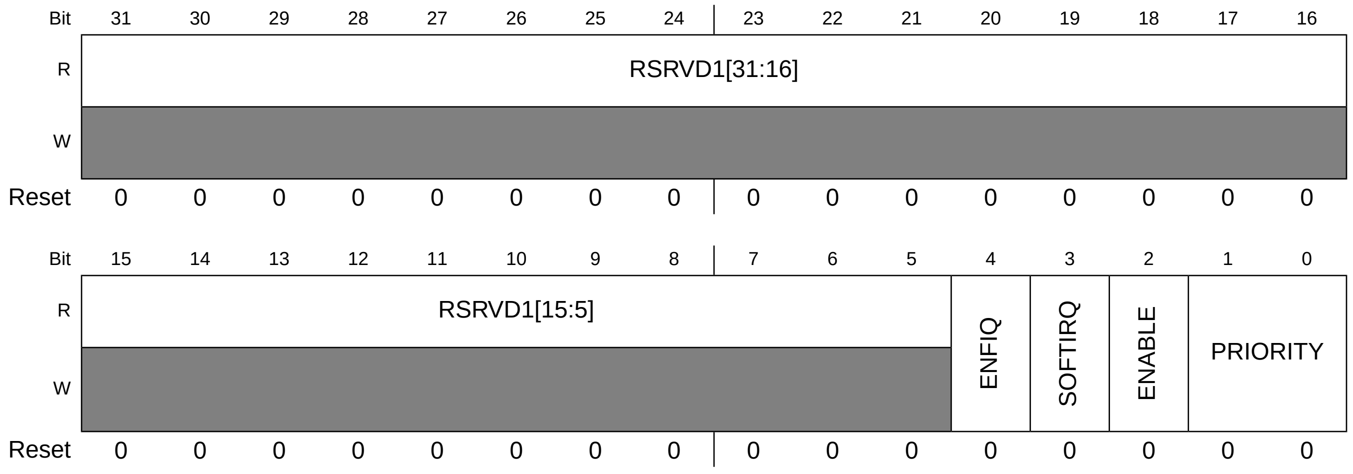
HW_ICOLL_INTERRUPT71_TOG: 0x59C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT71_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT71 – 8000_0000h base + 590h offset = 8000_0590h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT71 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.82 Interrupt Collector Interrupt Register 72 (HW_ICOLL_INTERRUPT72)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT72: 0x5A0

HW_ICOLL_INTERRUPT72_SET: 0x5A4

HW_ICOLL_INTERRUPT72_CLR: 0x5A8

HW_ICOLL_INTERRUPT72_TOG: 0x5AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT72_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT72 – 8000_0000h base + 5A0h offset = 8000_05A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT72 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT72 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.83 Interrupt Collector Interrupt Register 73 (HW_ICOLL_INTERRUPT73)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT73: 0x5B0

HW_ICOLL_INTERRUPT73_SET: 0x5B4

HW_ICOLL_INTERRUPT73_CLR: 0x5B8

HW_ICOLL_INTERRUPT73_TOG: 0x5BC

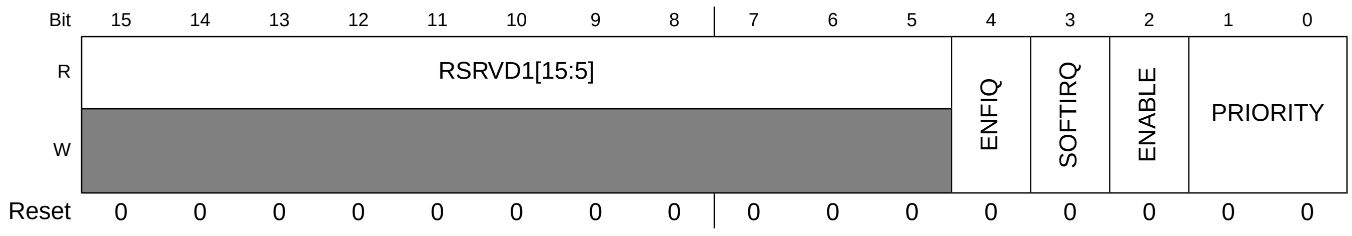
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT73_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT73 – 8000_0000h base + 5B0h offset = 8000_05B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT73 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.84 Interrupt Collector Interrupt Register 74 (HW_ICOLL_INTERRUPT74)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT74: 0x5C0

HW_ICOLL_INTERRUPT74_SET: 0x5C4

HW_ICOLL_INTERRUPT74_CLR: 0x5C8

HW_ICOLL_INTERRUPT74_TOG: 0x5CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT74_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT74 – 8000_0000h base + 5C0h offset = 8000_05C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT74 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.85 Interrupt Collector Interrupt Register 75 (HW_ICOLL_INTERRUPT75)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT75: 0x5D0

HW_ICOLL_INTERRUPT75_SET: 0x5D4

HW_ICOLL_INTERRUPT75_CLR: 0x5D8

HW_ICOLL_INTERRUPT75_TOG: 0x5DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT75_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT75 – 8000_0000h base + 5D0h offset = 8000_05D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT75 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT75 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.86 Interrupt Collector Interrupt Register 76 (HW_ICOLL_INTERRUPT76)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT76: 0x5E0

HW_ICOLL_INTERRUPT76_SET: 0x5E4

HW_ICOLL_INTERRUPT76_CLR: 0x5E8

HW_ICOLL_INTERRUPT76_TOG: 0x5EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT76_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT76 – 8000_0000h base + 5E0h offset = 8000_05E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT76 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.87 Interrupt Collector Interrupt Register 77 (HW_ICOLL_INTERRUPT77)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT77: 0x5F0

HW_ICOLL_INTERRUPT77_SET: 0x5F4

HW_ICOLL_INTERRUPT77_CLR: 0x5F8

HW_ICOLL_INTERRUPT77_TOG: 0x5FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT77_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT77 – 8000_0000h base + 5F0h offset = 8000_05F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT77 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT77 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.88 Interrupt Collector Interrupt Register 78 (HW_ICOLL_INTERRUPT78)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT78: 0x600

HW_ICOLL_INTERRUPT78_SET: 0x604

HW_ICOLL_INTERRUPT78_CLR: 0x608

HW_ICOLL_INTERRUPT78_TOG: 0x60C

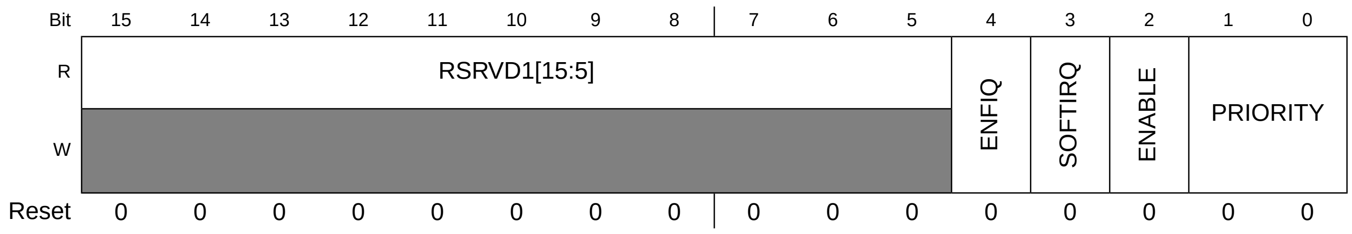
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT78_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT78 – 8000_0000h base + 600h offset = 8000_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT78 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.89 Interrupt Collector Interrupt Register 79 (HW_ICOLL_INTERRUPT79)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT79: 0x610

HW_ICOLL_INTERRUPT79_SET: 0x614

HW_ICOLL_INTERRUPT79_CLR: 0x618

HW_ICOLL_INTERRUPT79_TOG: 0x61C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT79_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT79 – 8000_0000h base + 610h offset = 8000_0610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT79 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.90 Interrupt Collector Interrupt Register 80 (HW_ICOLL_INTERRUPT80)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT80: 0x620

HW_ICOLL_INTERRUPT80_SET: 0x624

HW_ICOLL_INTERRUPT80_CLR: 0x628

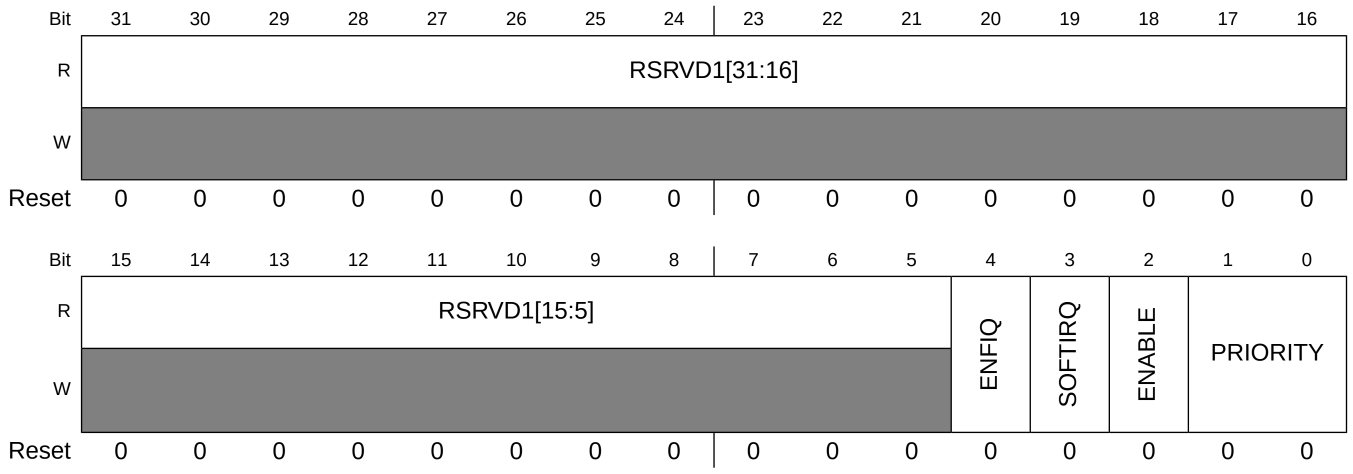
HW_ICOLL_INTERRUPT80_TOG: 0x62C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT80_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT80 – 8000_0000h base + 620h offset = 8000_0620h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT80 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT80 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.91 Interrupt Collector Interrupt Register 81 (HW_ICOLL_INTERRUPT81)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT81: 0x630

HW_ICOLL_INTERRUPT81_SET: 0x634

HW_ICOLL_INTERRUPT81_CLR: 0x638

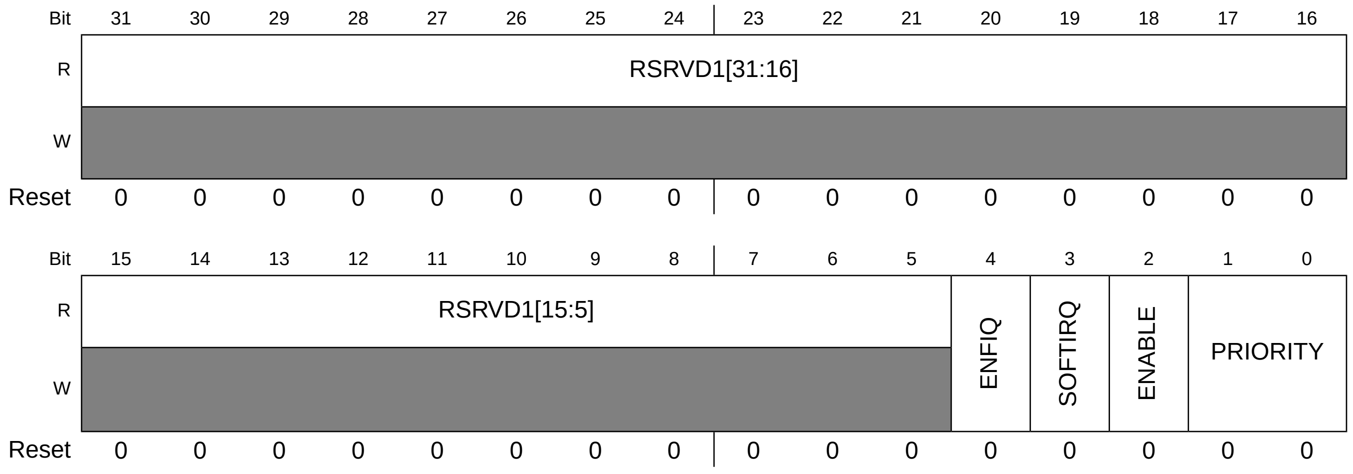
HW_ICOLL_INTERRUPT81_TOG: 0x63C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT81_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT81 – 8000_0000h base + 630h offset = 8000_0630h



HW_ICOLL_INTERRUPT81 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.92 Interrupt Collector Interrupt Register 82 (HW_ICOLL_INTERRUPT82)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT82: 0x640

HW_ICOLL_INTERRUPT82_SET: 0x644

HW_ICOLL_INTERRUPT82_CLR: 0x648

HW_ICOLL_INTERRUPT82_TOG: 0x64C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT82_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT82 – 8000_0000h base + 640h offset = 8000_0640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT82 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT82 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.93 Interrupt Collector Interrupt Register 83 (HW_ICOLL_INTERRUPT83)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT83: 0x650

HW_ICOLL_INTERRUPT83_SET: 0x654

HW_ICOLL_INTERRUPT83_CLR: 0x658

HW_ICOLL_INTERRUPT83_TOG: 0x65C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT83_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT83 – 8000_0000h base + 650h offset = 8000_0650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT83 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.94 Interrupt Collector Interrupt Register 84 (HW_ICOLL_INTERRUPT84)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT84: 0x660

HW_ICOLL_INTERRUPT84_SET: 0x664

HW_ICOLL_INTERRUPT84_CLR: 0x668

HW_ICOLL_INTERRUPT84_TOG: 0x66C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT84_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT84 – 8000_0000h base + 660h offset = 8000_0660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT84 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.95 Interrupt Collector Interrupt Register 85 (HW_ICOLL_INTERRUPT85)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT85: 0x670

HW_ICOLL_INTERRUPT85_SET: 0x674

HW_ICOLL_INTERRUPT85_CLR: 0x678

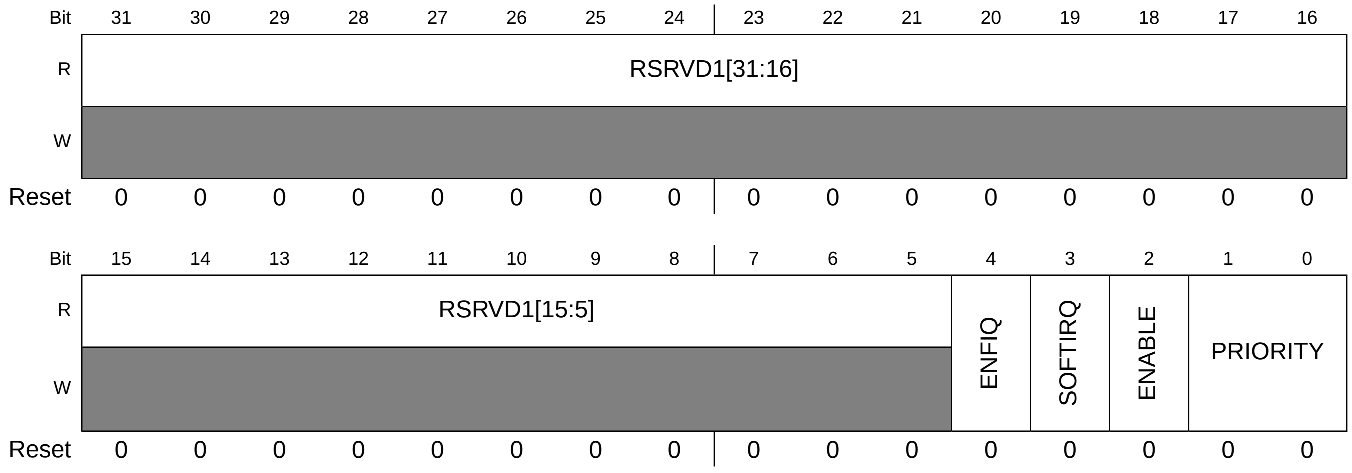
HW_ICOLL_INTERRUPT85_TOG: 0x67C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT85_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT85 – 8000_0000h base + 670h offset = 8000_0670h



HW_ICOLL_INTERRUPT85 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT85 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.96 Interrupt Collector Interrupt Register 86 (HW_ICOLL_INTERRUPT86)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT86: 0x680

HW_ICOLL_INTERRUPT86_SET: 0x684

HW_ICOLL_INTERRUPT86_CLR: 0x688

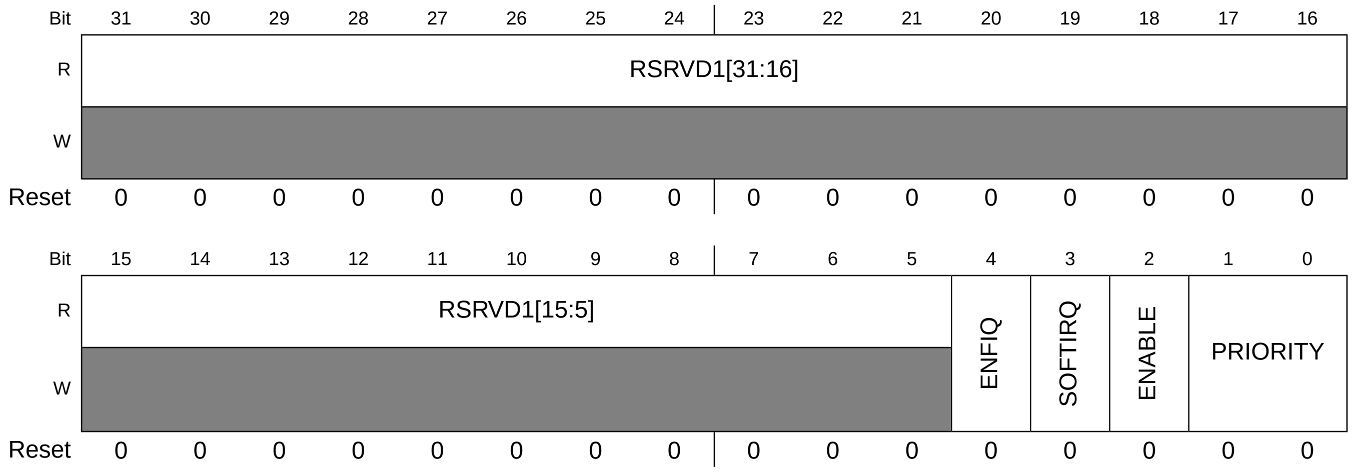
HW_ICOLL_INTERRUPT86_TOG: 0x68C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT86_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT86 – 8000_0000h base + 680h offset = 8000_0680h



HW_ICOLL_INTERRUPT86 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.97 Interrupt Collector Interrupt Register 87 (HW_ICOLL_INTERRUPT87)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT87: 0x690

HW_ICOLL_INTERRUPT87_SET: 0x694

HW_ICOLL_INTERRUPT87_CLR: 0x698

HW_ICOLL_INTERRUPT87_TOG: 0x69C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT87_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT87 – 8000_0000h base + 690h offset = 8000_0690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT87 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT87 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.98 Interrupt Collector Interrupt Register 88 (HW_ICOLL_INTERRUPT88)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT88: 0x6A0

HW_ICOLL_INTERRUPT88_SET: 0x6A4

HW_ICOLL_INTERRUPT88_CLR: 0x6A8

HW_ICOLL_INTERRUPT88_TOG: 0x6AC

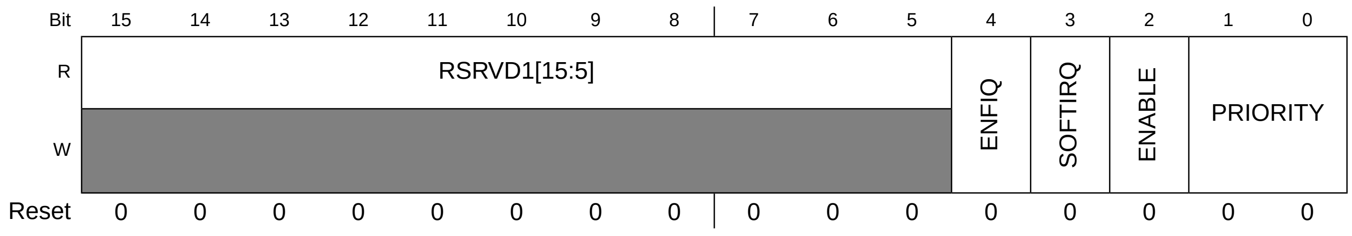
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT88_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT88 – 8000_0000h base + 6A0h offset = 8000_06A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT88 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.99 Interrupt Collector Interrupt Register 89 (HW_ICOLL_INTERRUPT89)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT89: 0x6B0

HW_ICOLL_INTERRUPT89_SET: 0x6B4

HW_ICOLL_INTERRUPT89_CLR: 0x6B8

HW_ICOLL_INTERRUPT89_TOG: 0x6BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT89_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT89 – 8000_0000h base + 6B0h offset = 8000_06B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT89 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.100 Interrupt Collector Interrupt Register 90 (HW_ICOLL_INTERRUPT90)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT90: 0x6C0

HW_ICOLL_INTERRUPT90_SET: 0x6C4

HW_ICOLL_INTERRUPT90_CLR: 0x6C8

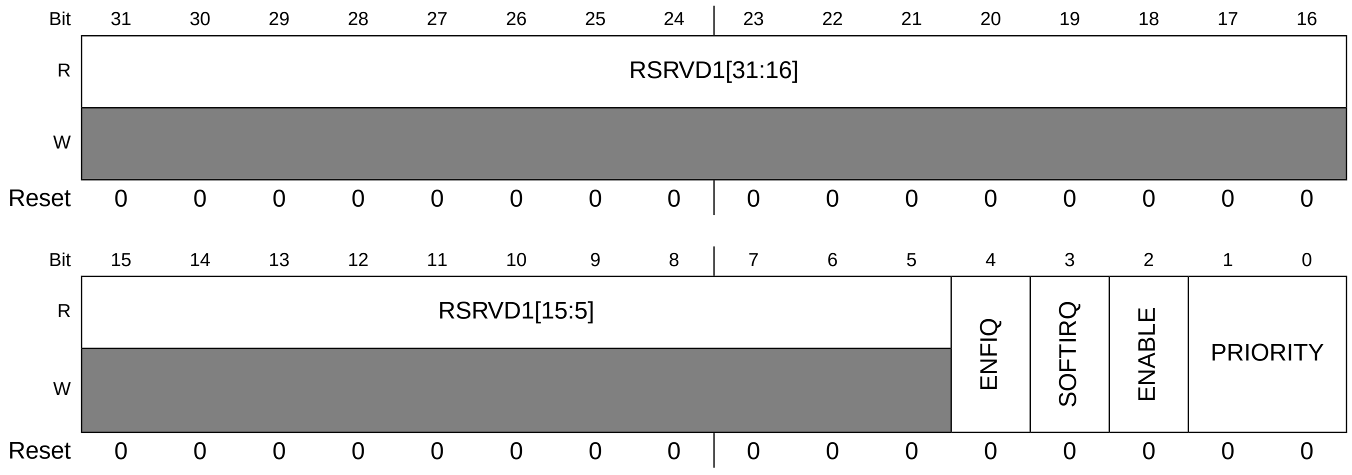
HW_ICOLL_INTERRUPT90_TOG: 0x6CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT90_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT90 – 8000_0000h base + 6C0h offset = 8000_06C0h



HW_ICOLL_INTERRUPT90 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT90 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.101 Interrupt Collector Interrupt Register 91 (HW_ICOLL_INTERRUPT91)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT91: 0x6D0

HW_ICOLL_INTERRUPT91_SET: 0x6D4

HW_ICOLL_INTERRUPT91_CLR: 0x6D8

HW_ICOLL_INTERRUPT91_TOG: 0x6DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT91_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT91 – 8000_0000h base + 6D0h offset = 8000_06D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT91 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.102 Interrupt Collector Interrupt Register 92 (HW_ICOLL_INTERRUPT92)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT92: 0x6E0

HW_ICOLL_INTERRUPT92_SET: 0x6E4

HW_ICOLL_INTERRUPT92_CLR: 0x6E8

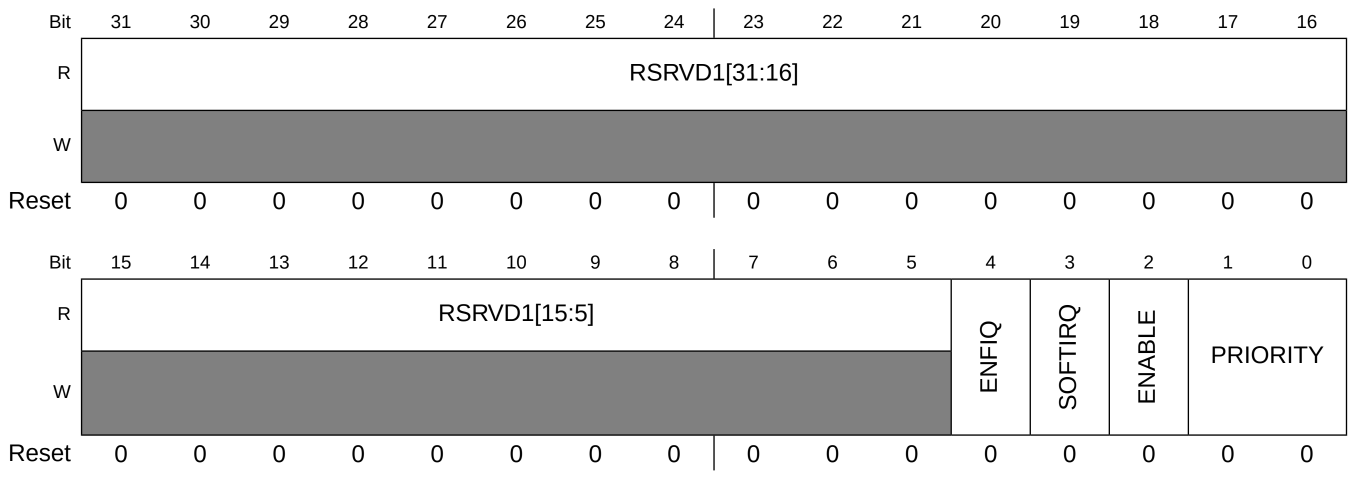
HW_ICOLL_INTERRUPT92_TOG: 0x6EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT92_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT92 – 8000_0000h base + 6E0h offset = 8000_06E0h



HW_ICOLL_INTERRUPT92 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT92 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.103 Interrupt Collector Interrupt Register 93 (HW_ICOLL_INTERRUPT93)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT93: 0x6F0

HW_ICOLL_INTERRUPT93_SET: 0x6F4

HW_ICOLL_INTERRUPT93_CLR: 0x6F8

HW_ICOLL_INTERRUPT93_TOG: 0x6FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT93_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT93 – 8000_0000h base + 6F0h offset = 8000_06F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT93 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.104 Interrupt Collector Interrupt Register 94 (HW_ICOLL_INTERRUPT94)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT94: 0x700

HW_ICOLL_INTERRUPT94_SET: 0x704

HW_ICOLL_INTERRUPT94_CLR: 0x708

HW_ICOLL_INTERRUPT94_TOG: 0x70C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT94_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT94 – 8000_0000h base + 700h offset = 8000_0700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT94 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.105 Interrupt Collector Interrupt Register 95 (HW_ICOLL_INTERRUPT95)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT95: 0x710

HW_ICOLL_INTERRUPT95_SET: 0x714

HW_ICOLL_INTERRUPT95_CLR: 0x718

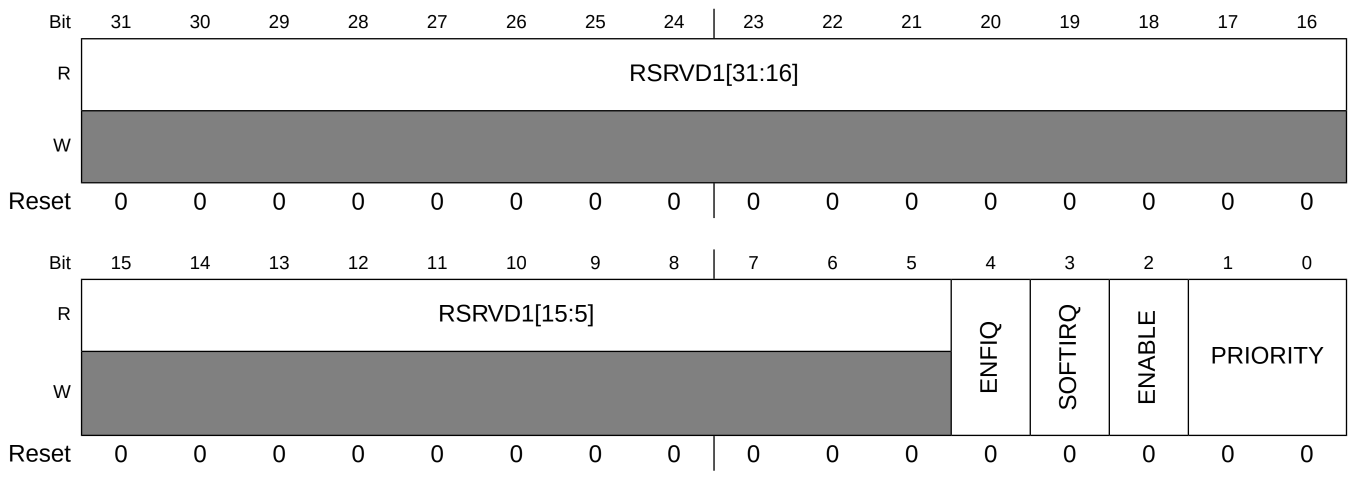
HW_ICOLL_INTERRUPT95_TOG: 0x71C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT95_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT95 – 8000_0000h base + 710h offset = 8000_0710h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT95 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT95 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.106 Interrupt Collector Interrupt Register 96 (HW_ICOLL_INTERRUPT96)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT96: 0x720

HW_ICOLL_INTERRUPT96_SET: 0x724

HW_ICOLL_INTERRUPT96_CLR: 0x728

HW_ICOLL_INTERRUPT96_TOG: 0x72C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT96_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT96 – 8000_0000h base + 720h offset = 8000_
0720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT96 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.107 Interrupt Collector Interrupt Register 97 (HW_ICOLL_INTERRUPT97)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT97: 0x730

HW_ICOLL_INTERRUPT97_SET: 0x734

HW_ICOLL_INTERRUPT97_CLR: 0x738

HW_ICOLL_INTERRUPT97_TOG: 0x73C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT97_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT97 – 8000_0000h base + 730h offset = 8000_0730h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT97 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT97 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.108 Interrupt Collector Interrupt Register 98 (HW_ICOLL_INTERRUPT98)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT98: 0x740

HW_ICOLL_INTERRUPT98_SET: 0x744

HW_ICOLL_INTERRUPT98_CLR: 0x748

HW_ICOLL_INTERRUPT98_TOG: 0x74C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT98_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT98 – 8000_0000h base + 740h offset = 8000_0740h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT98 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.109 Interrupt Collector Interrupt Register 99 (HW_ICOLL_INTERRUPT99)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT99: 0x750

HW_ICOLL_INTERRUPT99_SET: 0x754

HW_ICOLL_INTERRUPT99_CLR: 0x758

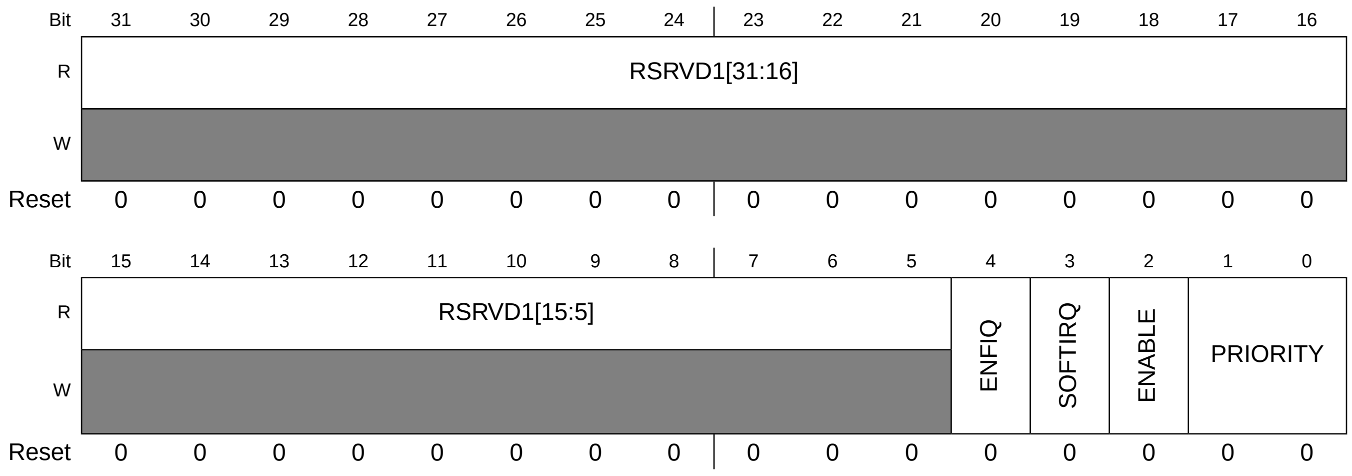
HW_ICOLL_INTERRUPT99_TOG: 0x75C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT99_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT99 – 8000_0000h base + 750h offset = 8000_0750h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD1[31:16]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT99 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.110 Interrupt Collector Interrupt Register 100 (HW_ICOLL_INTERRUPT100)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT100: 0x760

HW_ICOLL_INTERRUPT100_SET: 0x764

HW_ICOLL_INTERRUPT100_CLR: 0x768

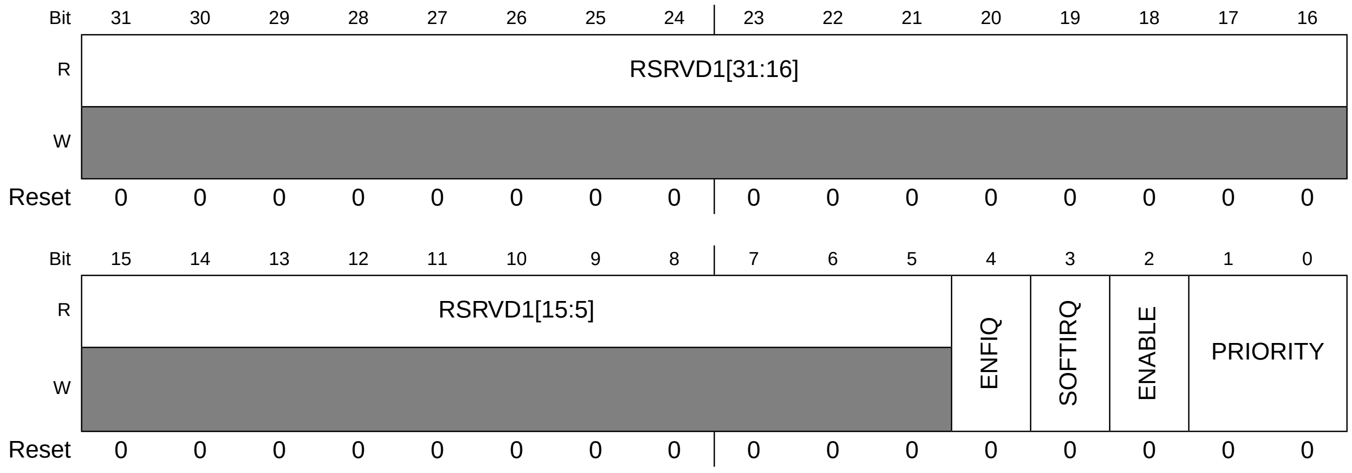
HW_ICOLL_INTERRUPT100_TOG: 0x76C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT100_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT100 – 8000_0000h base + 760h offset = 8000_0760h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT100 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT100 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.111 Interrupt Collector Interrupt Register 101 (HW_ICOLL_INTERRUPT101)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT101: 0x770

HW_ICOLL_INTERRUPT101_SET: 0x774

HW_ICOLL_INTERRUPT101_CLR: 0x778

HW_ICOLL_INTERRUPT101_TOG: 0x77C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT101_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT101 – 8000_0000h base + 770h offset = 8000_0770h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT101 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.112 Interrupt Collector Interrupt Register 102 (HW_ICOLL_INTERRUPT102)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT102: 0x780

HW_ICOLL_INTERRUPT102_SET: 0x784

HW_ICOLL_INTERRUPT102_CLR: 0x788

HW_ICOLL_INTERRUPT102_TOG: 0x78C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT102_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT102 – 8000_0000h base + 780h offset = 8000_0780h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT102 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT102 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.113 Interrupt Collector Interrupt Register 103 (HW_ICOLL_INTERRUPT103)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT103: 0x790

HW_ICOLL_INTERRUPT103_SET: 0x794

HW_ICOLL_INTERRUPT103_CLR: 0x798

HW_ICOLL_INTERRUPT103_TOG: 0x79C

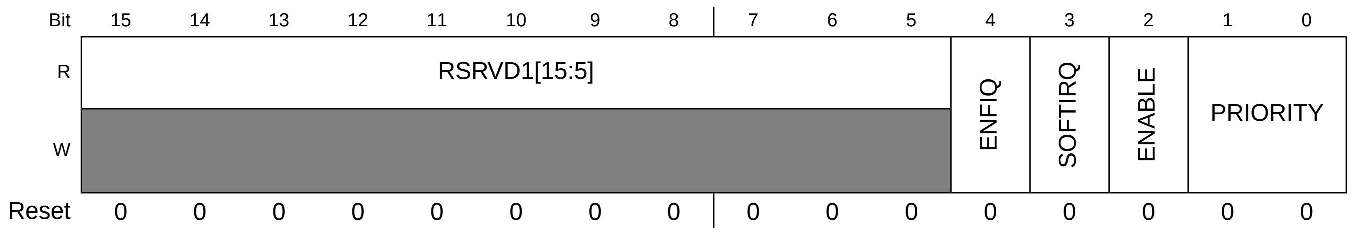
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT103_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT103 – 8000_0000h base + 790h offset = 8000_0790h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0


Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT103 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.114 Interrupt Collector Interrupt Register 104 (HW_ICOLL_INTERRUPT104)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT104: 0x7A0

HW_ICOLL_INTERRUPT104_SET: 0x7A4

HW_ICOLL_INTERRUPT104_CLR: 0x7A8

HW_ICOLL_INTERRUPT104_TOG: 0x7AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT104_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT104 – 8000_0000h base + 7A0h offset = 8000_07A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT104 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.115 Interrupt Collector Interrupt Register 105 (HW_ICOLL_INTERRUPT105)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT105: 0x7B0

HW_ICOLL_INTERRUPT105_SET: 0x7B4

HW_ICOLL_INTERRUPT105_CLR: 0x7B8

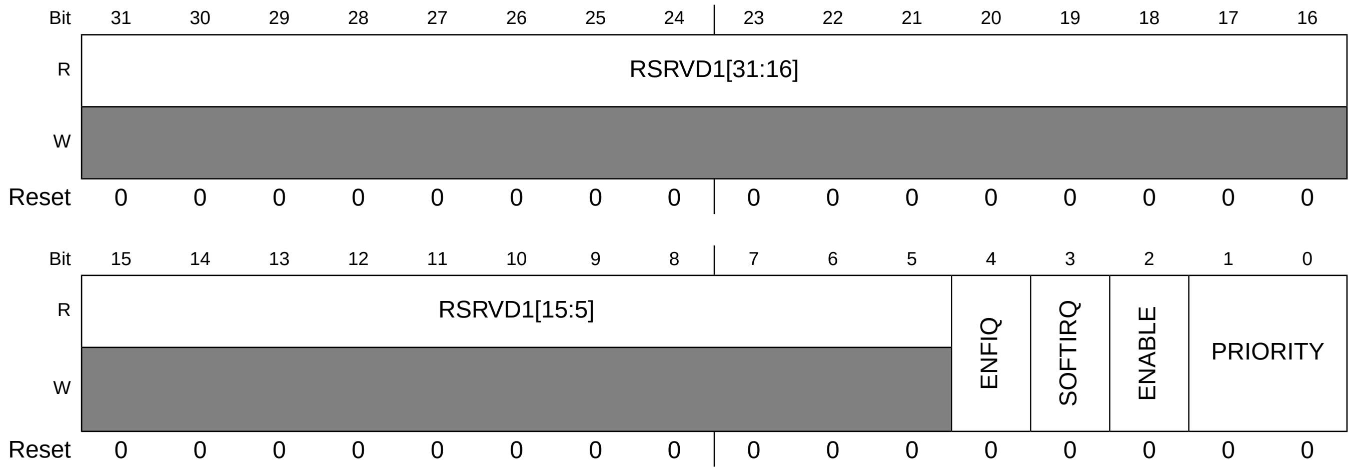
HW_ICOLL_INTERRUPT105_TOG: 0x7BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT105_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT105 – 8000_0000h base + 7B0h offset = 8000_07B0h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT105 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT105 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.116 Interrupt Collector Interrupt Register 106 (HW_ICOLL_INTERRUPT106)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT106: 0x7C0

HW_ICOLL_INTERRUPT106_SET: 0x7C4

HW_ICOLL_INTERRUPT106_CLR: 0x7C8

HW_ICOLL_INTERRUPT106_TOG: 0x7CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT106_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT106 – 8000_0000h base + 7C0h offset = 8000_07C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT106 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.117 Interrupt Collector Interrupt Register 107 (HW_ICOLL_INTERRUPT107)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT107: 0x7D0

HW_ICOLL_INTERRUPT107_SET: 0x7D4

HW_ICOLL_INTERRUPT107_CLR: 0x7D8

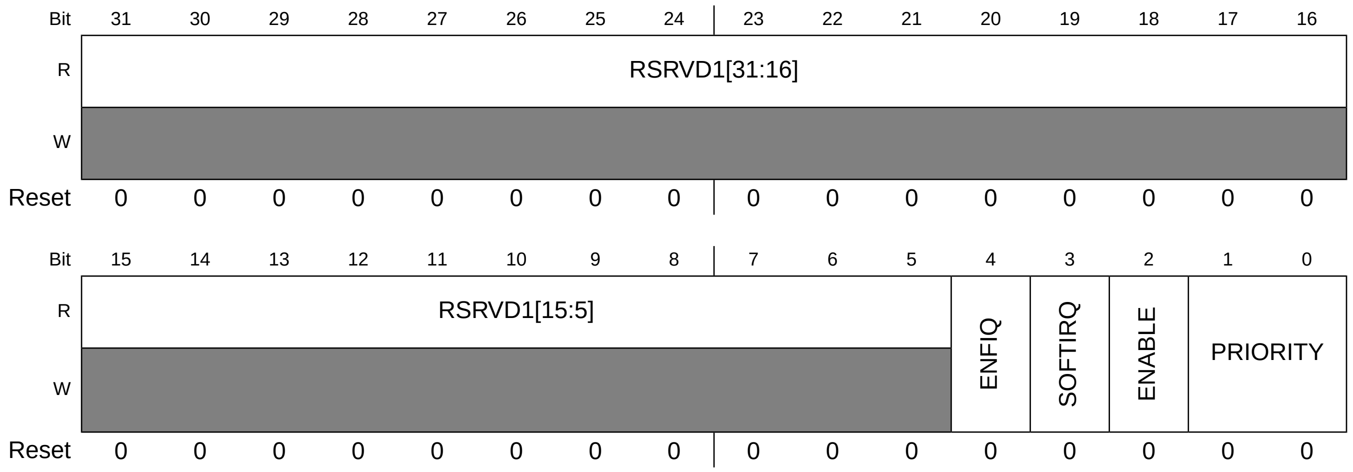
HW_ICOLL_INTERRUPT107_TOG: 0x7DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT107_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT107 – 8000_0000h base + 7D0h offset = 8000_07D0h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT107 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT107 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.118 Interrupt Collector Interrupt Register 108 (HW_ICOLL_INTERRUPT108)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT108: 0x7E0

HW_ICOLL_INTERRUPT108_SET: 0x7E4

HW_ICOLL_INTERRUPT108_CLR: 0x7E8

HW_ICOLL_INTERRUPT108_TOG: 0x7EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT108_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT108 – 8000_0000h base + 7E0h offset = 8000_07E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT108 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.119 Interrupt Collector Interrupt Register 109 (HW_ICOLL_INTERRUPT109)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT109: 0x7F0

HW_ICOLL_INTERRUPT109_SET: 0x7F4

HW_ICOLL_INTERRUPT109_CLR: 0x7F8

HW_ICOLL_INTERRUPT109_TOG: 0x7FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT109_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT109 – 8000_0000h base + 7F0h offset = 8000_07F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT109 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.120 Interrupt Collector Interrupt Register 110 (HW_ICOLL_INTERRUPT110)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT110: 0x800

HW_ICOLL_INTERRUPT110_SET: 0x804

HW_ICOLL_INTERRUPT110_CLR: 0x808

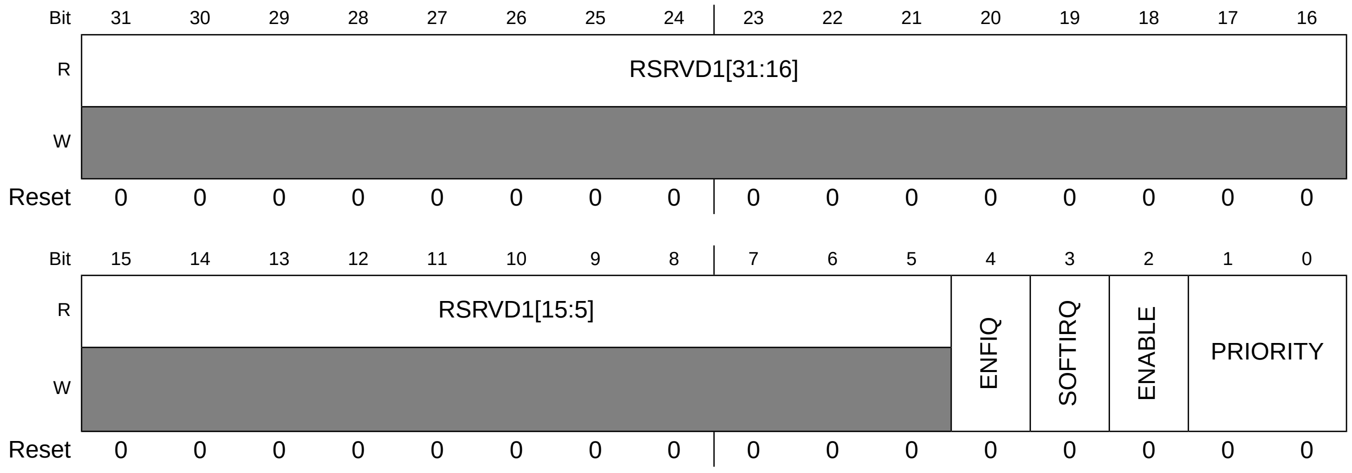
HW_ICOLL_INTERRUPT110_TOG: 0x80C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT110_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT110 – 8000_0000h base + 800h offset = 8000_0800h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT110 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT110 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.121 Interrupt Collector Interrupt Register 111 (HW_ICOLL_INTERRUPT111)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT111: 0x810

HW_ICOLL_INTERRUPT111_SET: 0x814

HW_ICOLL_INTERRUPT111_CLR: 0x818

HW_ICOLL_INTERRUPT111_TOG: 0x81C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT111_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT111 – 8000_0000h base + 810h offset = 8000_0810h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT111 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.122 Interrupt Collector Interrupt Register 112 (HW_ICOLL_INTERRUPT112)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT112: 0x820

HW_ICOLL_INTERRUPT112_SET: 0x824

HW_ICOLL_INTERRUPT112_CLR: 0x828

HW_ICOLL_INTERRUPT112_TOG: 0x82C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT112_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT112 – 8000_0000h base + 820h offset = 8000_0820h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT112 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT112 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.123 Interrupt Collector Interrupt Register 113 (HW_ICOLL_INTERRUPT113)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT113: 0x830

HW_ICOLL_INTERRUPT113_SET: 0x834

HW_ICOLL_INTERRUPT113_CLR: 0x838

HW_ICOLL_INTERRUPT113_TOG: 0x83C

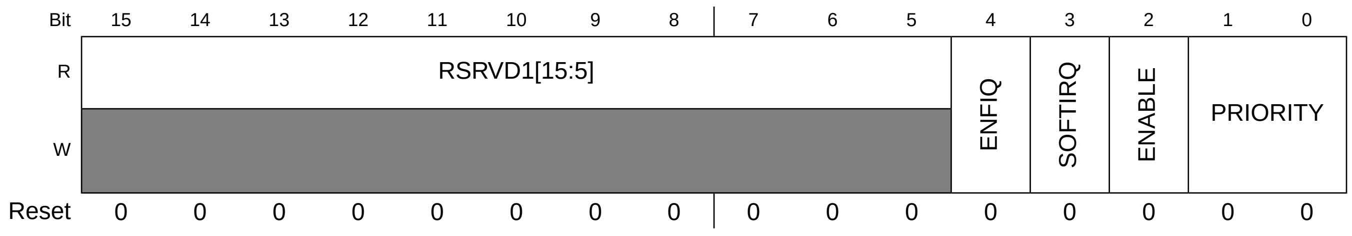
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT113_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT113 – 8000_0000h base + 830h offset = 8000_0830h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT113 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.124 Interrupt Collector Interrupt Register 114 (HW_ICOLL_INTERRUPT114)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT114: 0x840

HW_ICOLL_INTERRUPT114_SET: 0x844

HW_ICOLL_INTERRUPT114_CLR: 0x848

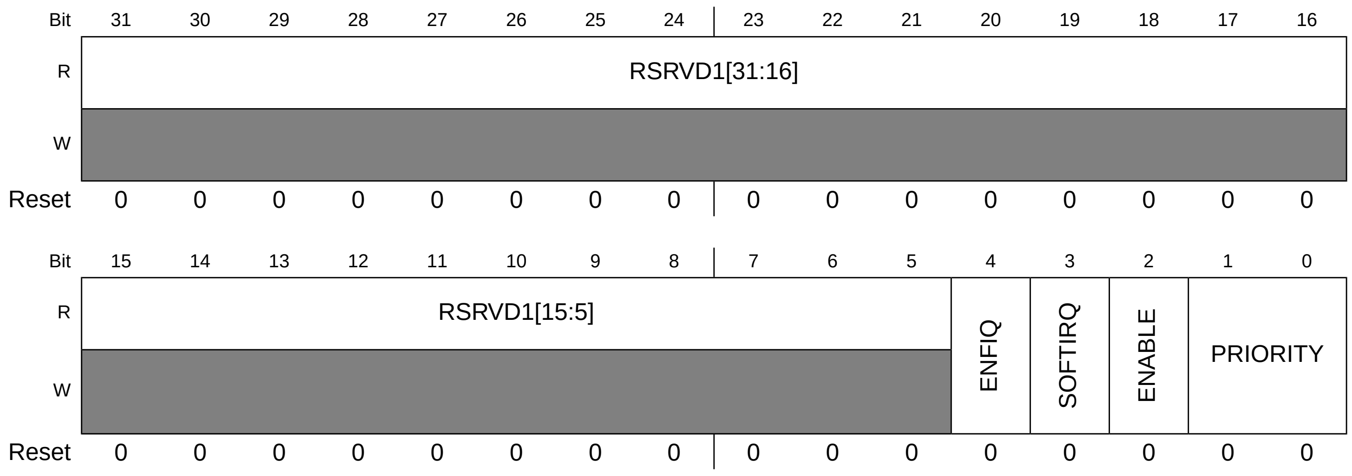
HW_ICOLL_INTERRUPT114_TOG: 0x84C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT114_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT114 – 8000_0000h base + 840h offset = 8000_0840h



HW_ICOLL_INTERRUPT114 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.125 Interrupt Collector Interrupt Register 115 (HW_ICOLL_INTERRUPT115)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT115: 0x850

HW_ICOLL_INTERRUPT115_SET: 0x854

HW_ICOLL_INTERRUPT115_CLR: 0x858

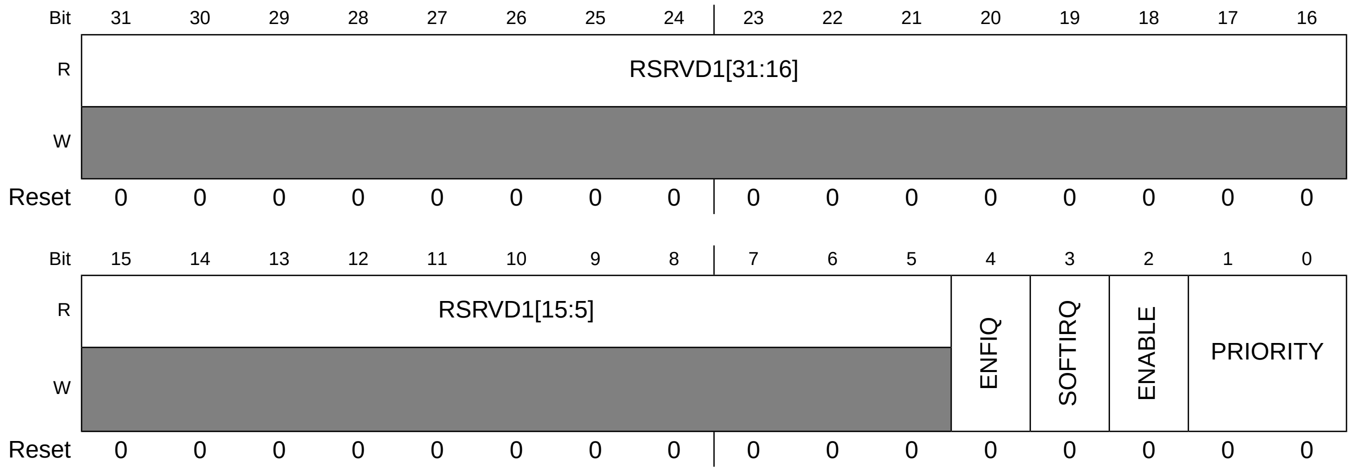
HW_ICOLL_INTERRUPT115_TOG: 0x85C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT115_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT115 – 8000_0000h base + 850h offset = 8000_0850h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT115 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT115 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.126 Interrupt Collector Interrupt Register 116 (HW_ICOLL_INTERRUPT116)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT116: 0x860

HW_ICOLL_INTERRUPT116_SET: 0x864

HW_ICOLL_INTERRUPT116_CLR: 0x868

HW_ICOLL_INTERRUPT116_TOG: 0x86C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT116_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT116 – 8000_0000h base + 860h offset = 8000_0860h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT116 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.127 Interrupt Collector Interrupt Register 117 (HW_ICOLL_INTERRUPT117)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT117: 0x870

HW_ICOLL_INTERRUPT117_SET: 0x874

HW_ICOLL_INTERRUPT117_CLR: 0x878

HW_ICOLL_INTERRUPT117_TOG: 0x87C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT117_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT117 – 8000_0000h base + 870h offset = 8000_0870h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT117 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT117 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.128 Interrupt Collector Interrupt Register 118 (HW_ICOLL_INTERRUPT118)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT118: 0x880

HW_ICOLL_INTERRUPT118_SET: 0x884

HW_ICOLL_INTERRUPT118_CLR: 0x888

HW_ICOLL_INTERRUPT118_TOG: 0x88C

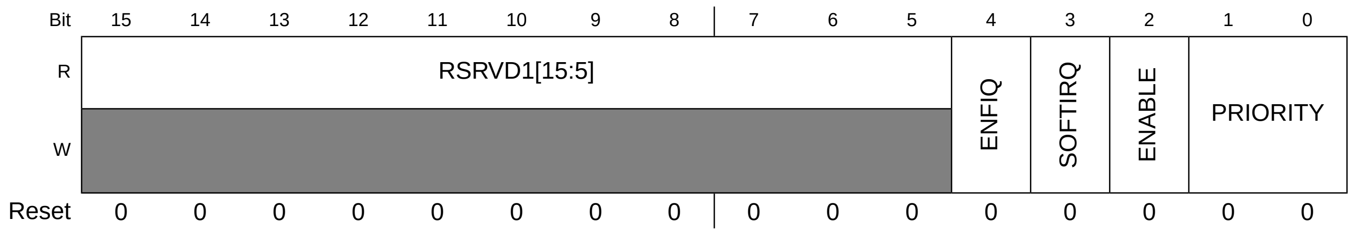
This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT118_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT118 – 8000_0000h base + 880h offset = 8000_0880h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_ICOLL_INTERRUPT118 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.129 Interrupt Collector Interrupt Register 119 (HW_ICOLL_INTERRUPT119)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT119: 0x890

HW_ICOLL_INTERRUPT119_SET: 0x894

HW_ICOLL_INTERRUPT119_CLR: 0x898

HW_ICOLL_INTERRUPT119_TOG: 0x89C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT119_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT119 – 8000_0000h base + 890h offset = 8000_0890h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT119 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.130 Interrupt Collector Interrupt Register 120 (HW_ICOLL_INTERRUPT120)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT120: 0x8A0

HW_ICOLL_INTERRUPT120_SET: 0x8A4

HW_ICOLL_INTERRUPT120_CLR: 0x8A8

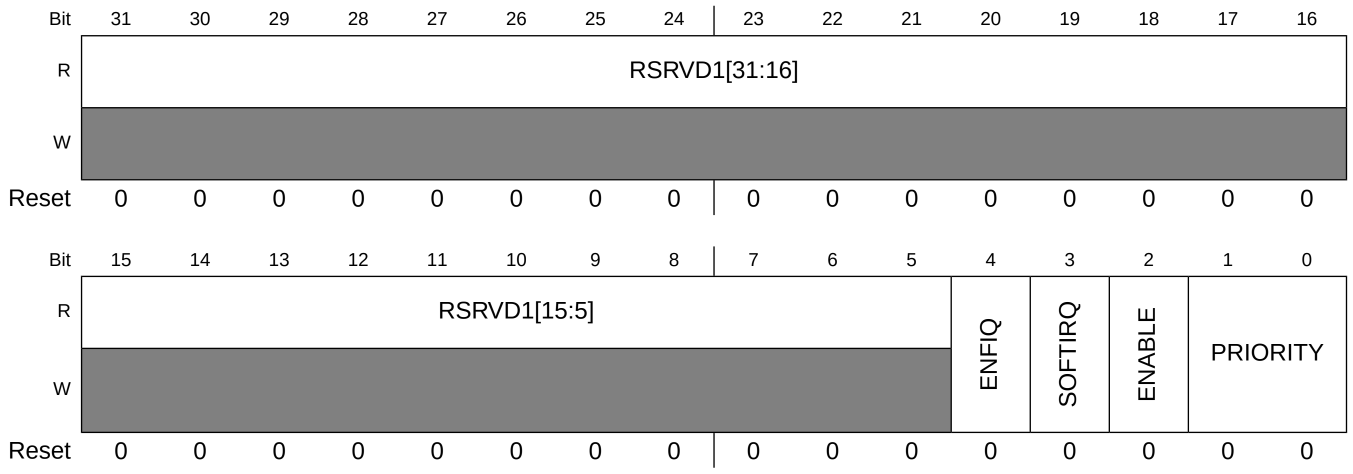
HW_ICOLL_INTERRUPT120_TOG: 0x8AC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT120_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT120 – 8000_0000h base + 8A0h offset = 8000_08A0h



HW_ICOLL_INTERRUPT120 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT120 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.131 Interrupt Collector Interrupt Register 121 (HW_ICOLL_INTERRUPT121)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT121: 0x8B0

HW_ICOLL_INTERRUPT121_SET: 0x8B4

HW_ICOLL_INTERRUPT121_CLR: 0x8B8

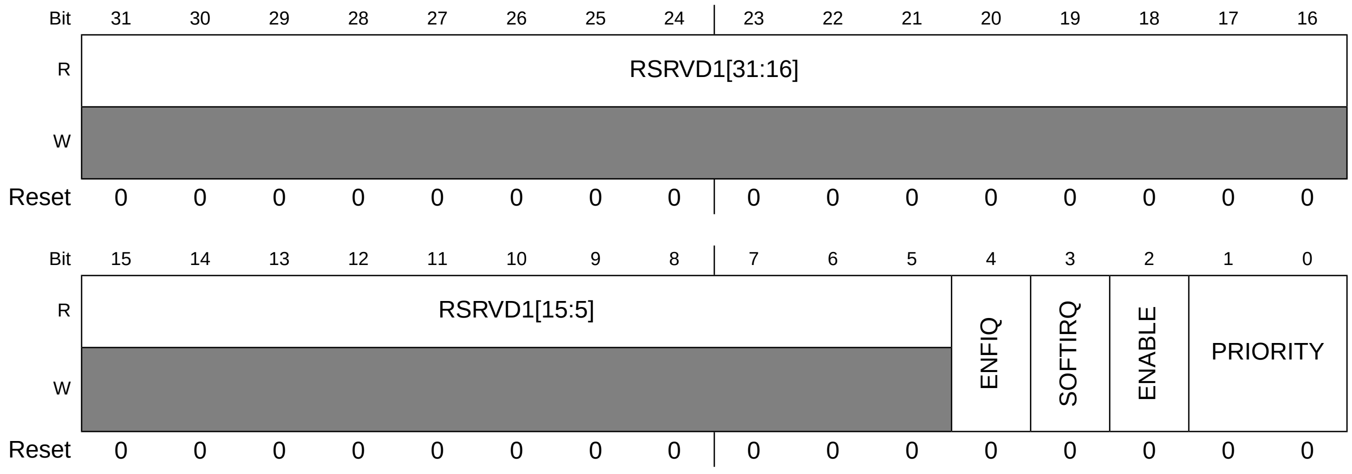
HW_ICOLL_INTERRUPT121_TOG: 0x8BC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT121_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT121 – 8000_0000h base + 8B0h offset = 8000_08B0h



HW_ICOLL_INTERRUPT121 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.132 Interrupt Collector Interrupt Register 122 (HW_ICOLL_INTERRUPT122)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT122: 0x8C0

HW_ICOLL_INTERRUPT122_SET: 0x8C4

HW_ICOLL_INTERRUPT122_CLR: 0x8C8

HW_ICOLL_INTERRUPT122_TOG: 0x8CC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT122_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT122 – 8000_0000h base + 8C0h offset = 8000_08C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W												ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT122 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT122 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.133 Interrupt Collector Interrupt Register 123 (HW_ICOLL_INTERRUPT123)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT123: 0x8D0

HW_ICOLL_INTERRUPT123_SET: 0x8D4

HW_ICOLL_INTERRUPT123_CLR: 0x8D8

HW_ICOLL_INTERRUPT123_TOG: 0x8DC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT123_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT123 – 8000_0000h base + 8D0h offset = 8000_08D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT123 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.134 Interrupt Collector Interrupt Register 124 (HW_ICOLL_INTERRUPT124)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT124: 0x8E0

HW_ICOLL_INTERRUPT124_SET: 0x8E4

HW_ICOLL_INTERRUPT124_CLR: 0x8E8

HW_ICOLL_INTERRUPT124_TOG: 0x8EC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT124_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT124 – 8000_0000h base + 8E0h offset = 8000_08E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT124 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.135 Interrupt Collector Interrupt Register 125 (HW_ICOLL_INTERRUPT125)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT125: 0x8F0

HW_ICOLL_INTERRUPT125_SET: 0x8F4

HW_ICOLL_INTERRUPT125_CLR: 0x8F8

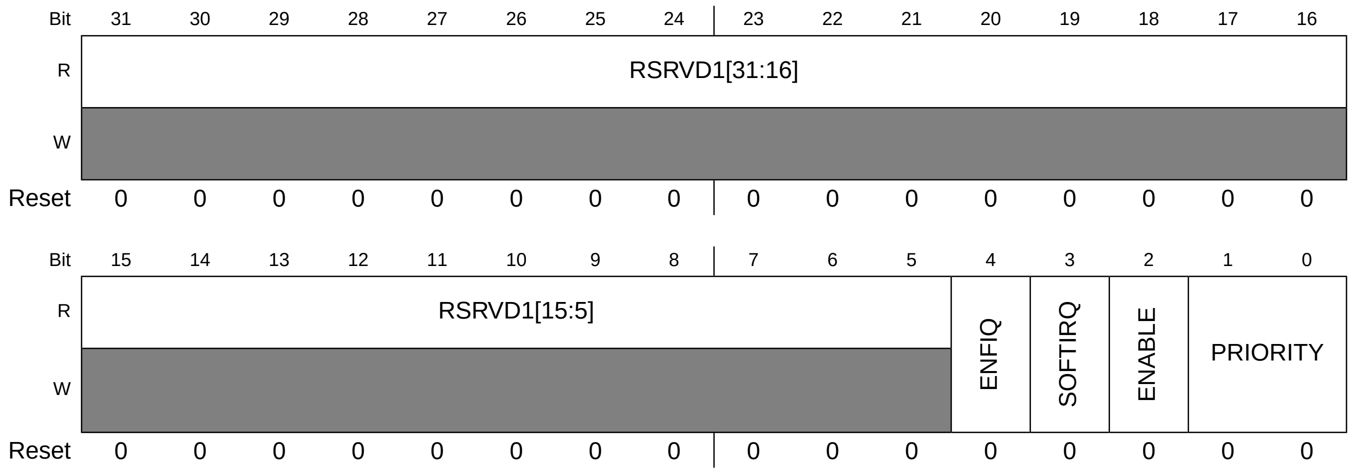
HW_ICOLL_INTERRUPT125_TOG: 0x8FC

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT125_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT125 – 8000_0000h base + 8F0h offset = 8000_08F0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:5]
W
ENFIQ
SOFTIRQ
ENABLE
PRIORITY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ICOLL_INTERRUPT125 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic.

HW_ICOLL_INTERRUPT125 field descriptions (continued)

Field	Description
	0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.136 Interrupt Collector Interrupt Register 126 (HW_ICOLL_INTERRUPT126)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT126: 0x900

HW_ICOLL_INTERRUPT126_SET: 0x904

HW_ICOLL_INTERRUPT126_CLR: 0x908

HW_ICOLL_INTERRUPT126_TOG: 0x90C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT126_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT126 – 8000_0000h base + 900h offset = 8000_0900h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT126 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.137 Interrupt Collector Interrupt Register 127 (HW_ICOLL_INTERRUPT127)

This register provides a mechanism to specify the priority level for an interrupt source. It also provides an enable and software interrupt for each one, as well as security designation.

HW_ICOLL_INTERRUPT127: 0x910

HW_ICOLL_INTERRUPT127_SET: 0x914

HW_ICOLL_INTERRUPT127_CLR: 0x918

HW_ICOLL_INTERRUPT127_TOG: 0x91C

This register provides a mechanism to specify the priority associated with an interrupt bit. In addition, this register controls the enable and software generated interrupt. WARNING: Modifying the priority of an enabled interrupt may result in undefined behavior. You should always disable an interrupt prior to changing its priority.

EXAMPLE

```
HW_ICOLL_INTERRUPT127_SET(0,0x00000001);
```

Address: HW_ICOLL_INTERRUPT127 – 8000_0000h base + 910h offset = 8000_0910h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											ENFIQ	SOFTIRQ	ENABLE	PRIORITY	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_INTERRUPT127 field descriptions

Field	Description
31–5 RSRVD1	Always write zeroes to this bitfield.
4 ENFIQ	Set this to 1 to steer this interrupt to the non-vectored FIQ line. When set to 0 the interrupt will pass through the main IRQ FSM and priority logic. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable
3 SOFTIRQ	Set this bit to one to force a software interrupt. 0x0 **NO_INTERRUPT** — turn off the software interrupt request. 0x1 **FORCE_INTERRUPT** — force a software interrupt
2 ENABLE	Enable the interrupt bit through the collector. 0x0 **DISABLE** — Disable 0x1 **ENABLE** — Enable

HW_ICOLL_INTERRUPT127 field descriptions (continued)

Field	Description
1–0 PRIORITY	Set the priority level for this interrupt, 0x3 is highest, 0x0 is lowest (weakest). 0x0 **LEVEL0** — level 0, lowest or weakest priority 0x1 **LEVEL1** — level 1 0x2 **LEVEL2** — level 2 0x3 **LEVEL3** — level 3, highest or strongest priority

5.4.138 Interrupt Collector Debug Register 0 (HW_ICOLL_DEBUG)

The contents of this register will be defined as the hardware is developed.

HW_ICOLL_DEBUG: 0x1120

HW_ICOLL_DEBUG_SET: 0x1124

HW_ICOLL_DEBUG_CLR: 0x1128

HW_ICOLL_DEBUG_TOG: 0x112C

This register provides diagnostic visibility into the IRQ request state machine and its various inputs.

EXAMPLE

```
if (BF_RD(ICOLL_DEBUG, LEVEL_REQUESTS) != HW_ICOLL_DEBUG_LEVEL_REQUESTS__LEVEL3)
   Error();
TPRINTF(TP_MED, ("ICOLL INSERVICE = 0x%x\n", BF_RD(ICOLL_DEBUG, INSERVICE)));
TPRINTF(TP_MED, ("ICOLL STATE = 0x%x\n", BF_RD(ICOLL_DEBUG, VECTOR_FSM)));
```

Address: HW_ICOLL_DEBUG – 8000_0000h base + 1120h offset = 8000_1120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INSERVICE				LEVEL_REQUESTS				REQUESTS_BY_LEVEL				RSRVD2		FIQ	IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						VECTOR_FSM									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DEBUG field descriptions

Field	Description
31–28 INSERVICE	read-only view of the Inservice bits used for nesting IRQs. 0x1 **LEVEL0** — LEVEL0 0x2 **LEVEL1** — LEVEL1 0x4 **LEVEL2** — LEVEL2 0x8 **LEVEL3** — LEVEL3
27–24 LEVEL_ REQUESTS	read-only view of the requsts by priority level for the current IRQ. 0x1 **LEVEL0** — LEVEL0 0x2 **LEVEL1** — LEVEL1 0x4 **LEVEL2** — LEVEL2 0x8 **LEVEL3** — LEVEL3
23–20 REQUESTS_BY_ LEVEL	read-only view of the requsts by priority level for the current IRQ. 0x1 **LEVEL0** — LEVEL0 0x2 **LEVEL1** — LEVEL1 0x4 **LEVEL2** — LEVEL2 0x8 **LEVEL3** — LEVEL3
19–18 RSRVD2	Always write zeroes to this bitfield.
17 FIQ	Read-Only View of the FIQ output to the CPU. 0x0 **NO_FIQ_REQUESTED** — No FIQ Requested 0x1 **FIQ_REQUESTED** — FIQ Requested
16 IRQ	Read-Only View of the FIQ output to the CPU. 0x0 **NO_IRQ_REQUESTED** — No IRQ Requested 0x1 **IRQ_REQUESTED** — IRQ Requested
15–10 RSRVD1	Always write zeroes to this bitfield.
9–0 VECTOR_FSM	Empty description. 0x000 **FSM_IDLE** — FSM_IDLE 0x001 **FSM_MULTICYCLE1** — FSM_MULTICYCLE1 0x002 **FSM_MULTICYCLE2** — FSM_MULTICYCLE2 0x004 **FSM_PENDING** — FSM_PENDING 0x008 **FSM_MULTICYCLE3** — FSM_MULTICYCLE3 0x010 **FSM_MULTICYCLE4** — FSM_MULTICYCLE4 0x020 **FSM_ISR_RUNNING1** — FSM_ISR_RUNNING1 0x040 **FSM_ISR_RUNNING2** — FSM_ISR_RUNNING2 0x080 **FSM_ISR_RUNNING3** — FSM_ISR_RUNNING3 0x100 **FSM_MULTICYCLE5** — FSM_MULTICYCLE5 0x200 **FSM_MULTICYCLE6** — FSM_MULTICYCLE6

5.4.139 Interrupt Collector Debug Read Register 0 (HW_ICOLL_DBGREAD0)

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD0: 0x1130

HW_ICOLL_DBGREAD0_SET: 0x1134

HW_ICOLL_DBGREAD0_CLR: 0x1138

HW_ICOLL_DBGREAD0_TOG: 0x113C

This register is used to test the read mux paths on the APBH.

EXAMPLE

```
if (HW_ICOLL_DBGREAD0_RD != 0xECA94567)
   Error();
```

Address: HW_ICOLL_DBGREAD0 – 8000_0000h base + 1130h offset = 8000_1130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W																																
Reset	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	1	0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1

HW_ICOLL_DBGREAD0 field descriptions

Field	Description
31–0 VALUE	Fixed read-only value.

5.4.140 Interrupt Collector Debug Read Register 1 (HW_ICOLL_DBGREAD1)

This register always returns a known read value for debug purposes.

HW_ICOLL_DBGREAD1: 0x1140

HW_ICOLL_DBGREAD1_SET: 0x1144

HW_ICOLL_DBGREAD1_CLR: 0x1148

HW_ICOLL_DBGREAD1_TOG: 0x114C

This register is used to test the read mux paths on the APBH.

EXAMPLE

```
if (HW_ICOLL_DBGREAD1_RD != 0x1356DA98)
   Error();
```

Address: HW_ICOLL_DBGREAD1 – 8000_0000h base + 1140h offset = 8000_1140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W																																
Reset	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1	0	1	1	0	1	1	0	1	0	1	0	0	1	1	0	0	0

HW_ICOLL_DBGREAD1 field descriptions

Field	Description
31–0 VALUE	Fixed read-only value.

5.4.141 Interrupt Collector Debug Flag Register (HW_ICOLL_DBGFLAG)

The Interrupt Collector debug flag register is used to post diagnostic state into simulation.

HW_ICOLL_DBGFLAG: 0x1150

HW_ICOLL_DBGFLAG_SET: 0x1154

HW_ICOLL_DBGFLAG_CLR: 0x1158

HW_ICOLL_DBGFLAG_TOG: 0x115C

This register provides a posting register to synchronize C program execution and the internal simulation environment.

EXAMPLE

```
BF_WR(ICOLL_DBGFLAG, FLAG, 3);
// ...  do some diagnostic action
BF_WR(ICOLL_DBGFLAG, FLAG, 4);
// ...  do some more diagnostic actions
BF_WR(ICOLL_DBGFLAG, FLAG, 5);
```

Address: HW_ICOLL_DBGFLAG – 8000_0000h base + 1150h offset = 8000_1150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1																FLAG															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DBGFLAG field descriptions

Field	Description
31–16 RSRVD1	Always write zeroes to this bitfield.
15–0 FLAG	This debug facility is probably temporary.

5.4.142 Interrupt Collector Debug Read Request Register 0 (HW_ICOLL_DBGREQUEST0)

read-only view into the low 32 bits of the request holding register.

HW_ICOLL_DBGREQUEST0: 0x1160

HW_ICOLL_DBGREQUEST0_SET: 0x1164

HW_ICOLL_DBGREQUEST0_CLR: 0x1168

HW_ICOLL_DBGREQUEST0_TOG: 0x116C

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE

```
if (HW_ICOLL_DBGREQUESTn_RD(0) != 0x00000000)
   Error();
```

Address: HW_ICOLL_DBGREQUEST0 – 8000_0000h base + 1160h offset = 8000_1160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DBGREQUEST0 field descriptions

Field	Description
31–0 BITS	Low 32 bits of the request holding register.

5.4.143 Interrupt Collector Debug Read Request Register 1 (HW_ICOLL_DBGREQUEST1)

read-only view into bits 32-63 of the request holding register.

HW_ICOLL_DBGREQUEST1: 0x1170

HW_ICOLL_DBGREQUEST1_SET: 0x1174

HW_ICOLL_DBGREQUEST1_CLR: 0x1178

HW_ICOLL_DBGREQUEST1_TOG: 0x117C

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
   Error();
```

Address: HW_ICOLL_DBGREQUEST1 – 8000_0000h base + 1170h offset = 8000_1170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DBGREQUEST1 field descriptions

Field	Description
31–0 BITS	Bits 32-63 of the request holding register.

5.4.144 Interrupt Collector Debug Read Request Register 2 (HW_ICOLL_DBGREQUEST2)

read-only view into bits 64-95 of the request holding register.

HW_ICOLL_DBGREQUEST2: 0x1180

HW_ICOLL_DBGREQUEST2_SET: 0x1184

HW_ICOLL_DBGREQUEST2_CLR: 0x1188

HW_ICOLL_DBGREQUEST2_TOG: 0x118C

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
   Error();
```

Address: HW_ICOLL_DBGREQUEST2 – 8000_0000h base + 1180h offset = 8000_1180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DBGREQUEST2 field descriptions

Field	Description
31–0 BITS	Bits 64-95 of the request holding register.

5.4.145 Interrupt Collector Debug Read Request Register 3 (HW_ICOLL_DBGREQUEST3)

read-only view into bits 96-127 of the request holding register.

HW_ICOLL_DBGREQUEST3: 0x1190

HW_ICOLL_DBGREQUEST3_SET: 0x1194

HW_ICOLL_DBGREQUEST3_CLR: 0x1198

HW_ICOLL_DBGREQUEST3_TOG: 0x119C

This register is used to test interrupt collector state machine and its associated request holding register.

EXAMPLE

```
if (HW_ICOLL_DBGREQUESTn_RD(n) != 0x00000000)
   Error();
```

Address: HW_ICOLL_DBGREQUEST3 – 8000_0000h base + 1190h offset = 8000_1190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_DBGREQUEST3 field descriptions

Field	Description
31–0 BITS	Bits 96-127 of the request holding register.

5.4.146 Interrupt Collector Version Register (HW_ICOLL_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_ICOLL_VERSION.B.MAJOR != 3)
   Error();
```

Address: HW_ICOLL_VERSION – 8000_0000h base + 11E0h offset = 8000_11E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ICOLL_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 6
AHB-to-APBH Bridge with DMA (APBH-Bridge-DMA)

6.1 AHB-to-APBH Bridge Overview

The AHB-to-APBH bridge provides the i.MX28 with an inexpensive peripheral attachment bus running on the AHB's HCLK. (The H in APBH denotes that the APB*H* is synchronous to HCLK, as compared to APB*X*, which runs on the crystal-derived XCLK.)

As shown in the following figure, the AHB-to-APBH bridge includes the AHB-to-APB PIO bridge for a memory-mapped I/O to the APB devices, as well as a central DMA facility for devices on this bus and a vectored interrupt controller for the ARM926 core. Each one of the APB peripherals, including the vectored interrupt controller, is documented in their own chapters elsewhere in this document.

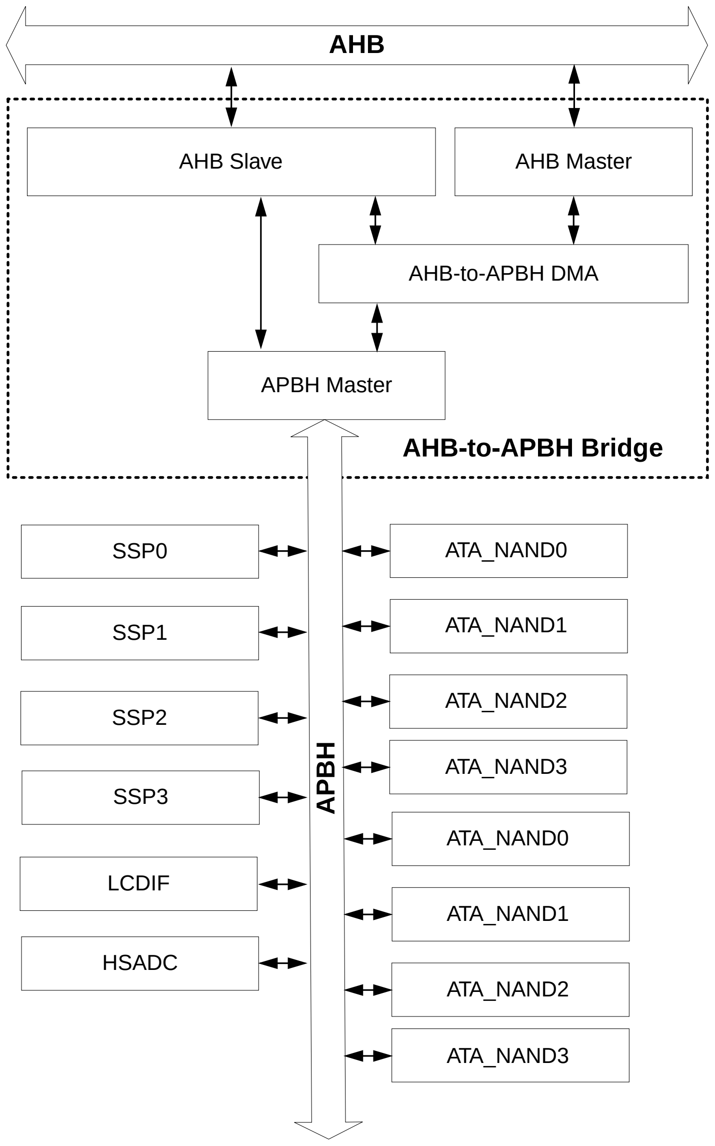


Figure 6-1. AHB-to-APBH Bridge DMA Block Diagram

The DMA controller uses the APBH bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBH bus and the AHB-to-APB bridge functions' use of the APBH is mediated by an internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report "not ready" through its HREADY output until the bridge transfer can complete. The arbiter tracks repeated lockouts and inverts the priority, guaranteeing the CPU every fourth transfer on the APB.

6.2 APBH DMA

The DMA supports sixteen channels of DMA services, as shown in the following table. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the general structure, as shown in Figure 6-2.

Table 6-1. APBH DMA Channel Assignments

APBH DMA CHANNEL #	USAGE
0	SSP0
1	SSP1
2	SSP2
3	SSP3
4	GPMI0
5	GPMI1
6	GPMI2
7	GPMI3
8	GPMI4
9	GPMI5
10	GPMI6
11	GPMI7
12	HSADC
13	LCDIF
14	EMPTY
15	EMPTY

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA, and have no further concern for the device until the DMA completion interrupt occurs. The goal is to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 KHz (arrival intervals longer than 1 ms).

A single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and the controls, it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the GPMI controller to send NAND command bytes, address bytes, and data transfers where the command and the address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA. Each DMA structure can have 0–15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle.

The DMA master generates only normal read/write transfers to the APBH. It does *not* generate set, clear, or toggle (SCT) transfers.

After any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure.

Command chains are built up using the general structure, as shown in the figure.

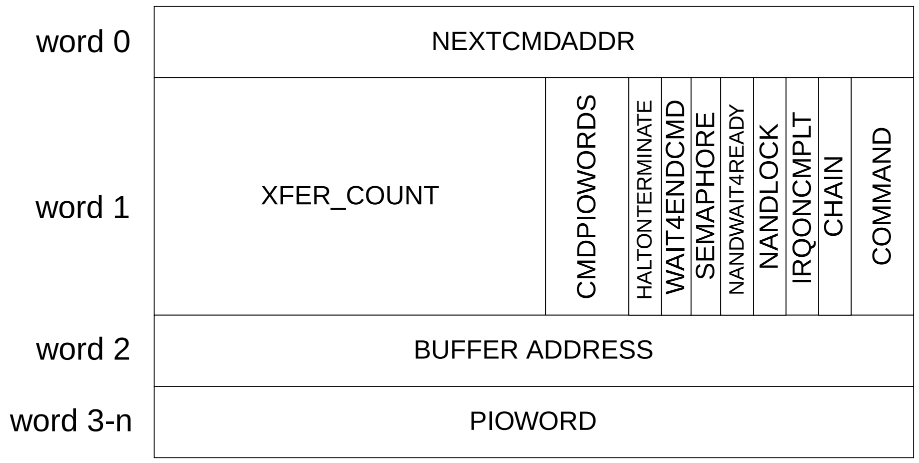


Figure 6-2. AHB-to-APBH Bridge DMA Channel Command Structure

The following table shows the four commands implemented by the DMA.

Table 6-2. APBH DMA Commands

DMA COMMAND	USAGE
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, then perform a DMA transfer from the peripheral for the specified number of bytes.
10	DMA_READ. Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

DMA COMMAND	USAGE
11	DMA_SENSE. Perform any requested PIO word transfers, then perform a conditional branch to the next chained device. Follow the NEXTCMD_ADDR pointer if the peripheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true. This command becomes a no-operation for any channel other than a GPMI channel.

DMA_WRITE operations copy data bytes to the system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from the system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers. This command is useful in such applications as activating the NAND devices CHECKSTATUS operation. The check status command in the NAND peripheral reads a status byte from the NAND device, performs an XOR and MASK against an expected value supplied as part of the PIO transfer. Once the read check completes (see NAND Read Status Polling Example), the NO_DMA_XFER command completes. The result in the peripheral is that its PSENSE line is driven by the results of the comparison. The sense flip-flop is only updated by CHECKSTATUS for the device that is executed. At some future point, the chain contains a DMA command structure with the fourth and final command value, that is, the DMA_SENSE command.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. The DMA_SENSE command uses the DMA buffer pointer word of the command structure to point to an alternate DMA command structure chain or list. The DMA_SENSE command examines the sense line of the associated peripheral. If the sense line is false, then the DMA follows the standard list found whose next command is found from the pointer in the NEXTCMD_ADDR word of the command structure. If the sense line is true, then the DMA follows the alternate list whose next command is found from the pointer in the DMA Buffer Pointer word of the DMA_SENSE command structure (see Figure 6-2). The sense command ignores the CHAIN bit, so that both pointers must be valid when the DMA comes to a sense command.

If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel waits for the device to signal completion of a command by toggling the endcmcd signal before proceeding to load and execute the next command structure. Then, if DECREMENT_SEMAPHORE is set, the semaphore will be decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in the following table, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer-count mechanism is duplicated in the associated peripheral, either as an implied or as a specified count in the peripheral.

Table 6-3. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															
Number DMA Bytes to Transfer																Number PIO Words to Write							HALTONTERMINATE	WAIT4ENDCMD	DECREMENT SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQ_COMPLETE	CHAIN	COMMAND	
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBH Bus																															

Figure 6-2 also shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1, if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it is not detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt-status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by the software. It can be used to interrupt the CPU.

The NAND_LOCK bit is monitored by the DMA channel arbiter. After a NAND channel (from channel 4 to channel 11) succeeds in the arbiter with its NAND_LOCK bit set, then the arbiter ignores the other seven NAND channels until a command is completed in which the NAND_LOCK is not set. Notice that the semantic here is that the NAND_LOCK state is to limit scheduling of a non-locked DMA. A DMA channel can go from unlocked to locked in the arbiter at the beginning of a command when the NAND_LOCK bit is set.

When the last DMA command of an atomic sequence is completed, the lock should be removed. To accomplish this, the last command does not have the NAND_LOCK bit. It is still locked in the atomic state within the arbiter when the command starts, so that it is the only NAND command that can be executed. At the end, it drops from the atomic state within the arbiter.

The NAND_WAIT4READY bit also has a special use for DMA channels (from channel 4 to channel 11), that is, the NAND device channels. The NAND device supplies a sample of the ready line for the NAND device. This ready value is used to hold off of a command with this bit set until the ready line is asserted to 1. Once the arbiter sees a command with a wait-for-ready set, it holds off that channel until ready is asserted.

Receiving an IRQ for HALTONTERMINATE (HOT) is a new feature in the APBH/X DMA descriptor that allows certain peripheral block (for example, GPMI, SSP, I2C) to signal to the DMA engine that an error has occurred. In prior chips, if a block is stalled due to an error, the only practical way to discover this in software was through a timer of some sort, or to poll the block. Now, an HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed. Note not all peripheral block support this termination feature.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH/X channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:
    - Reset the channel.
    - Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, and so on).

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run, process commands and perform DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by the software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBH_CHn_NXTCMDAR register (next command address register) to fetch a pointer to the next command to process. NOTE: This is a double indirect case. This method allows the software to append to a running command list under the protection of the counting semaphore.

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBH_CHn_NXTCMDAR register, and then writes 1 to the counting semaphore in HW_APBH_CHn_SEMA. The DMA channel loads HW_APBH_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When the software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBH_CHn_CURCMDAR at any time to determine the location of the command structure currently being processed.

6.3 Implementation Examples

6.3.1 NAND Read Status Polling Example

The following figure shows a more complicated scenario. This subset of a NAND device workload shows that the first two command structures are used during the data-write phase of an NAND device write operation (CLE and ALE transfers omitted for clarity).

- After writing the data, one must wait until the NAND device status register indicates that the write charge has been transferred. This is built into the workload using a check status command in the NAND in a loop created from the next two DMA command structures.
- The NO_DMA_TRANSFER command is shown here performing the read check, followed by a DMA_SENSE command to branch the DMA command structure list, based on the status of a bit in the external NAND device.

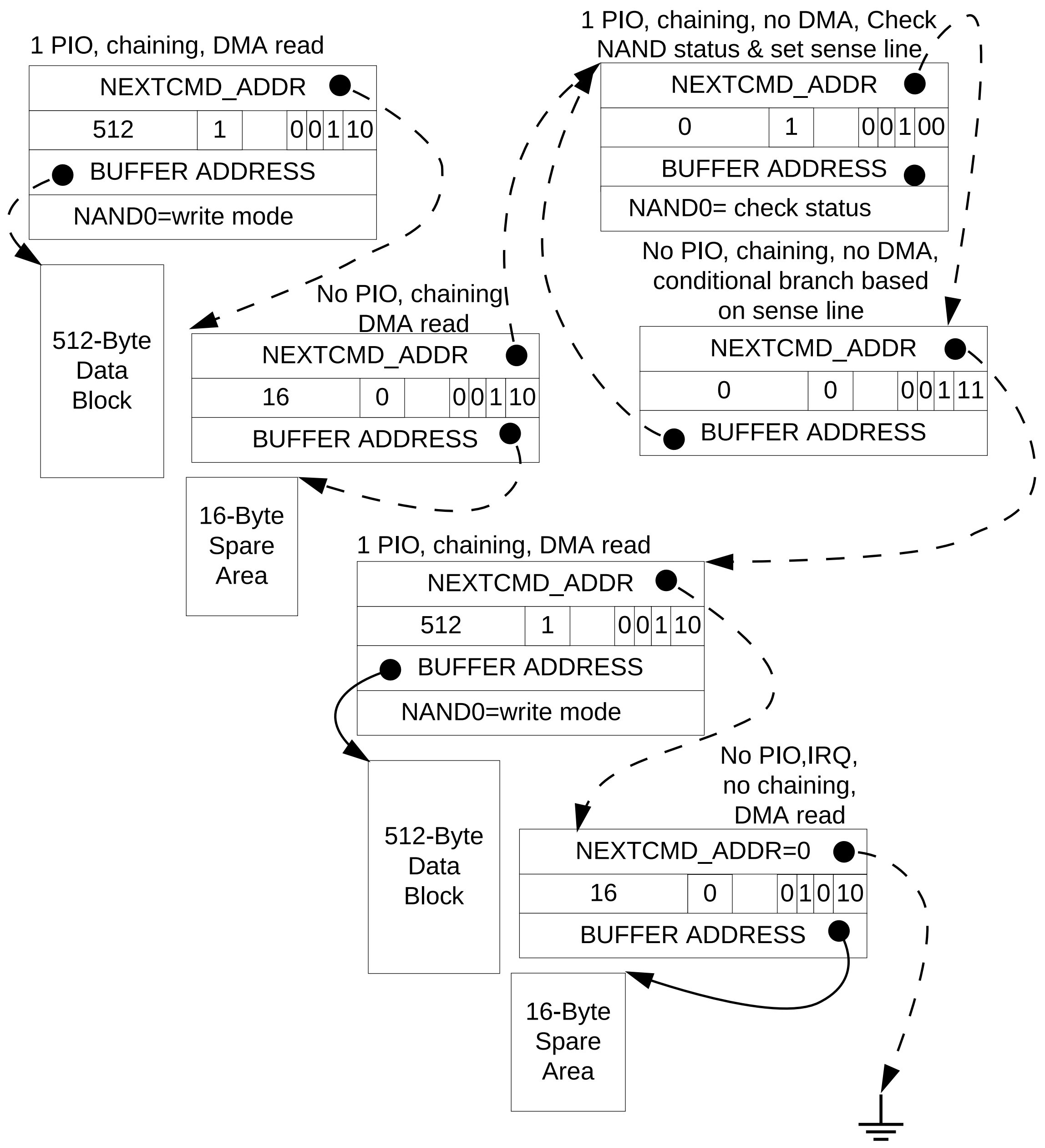


Figure 6-3. AHB-to-APBH Bridge DMA NAND Read Status Polling with DMA Sense Command

The example in the above figure shows the workload continuing immediately to the next NAND page transfer. However, one could perform a second sense operation to see if an error has occurred after the write. One could then point the sense command alternate branch at a NO_DMA_XFER command with the interrupt bit set. If the CHAIN bit is not set on this failure branch, then the CPU is interrupted immediately, and the channel process is also immediately terminated in the presence of a workload-detected NAND error bit.

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBH bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register.

To start DMA processing for the first command, initialize the PIO registers of the desired channel, as follows:

- First, load the next command address register with a pointer to the first command to be loaded.
- Then, write 1 to the counting semaphore register. This causes the DMA to schedule the targeted channel for the DMA command structure load, as if it just finished its previous command.

6.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

6.5 Programmable Registers

APBH Hardware Register Format Summary

HW_APBH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_4000	AHB to APBH Bridge Control and Status Register 0 (HW_APBH_CTRL0)	32	R/W	E000_0000h	6.5.1/358
8000_4010	AHB to APBH Bridge Control and Status Register 1 (HW_APBH_CTRL1)	32	R/W	0000_0000h	6.5.2/359
8000_4020	AHB to APBH Bridge Control and Status Register 2 (HW_APBH_CTRL2)	32	R/W	0000_0000h	6.5.3/363
8000_4030	AHB to APBH Bridge Channel Register (HW_APBH_CHANNEL_CTRL)	32	R/W	0000_0000h	6.5.4/367
8000_4040	AHB to APBH DMA Device Assignment Register (HW_APBH_DEVSEL)	32	R	0000_0000h	6.5.5/369
8000_4050	AHB to APBH DMA burst size (HW_APBH_DMA_BURST_SIZE)	32	R/W	0055_5555h	6.5.6/370
8000_4060	AHB to APBH DMA Debug Register (HW_APBH_DEBUG)	32	R/W	0000_0000h	6.5.7/371

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_4100	APBH DMA Channel 0 Current Command Address Register (HW_APBH_CH0_CURCMDAR)	32	R	0000_0000h	6.5.8/372
8000_4110	APBH DMA Channel 0 Next Command Address Register (HW_APBH_CH0_NXTCMDAR)	32	R/W	0000_0000h	6.5.9/373
8000_4120	APBH DMA Channel 0 Command Register (HW_APBH_CH0_CMD)	32	R	0000_0000h	6.5.10/373
8000_4130	APBH DMA Channel 0 Buffer Address Register (HW_APBH_CH0_BAR)	32	R	0000_0000h	6.5.11/376
8000_4140	APBH DMA Channel 0 Semaphore Register (HW_APBH_CH0_SEMA)	32	R/W	0000_0000h	6.5.12/376
8000_4150	AHB to APBH DMA Channel 0 Debug Information (HW_APBH_CH0_DEBUG1)	32	R	00A0_0000h	6.5.13/377
8000_4160	AHB to APBH DMA Channel 0 Debug Information (HW_APBH_CH0_DEBUG2)	32	R	0000_0000h	6.5.14/380
8000_4170	APBH DMA Channel 1 Current Command Address Register (HW_APBH_CH1_CURCMDAR)	32	R/W	0000_0000h	6.5.15/380
8000_4180	APBH DMA Channel 1 Next Command Address Register (HW_APBH_CH1_NXTCMDAR)	32	R/W	0000_0000h	6.5.16/381
8000_4190	APBH DMA Channel 1 Command Register (HW_APBH_CH1_CMD)	32	R	0000_0000h	6.5.17/382
8000_41A0	APBH DMA Channel 1 Buffer Address Register (HW_APBH_CH1_BAR)	32	R	0000_0000h	6.5.18/384
8000_41B0	APBH DMA Channel 1 Semaphore Register (HW_APBH_CH1_SEMA)	32	R/W	0000_0000h	6.5.19/385
8000_41C0	AHB to APBH DMA Channel 1 Debug Information (HW_APBH_CH1_DEBUG1)	32	R	00A0_0000h	6.5.20/386
8000_41D0	AHB to APBH DMA Channel 1 Debug Information (HW_APBH_CH1_DEBUG2)	32	R	0000_0000h	6.5.21/388
8000_41E0	APBH DMA Channel 2 Current Command Address Register (HW_APBH_CH2_CURCMDAR)	32	R	0000_0000h	6.5.22/389
8000_41F0	APBH DMA Channel 2 Next Command Address Register (HW_APBH_CH2_NXTCMDAR)	32	R/W	0000_0000h	6.5.23/389
8000_4200	APBH DMA Channel 2 Command Register (HW_APBH_CH2_CMD)	32	R	0000_0000h	6.5.24/390
8000_4210	APBH DMA Channel 2 Buffer Address Register (HW_APBH_CH2_BAR)	32	R	0000_0000h	6.5.25/392
8000_4220	APBH DMA Channel 2 Semaphore Register (HW_APBH_CH2_SEMA)	32	R/W	0000_0000h	6.5.26/393
8000_4230	AHB to APBH DMA Channel 2 Debug Information (HW_APBH_CH2_DEBUG1)	32	R	00A0_0000h	6.5.27/393
8000_4240	AHB to APBH DMA Channel 2 Debug Information (HW_APBH_CH2_DEBUG2)	32	R	0000_0000h	6.5.28/396

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_4250	APBH DMA Channel 3 Current Command Address Register (HW_APBH_CH3_CURCMDAR)	32	R	0000_0000h	6.5.29/396
8000_4260	APBH DMA Channel 3 Next Command Address Register (HW_APBH_CH3_NXTCMDAR)	32	R/W	0000_0000h	6.5.30/397
8000_4270	APBH DMA Channel 3 Command Register (HW_APBH_CH3_CMD)	32	R	0000_0000h	6.5.31/398
8000_4280	APBH DMA Channel 3 Buffer Address Register (HW_APBH_CH3_BAR)	32	R	0000_0000h	6.5.32/400
8000_4290	APBH DMA Channel 3 Semaphore Register (HW_APBH_CH3_SEMA)	32	R/W	0000_0000h	6.5.33/400
8000_42A0	AHB to APBH DMA Channel 3 Debug Information (HW_APBH_CH3_DEBUG1)	32	R	00A0_0000h	6.5.34/401
8000_42B0	AHB to APBH DMA Channel 3 Debug Information (HW_APBH_CH3_DEBUG2)	32	R	0000_0000h	6.5.35/403
8000_42C0	APBH DMA Channel 4 Current Command Address Register (HW_APBH_CH4_CURCMDAR)	32	R	0000_0000h	6.5.36/404
8000_42D0	APBH DMA Channel 4 Next Command Address Register (HW_APBH_CH4_NXTCMDAR)	32	R/W	0000_0000h	6.5.37/405
8000_42E0	APBH DMA Channel 4 Command Register (HW_APBH_CH4_CMD)	32	R	0000_0000h	6.5.38/405
8000_42F0	APBH DMA Channel 4 Buffer Address Register (HW_APBH_CH4_BAR)	32	R	0000_0000h	6.5.39/407
8000_4300	APBH DMA Channel 4 Semaphore Register (HW_APBH_CH4_SEMA)	32	R/W	0000_0000h	6.5.40/408
8000_4310	AHB to APBH DMA Channel 4 Debug Information (HW_APBH_CH4_DEBUG1)	32	R	00A0_0000h	6.5.41/409
8000_4320	AHB to APBH DMA Channel 4 Debug Information (HW_APBH_CH4_DEBUG2)	32	R	0000_0000h	6.5.42/411
8000_4330	APBH DMA Channel 5 Current Command Address Register (HW_APBH_CH5_CURCMDAR)	32	R	0000_0000h	6.5.43/412
8000_4340	APBH DMA Channel 5 Next Command Address Register (HW_APBH_CH5_NXTCMDAR)	32	R/W	0000_0000h	6.5.44/412
8000_4350	APBH DMA Channel 5 Command Register (HW_APBH_CH5_CMD)	32	R	0000_0000h	6.5.45/413
8000_4360	APBH DMA Channel 5 Buffer Address Register (HW_APBH_CH5_BAR)	32	R	0000_0000h	6.5.46/415
8000_4370	APBH DMA Channel 5 Semaphore Register (HW_APBH_CH5_SEMA)	32	R/W	0000_0000h	6.5.47/416
8000_4380	AHB to APBH DMA Channel 5 Debug Information (HW_APBH_CH5_DEBUG1)	32	R	00A0_0000h	6.5.48/417
8000_4390	AHB to APBH DMA Channel 5 Debug Information (HW_APBH_CH5_DEBUG2)	32	R	0000_0000h	6.5.49/419

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_43A0	APBH DMA Channel 6 Current Command Address Register (HW_APBH_CH6_CURCMDAR)	32	R	0000_0000h	6.5.50/420
8000_43B0	APBH DMA Channel 6 Next Command Address Register (HW_APBH_CH6_NXTCMDAR)	32	R/W	0000_0000h	6.5.51/420
8000_43C0	APBH DMA Channel 6 Command Register (HW_APBH_CH6_CMD)	32	R	0000_0000h	6.5.52/421
8000_43D0	APBH DMA Channel 6 Buffer Address Register (HW_APBH_CH6_BAR)	32	R	0000_0000h	6.5.53/423
8000_43E0	APBH DMA Channel 6 Semaphore Register (HW_APBH_CH6_SEMA)	32	R/W	0000_0000h	6.5.54/423
8000_43F0	AHB to APBH DMA Channel 6 Debug Information (HW_APBH_CH6_DEBUG1)	32	R	00A0_0000h	6.5.55/424
8000_4400	AHB to APBH DMA Channel 6 Debug Information (HW_APBH_CH6_DEBUG2)	32	R	0000_0000h	6.5.56/427
8000_4410	APBH DMA Channel 7 Current Command Address Register (HW_APBH_CH7_CURCMDAR)	32	R	0000_0000h	6.5.57/427
8000_4420	APBH DMA Channel 7 Next Command Address Register (HW_APBH_CH7_NXTCMDAR)	32	R/W	0000_0000h	6.5.58/428
8000_4430	APBH DMA Channel 7 Command Register (HW_APBH_CH7_CMD)	32	R	0000_0000h	6.5.59/428
8000_4440	APBH DMA Channel 7 Buffer Address Register (HW_APBH_CH7_BAR)	32	R	0000_0000h	6.5.60/430
8000_4450	APBH DMA Channel 7 Semaphore Register (HW_APBH_CH7_SEMA)	32	R/W	0000_0000h	6.5.61/431
8000_4460	AHB to APBH DMA Channel 7 Debug Information (HW_APBH_CH7_DEBUG1)	32	R	00A0_0000h	6.5.62/432
8000_4470	AHB to APBH DMA Channel 7 Debug Information (HW_APBH_CH7_DEBUG2)	32	R	0000_0000h	6.5.63/434
8000_4480	APBH DMA Channel 8 Current Command Address Register (HW_APBH_CH8_CURCMDAR)	32	R	0000_0000h	6.5.64/435
8000_4490	APBH DMA Channel 8 Next Command Address Register (HW_APBH_CH8_NXTCMDAR)	32	R/W	0000_0000h	6.5.65/435
8000_44A0	APBH DMA Channel 8 Command Register (HW_APBH_CH8_CMD)	32	R	0000_0000h	6.5.66/436
8000_44B0	APBH DMA Channel 8 Buffer Address Register (HW_APBH_CH8_BAR)	32	R	0000_0000h	6.5.67/438
8000_44C0	APBH DMA Channel 8 Semaphore Register (HW_APBH_CH8_SEMA)	32	R/W	0000_0000h	6.5.68/439
8000_44D0	AHB to APBH DMA Channel 8 Debug Information (HW_APBH_CH8_DEBUG1)	32	R	00A0_0000h	6.5.69/440
8000_44E0	AHB to APBH DMA Channel 8 Debug Information (HW_APBH_CH8_DEBUG2)	32	R	0000_0000h	6.5.70/442

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_44F0	APBH DMA Channel 9 Current Command Address Register (HW_APBH_CH9_CURCMDAR)	32	R	0000_0000h	6.5.71/443
8000_4500	APBH DMA Channel 9 Next Command Address Register (HW_APBH_CH9_NXTCMDAR)	32	R/W	0000_0000h	6.5.72/443
8000_4510	APBH DMA Channel 9 Command Register (HW_APBH_CH9_CMD)	32	R	0000_0000h	6.5.73/444
8000_4520	APBH DMA Channel 9 Buffer Address Register (HW_APBH_CH9_BAR)	32	R	0000_0000h	6.5.74/446
8000_4530	APBH DMA Channel 9 Semaphore Register (HW_APBH_CH9_SEMA)	32	R/W	0000_0000h	6.5.75/447
8000_4540	AHB to APBH DMA Channel 9 Debug Information (HW_APBH_CH9_DEBUG1)	32	R	00A0_0000h	6.5.76/447
8000_4550	AHB to APBH DMA Channel 9 Debug Information (HW_APBH_CH9_DEBUG2)	32	R	0000_0000h	6.5.77/450
8000_4560	APBH DMA channel 10 Current Command Address Register (HW_APBH_CH10_CURCMDAR)	32	R	0000_0000h	6.5.78/450
8000_4570	APBH DMA channel 10 Next Command Address Register (HW_APBH_CH10_NXTCMDAR)	32	R/W	0000_0000h	6.5.79/451
8000_4580	APBH DMA channel 10 Command Register (HW_APBH_CH10_CMD)	32	R	0000_0000h	6.5.80/451
8000_4590	APBH DMA channel 10 Buffer Address Register (HW_APBH_CH10_BAR)	32	R	0000_0000h	6.5.81/454
8000_45A0	APBH DMA channel 10 Semaphore Register (HW_APBH_CH10_SEMA)	32	R/W	0000_0000h	6.5.82/454
8000_45B0	AHB to APBH DMA channel 10 Debug Information (HW_APBH_CH10_DEBUG1)	32	R	00A0_0000h	6.5.83/455
8000_45C0	AHB to APBH DMA channel 10 Debug Information (HW_APBH_CH10_DEBUG2)	32	R	0000_0000h	6.5.84/457
8000_45D0	APBH DMA Channel 11 Current Command Address Register (HW_APBH_CH11_CURCMDAR)	32	R	0000_0000h	6.5.85/458
8000_45E0	APBH DMA Channel 11 Next Command Address Register (HW_APBH_CH11_NXTCMDAR)	32	R/W	0000_0000h	6.5.86/458
8000_45F0	APBH DMA Channel 11 Command Register (HW_APBH_CH11_CMD)	32	R	0000_0000h	6.5.87/459
8000_4600	APBH DMA Channel 11 Buffer Address Register (HW_APBH_CH11_BAR)	32	R	0000_0000h	6.5.88/461
8000_4610	APBH DMA Channel 11 Semaphore Register (HW_APBH_CH11_SEMA)	32	R/W	0000_0000h	6.5.89/462
8000_4620	AHB to APBH DMA Channel 11 Debug Information (HW_APBH_CH11_DEBUG1)	32	R	00A0_0000h	6.5.90/463
8000_4630	AHB to APBH DMA Channel 11 Debug Information (HW_APBH_CH11_DEBUG2)	32	R	0000_0000h	6.5.91/465

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_4640	APBH DMA channel 12 Current Command Address Register (HW_APBH_CH12_CURCMDAR)	32	R	0000_0000h	6.5.92/466
8000_4650	APBH DMA channel 12 Next Command Address Register (HW_APBH_CH12_NXTCMDAR)	32	R/W	0000_0000h	6.5.93/466
8000_4660	APBH DMA channel 12 Command Register (HW_APBH_CH12_CMD)	32	R	0000_0000h	6.5.94/467
8000_4670	APBH DMA channel 12 Buffer Address Register (HW_APBH_CH12_BAR)	32	R	0000_0000h	6.5.95/469
8000_4680	APBH DMA channel 12 Semaphore Register (HW_APBH_CH12_SEMA)	32	R/W	0000_0000h	6.5.96/470
8000_4690	AHB to APBH DMA channel 12 Debug Information (HW_APBH_CH12_DEBUG1)	32	R	00A0_0000h	6.5.97/470
8000_46A0	AHB to APBH DMA channel 12 Debug Information (HW_APBH_CH12_DEBUG2)	32	R	0000_0000h	6.5.98/473
8000_46B0	APBH DMA Channel 13 Current Command Address Register (HW_APBH_CH13_CURCMDAR)	32	R	0000_0000h	6.5.99/473
8000_46C0	APBH DMA Channel 13 Next Command Address Register (HW_APBH_CH13_NXTCMDAR)	32	R/W	0000_0000h	6.5.100/474
8000_46D0	APBH DMA Channel 13 Command Register (HW_APBH_CH13_CMD)	32	R	0000_0000h	6.5.101/474
8000_46E0	APBH DMA Channel 13 Buffer Address Register (HW_APBH_CH13_BAR)	32	R	0000_0000h	6.5.102/477
8000_46F0	APBH DMA Channel 13 Semaphore Register (HW_APBH_CH13_SEMA)	32	R/W	0000_0000h	6.5.103/477
8000_4700	AHB to APBH DMA Channel 13 Debug Information (HW_APBH_CH13_DEBUG1)	32	R	00A0_0000h	6.5.104/478
8000_4710	AHB to APBH DMA Channel 13 Debug Information (HW_APBH_CH13_DEBUG2)	32	R	0000_0000h	6.5.105/480
8000_4720	APBH DMA channel 14 Current Command Address Register (HW_APBH_CH14_CURCMDAR)	32	R	0000_0000h	6.5.106/481
8000_4730	APBH DMA channel 14 Next Command Address Register (HW_APBH_CH14_NXTCMDAR)	32	R/W	0000_0000h	6.5.107/481
8000_4740	APBH DMA channel 14 Command Register (HW_APBH_CH14_CMD)	32	R	0000_0000h	6.5.108/482
8000_4750	APBH DMA channel 14 Buffer Address Register (HW_APBH_CH14_BAR)	32	R	0000_0000h	6.5.109/484
8000_4760	APBH DMA channel 14 Semaphore Register (HW_APBH_CH14_SEMA)	32	R/W	0000_0000h	6.5.110/485
8000_4770	AHB to APBH DMA channel 14 Debug Information (HW_APBH_CH14_DEBUG1)	32	R	0000_0000h	6.5.111/485
8000_4780	AHB to APBH DMA channel 14 Debug Information (HW_APBH_CH14_DEBUG2)	32	R	0000_0000h	6.5.112/488

HW_APBH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_4790	APBH DMA channel 15 Current Command Address Register (HW_APBH_CH15_CURCMDAR)	32	R	0000_0000h	6.5.113/488
8000_47A0	APBH DMA channel 15 Next Command Address Register (HW_APBH_CH15_NXTCMDAR)	32	R/W	0000_0000h	6.5.114/489
8000_47B0	APBH DMA channel 15 Command Register (HW_APBH_CH15_CMD)	32	R	0000_0000h	6.5.115/490
8000_47C0	APBH DMA channel 15 Buffer Address Register (HW_APBH_CH15_BAR)	32	R	0000_0000h	6.5.116/492
8000_47D0	APBH DMA channel 15 Semaphore Register (HW_APBH_CH15_SEMA)	32	R/W	0000_0000h	6.5.117/492
8000_47E0	AHB to APBH DMA channel 15 Debug Information (HW_APBH_CH15_DEBUG1)	32	R	0000_0000h	6.5.118/493
8000_47F0	AHB to APBH DMA channel 15 Debug Information (HW_APBH_CH15_DEBUG2)	32	R	0000_0000h	6.5.119/495
8000_4800	APBH Bridge Version Register (HW_APBH_VERSION)	32	R	0301_0000h	6.5.120/496

6.5.1 AHB to APBH Bridge Control and Status Register 0 (HW_APBH_CTRL0)

The APBH CTRL 0 provides overall control of the AHB to APBH bridge and DMA.

HW_APBH_CTRL0: 0x000

HW_APBH_CTRL0_SET: 0x004

HW_APBH_CTRL0_CLR: 0x008

HW_APBH_CTRL0_TOG: 0x00C

This register contains module softreset, clock gating, channel clock gating/freeze bits.

Address: HW_APBH_CTRL0 – 8000_4000h base + 0h offset = 8000_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	AHB_ BURST8_EN	APB_ BURST_EN	RSVD0											
W					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLKGATE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CTRL0 field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal APBH DMA operation. Set this bit to one (default) to disable clocking with the APBH DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBH DMA block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 AHB_BURST8_ EN	Set this bit to one (default) to enable AHB 8-beat burst. Set to zero to disable 8-beat burst on AHB interface.
28 APB_BURST_EN	Set this bit to one to enable apb master do a continous transfers when a device request a burst dma. Set to zero will treat a burst dma request as 4/8 individual requests.
27–16 RSVD0	Reserved, always set to zero.
15–0 CLKGATE_ CHANNEL	These bits must be set to zero for normal operation of each channel. When set to one they gate off the individual clocks to the channels. 0x0001 **SSP0** — 0x0002 **SSP1** — 0x0004 **SSP2** — 0x0008 **SSP3** — 0x0010 **NAND0** — 0x0020 **NAND1** — 0x0040 **NAND2** — 0x0080 **NAND3** — 0x0100 **NAND4** — 0x0200 **NAND5** — 0x0400 **NAND6** — 0x0800 **NAND7** — 0x1000 **HSADC** — 0x2000 **LCDIF** —

6.5.2 AHB to APBH Bridge Control and Status Register 1 (HW_APBH_CTRL1)

The APBH CTRL one provides overall control of the interrupts generated by the AHB to APBH DMA.

HW_APBH_CTRL1: 0x010

HW_APBH_CTRL1_SET: 0x014

HW_APBH_CTRL1_CLR: 0x018

HW_APBH_CTRL1_TOG: 0x01C

This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register struct's bitfield
```

Address: HW_APBH_CTRL1 – 8000_4000h base + 10h offset = 8000_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CTRL1 field descriptions

Field	Description
31 CH15_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 15.
30 CH14_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 14.
29 CH13_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 13.

HW_APBH_CTRL1 field descriptions (continued)

Field	Description
28 CH12_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 12.
27 CH11_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 11.
26 CH10_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 10.
25 CH9_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 9.
24 CH8_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 8.
23 CH7_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 7.
22 CH6_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 6.
21 CH5_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 5.
20 CH4_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 4.
19 CH3_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 3.
18 CH2_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 2.
17 CH1_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 1.

HW_APBH_CTRL1 field descriptions (continued)

Field	Description
16 CH0_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBH DMA channel 0.
15 CH15_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14 CH14_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
13 CH13_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12 CH12_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11 CH11_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10 CH10_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9 CH9_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8 CH8_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7 CH7_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6 CH6_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5 CH5_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4 CH4_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3 CH3_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2 CH2_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

HW_APBH_CTRL1 field descriptions (continued)

Field	Description
1 CH1_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0 CH0_ CMDCMPLT_IRQ	Interrupt request status bit for APBH DMA channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

6.5.3 AHB to APBH Bridge Control and Status Register 2 (HW_APBH_CTRL2)

The APBH CTRL 2 provides channel error interrupts generated by the AHB to APBH DMA.

HW_APBH_CTRL2: 0x020

HW_APBH_CTRL2_SET: 0x024

HW_APBH_CTRL2_CLR: 0x028

HW_APBH_CTRL2_TOG: 0x02C

This register contains the per channel interrupt status bits and the per channel interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
BF_WR(APBH_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
BF_APBH_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register struct's bitfield
```

Address: HW_APBH_CTRL2 – 8000_4000h base + 20h offset = 8000_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CTRL2 field descriptions

Field	Description
31 CH15_ERROR_ STATUS	Error status bit for APBH DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
30 CH14_ERROR_ STATUS	Error status bit for APBH DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
29 CH13_ERROR_ STATUS	Error status bit for APBH DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
28 CH12_ERROR_ STATUS	Error status bit for APBH DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
27 CH11_ERROR_ STATUS	Error status bit for APBH DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
26 CH10_ERROR_ STATUS	Error status bit for APBH DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.

HW_APBH_CTRL2 field descriptions (continued)

Field	Description
25 CH9_ERROR_ STATUS	Error status bit for APBH DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
24 CH8_ERROR_ STATUS	Error status bit for APBH DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
23 CH7_ERROR_ STATUS	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
22 CH6_ERROR_ STATUS	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
21 CH5_ERROR_ STATUS	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
20 CH4_ERROR_ STATUS	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
19 CH3_ERROR_ STATUS	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
18 CH2_ERROR_ STATUS	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination.

HW_APBH_CTRL2 field descriptions (continued)

Field	Description
	0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
17 CH1_ERROR_ STATUS	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
16 CH0_ERROR_ STATUS	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
15 CH15_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
14 CH14_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13 CH13_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12 CH12_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
11 CH11_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10 CH10_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9 CH9_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8 CH8_ERROR_ IRQ	Error interrupt status bit for APBH DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7 CH7_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6 CH6_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

HW_APBH_CTRL2 field descriptions (continued)

Field	Description
5 CH5_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4 CH4_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3 CH3_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2 CH2_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
1 CH1_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0 CH0_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

6.5.4 AHB to APBH Bridge Channel Register (HW_APBH_CHANNEL_CTRL)

The APBH CHANNEL CTRL provides reset/freeze control of each DMA channel.

HW_APBH_CHANNEL_CTRL: 0x030

HW_APBH_CHANNEL_CTRL_SET: 0x034

HW_APBH_CHANNEL_CTRL_CLR: 0x038

HW_APBH_CHANNEL_CTRL_TOG: 0x03C

This register contains individual channel reset/freeze bits.

Address: HW_APBH_CHANNEL_CTRL – 8000_4000h base + 30h offset = 8000_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RESET_CHANNEL																FREEZE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CHANNEL_CTRL field descriptions

Field	Description
31–16 RESET_ CHANNEL	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. 0x0001 **SSP0** — 0x0002 **SSP1** — 0x0004 **SSP2** — 0x0008 **SSP3** — 0x0010 **NAND0** — 0x0020 **NAND1** — 0x0040 **NAND2** — 0x0080 **NAND3** — 0x0100 **NAND4** — 0x0200 **NAND5** — 0x0400 **NAND6** — 0x0800 **NAND7** — 0x1000 **HSADC** — 0x2000 **LCDIF** —
15–0 FREEZE_ CHANNEL	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is denied access to the central DMA resources. Note: 1. DMA PIO write to associated peripheral is not supported when Freeze bit is set, and use ARM instead to configure peripheral. 2. After FREEZE bit is set, no more access, neither AHB access to memory nor APB access to peripherals, will be allowed by arbiter. But, there might be on-going channel access exactly when FREEZE bit is set, either INCR8/INCR4/SINGLE AHB access or APB peripheral access, this on-going access will not be affected by FREEZE bit and will finish as normal. That is to say, setting FREEZE bit might not freeze channel access immediately, it only freezes further channel access, and you have to wait a while to freeze channel access completely. To make sure that there is no more access from freezed channel, channel state machine should be checked by reading channel DEBUG1 register, wait till state stunk at any of IDLE, READ_REQ, WRITE, or CHAIN_WAIT. 0x0001 **SSP0** — 0x0002 **SSP1** — 0x0004 **SSP2** — 0x0008 **SSP3** — 0x0010 **NAND0** — 0x0020 **NAND1** — 0x0040 **NAND2** — 0x0080 **NAND3** — 0x0100 **NAND4** — 0x0200 **NAND5** — 0x0400 **NAND6** — 0x0800 **NAND7** — 0x1000 **HSADC** — 0x2000 **LCDIF** —

6.5.5 AHB to APBH DMA Device Assignment Register (HW_APBH_DEVSEL)

This register allows reassignment of the APBH device connected to the DMA Channels.

In this chip, apbhdma channel resource is enough for high speed peripherals, so this register is of no use and reserved.

Address: HW_APBH_DEVSEL – 8000_4000h base + 40h offset = 8000_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
W	▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_DEVSEL field descriptions

Field	Description
31–30 CH15	Reserved.
29–28 CH14	Reserved.
27–26 CH13	Reserved.
25–24 CH12	Reserved.
23–22 CH11	Reserved.
21–20 CH10	Reserved.
19–18 CH9	Reserved.
17–16 CH8	Reserved.
15–14 CH7	Reserved.
13–12 CH6	Reserved.
11–10 CH5	Reserved.
9–8 CH4	Reserved.
7–6 CH3	Reserved.

HW_APBH_DEVSEL field descriptions (continued)

Field	Description
5–4 CH2	Reserved.
3–2 CH1	Reserved.
1–0 CH0	Reserved.

6.5.6 AHB to APBH DMA burst size (HW_APBH_DMA_BURST_SIZE)

This register programs the apbh burst size of the APBH DMA devices when a DMA burst request is issued.

It provides a mechanism for improving bandwidth between DMA and device if device's FIFO is large enough.

Address: HW_APBH_DMA_BURST_SIZE – 8000_4000h base + 50h offset = 8000_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
W	▨		▨		▨		▨																									
Reset	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

HW_APBH_DMA_BURST_SIZE field descriptions

Field	Description
31–30 CH15	Reserved.
29–28 CH14	Reserved.
27–26 CH13	Reserved. HSADC not support DMA burst request.
25–24 CH12	Reserved.LCDIF not support DMA burst request.
23–22 CH11	DMA burst size for GPMI channel 7. Do not change. GPMI only support burst size 4.
21–20 CH10	DMA burst size for GPMI channel 6. Do not change. GPMI only support burst size 4.
19–18 CH9	DMA burst size for GPMI channel 5. Do not change. GPMI only support burst size 4.
17–16 CH8	DMA burst size for GPMI channel 4. Do not change. GPMI only support burst size 4.

HW_APBH_DMA_BURST_SIZE field descriptions (continued)

Field	Description
15–14 CH7	DMA burst size for GPMI channel 3. Do not change. GPMI only support burst size 4.
13–12 CH6	DMA burst size for GPMI channel 2. Do not change. GPMI only support burst size 4.
11–10 CH5	DMA burst size for GPMI channel 1. Do not change. GPMI only support burst size 4.
9–8 CH4	DMA burst size for GPMI channel 0. Do not change. GPMI only support burst size 4.
7–6 CH3	DMA burst size for SSP3. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
5–4 CH2	DMA burst size for SSP2. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
3–2 CH1	DMA burst size for SSP1. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —
1–0 CH0	DMA burst size for SSP0. 0x0 **BURST0** — 0x1 **BURST4** — 0x2 **BURST8** —

6.5.7 AHB to APBH DMA Debug Register (HW_APBH_DEBUG)

This register is for debug purpose.

It is for internal use only. Not recommend for customer usage.

Address: HW_APBH_DEBUG – 8000_4000h base + 60h offset = 8000_4060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:1]															GPMI_ONE_ FIFO
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_DEBUG field descriptions

Field	Description
31–1 RSVD	Reserved, always set to zero.
0 GPMI_ONE_ FIFO	Set to one and the eight GPMI channels will share the DMA FIFO, and when set to zero, the eight GPMI channels will use its own DMA FIFO.

6.5.8 APBH DMA Channel 0 Current Command Address Register (HW_APBH_CH0_CURCMDAR)

The APBH DMA channel 0 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 0 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR_RD(0);              // read the whole
register, since there is only one field
pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 0, CMD_ADDR);  // or, use
multi-register bitfield read macro
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR(0).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: HW_APBH_CH0_CURCMDAR – 8000_4000h base + 100h offset = 8000_4100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 0.

6.5.9 APBH DMA Channel 0 Next Command Address Register (HW_APBH_CH0_NXTCMDAR)

The APBH DMA Channel 0 Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

APBH DMA Channel 0 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
HW_APBH_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure);        // write the entire register,
 since there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure);       // or, use multi-register
 bitfield write macro
HW_APBH_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
 of indexed register's struct
```

Address: HW_APBH_CH0_NXTCMDAR – 8000_4000h base + 110h offset = 8000_4110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 0.

6.5.10 APBH DMA Channel 0 Command Register (HW_APBH_CH0_CMD)

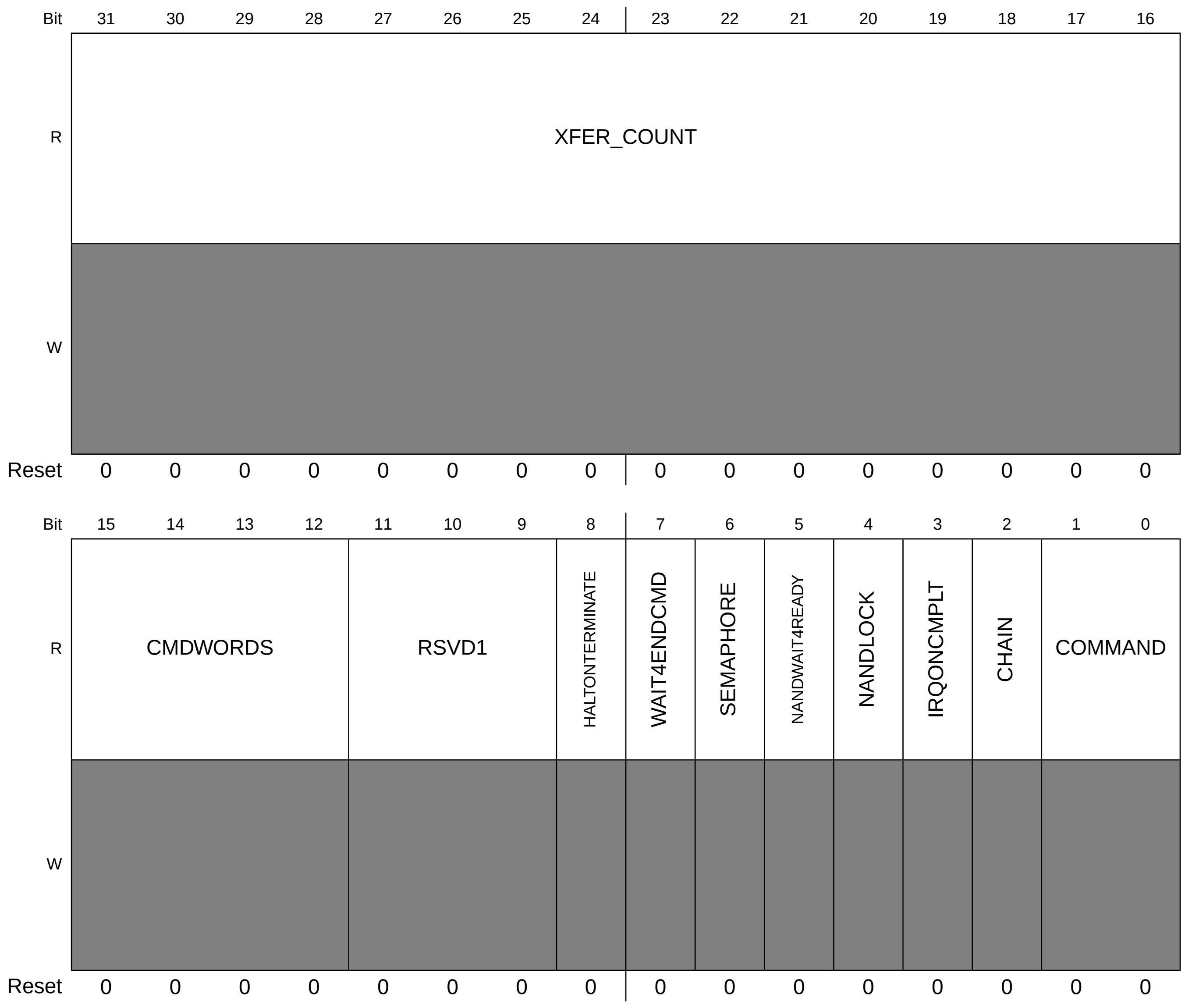
The APBH DMA Channel 0 command register specifies the DMA transaction to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE

```
hw_apbh_chn_cmd_t  dma_cmd;
dma_cmd.XFER_COUNT = 512;                                 // transfer 512 bytes
dma_cmd.COMMAND = BV_APBH_CHn_CMD_COMMAND__DMA_WRITE;  // transfer to system memory from
peripheral device
dma_cmd.CHAIN = 1;                                     // chain an additional command structure
 on to the list
dma_cmd.IRQONCMPLT = 1;                                  // generate an interrupt on completion
 of this command structure
```

Address: HW_APBH_CH0_CMD – 8000_4000h base + 120h offset = 8000_4120h



HW_APBH_CH0_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP0 device. A value of 0 indicates a 64 KBytes transfer.

HW_APBH_CH0_CMD field descriptions (continued)

Field	Description
15–12 CMDWORDS	This field indicates the number of command words to send to the SSP0, starting with the base PIO address of the SSP0 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH0_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the SSP0 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.11 APBH DMA Channel 0 Buffer Address Register (HW_APBH_CH0_BAR)

The APBH DMA Channel 0 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
hw_apbh_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: HW_APBH_CH0_BAR – 8000_4000h base + 130h offset = 8000_4130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.12 APBH DMA Channel 0 Semaphore Register (HW_APBH_CH0_SEMA)

The APBH DMA Channel 0 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Address: HW_APBH_CH0_SEMA – 8000_4000h base + 140h offset = 8000_4140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.13 AHB to APBH DMA Channel 0 Debug Information (HW_APBH_CH0_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 0 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 0.

Address: HW_APBH_CH0_DEBUG1 – 8000_4000h base + 150h offset = 8000_4150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH0_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit is reserved for this Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 0 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed.

HW_APBH_CH0_DEBUG1 field descriptions (continued)

Field	Description
	0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.14 AHB to APBH DMA Channel 0 Debug Information (HW_APBH_CH0_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

This register allows debug visibility of the APBH DMA Channel 0.

Address: HW_APBH_CH0_DEBUG2 – 8000_4000h base + 160h offset = 8000_4160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH0_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.15 APBH DMA Channel 1 Current Command Address Register (HW_APBH_CH1_CURCMDAR)

The APBH DMA Channel 1 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 1 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR_RD(1);              // read the whole
register, since there is only one field
pCurCmd = (hw_apbh_chn_cmd_t *) BF_RDn(APBH_CHn_CURCMDAR, 1, CMD_ADDR);  // or, use
multi-register bitfield read macro
pCurCmd = (hw_apbh_chn_cmd_t *) HW_APBH_CHn_CURCMDAR(1).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: HW_APBH_CH1_CURCMDAR – 8000_4000h base + 170h offset = 8000_4170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 1.

6.5.16 APBH DMA Channel 1 Next Command Address Register (HW_APBH_CH1_NXTCMDAR)

The APBH DMA Channel 1 Next Command Address register contains the address of the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to 1 in the DMA command word to process command lists.

APBH DMA Channel 1 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
HW_APBH_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure);       // write the entire register,
 since there is only one field
BF_WRn(APBH_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure);       // or, use multi-register
 bitfield write macro
HW_APBH_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
 of indexed register's struct
```

Address: HW_APBH_CH1_NXTCMDAR – 8000_4000h base + 180h offset = 8000_4180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 1.

6.5.17 APBH DMA Channel 1 Command Register (HW_APBH_CH1_CMD)

The APBH DMA Channel 1 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH1_CMD – 8000_4000h base + 190h offset = 8000_4190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP1 device. A value of 0 indicates a 64 KBytes trasnfer size.
15–12 CMDWORDS	This field indicates the number of command words to send to the SSP1, starting with the base PIO address of the SSP1 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.

HW_APBH_CH1_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the SSP1 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.18 APBH DMA Channel 1 Buffer Address Register (HW_APBH_CH1_BAR)

The APBH DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
hw_apbh_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: HW_APBH_CH1_BAR – 8000_4000h base + 1A0h offset = 8000_41A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.19 APBH DMA Channel 1 Semaphore Register (HW_APBH_CH1_SEMA)

The APBH DMA Channel 1 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 1, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 1, PHORE);  // get instantaneous value
```

Address: HW_APBH_CH1_SEMA – 8000_4000h base + 1B0h offset = 8000_41B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.20 AHB to APBH DMA Channel 1 Debug Information (HW_APBH_CH1_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 1 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 1.

Address: HW_APBH_CH1_DEBUG1 – 8000_4000h base + 1C0h offset = 8000_41C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit is reserved for this Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 1 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

HW_APBH_CH1_DEBUG1 field descriptions (continued)

Field	Description
	0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.21 AHB to APBH DMA Channel 1 Debug Information (HW_APBH_CH1_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

This register allows debug visibility of the APBH DMA Channel 1.

Address: HW_APBH_CH1_DEBUG2 – 8000_4000h base + 1D0h offset = 8000_41D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH1_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.22 APBH DMA Channel 2 Current Command Address Register (HW_APBH_CH2_CURCMDAR)

The APBH DMA Channel 2 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 2 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH2_CURCMDAR – 8000_4000h base + 1E0h offset = 8000_41E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 2.

6.5.23 APBH DMA Channel 2 Next Command Address Register (HW_APBH_CH2_NXTCMDAR)

The APBH DMA Channel 2 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 2 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH2_NXTCMDAR – 8000_4000h base + 1F0h offset = 8000_41F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 2.

6.5.24 APBH DMA Channel 2 Command Register (HW_APBH_CH2_CMD)

The APBH DMA Channel 2 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH2_CMD – 8000_4000h base + 200h offset = 8000_4200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨			▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP2 device. A value of 0 indicates a 64 KBytes trasnfer size.
15–12 CMDWORDS	This field indicates the number of command words to send to the SSP2, starting with the base PIO address of the SSP2 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.

HW_APBH_CH2_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the SSP2 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.25 APBH DMA Channel 2 Buffer Address Register (HW_APBH_CH2_BAR)

The APBH DMA Channel 2 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH2_BAR – 8000_4000h base + 210h offset = 8000_4210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.26 APBH DMA Channel 2 Semaphore Register (HW_APBH_CH2_SEMA)

The APBH DMA Channel 2 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH2_SEMA – 8000_4000h base + 220h offset = 8000_4220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.27 AHB to APBH DMA Channel 2 Debug Information (HW_APBH_CH2_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 2 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 2.

Address: HW_APBH_CH2_DEBUG1 – 8000_4000h base + 230h offset = 8000_4230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH2_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit is reserved for this Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 2 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed.

HW_APBH_CH2_DEBUG1 field descriptions (continued)

Field	Description
	0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.28 AHB to APBH DMA Channel 2 Debug Information (HW_APBH_CH2_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

This register allows debug visibility of the APBH DMA Channel 2.

Address: HW_APBH_CH2_DEBUG2 – 8000_4000h base + 240h offset = 8000_4240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH2_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.29 APBH DMA Channel 3 Current Command Address Register (HW_APBH_CH3_CURCMDAR)

The APBH DMA Channel 3 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 3 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH3_CURCMDAR – 8000_4000h base + 250h offset = 8000_4250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 3.

6.5.30 APBH DMA Channel 3 Next Command Address Register (HW_APBH_CH3_NXTCMDAR)

The APBH DMA Channel 3 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 3 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 3 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH3_NXTCMDAR – 8000_4000h base + 260h offset = 8000_4260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 3.

6.5.31 APBH DMA Channel 3 Command Register (HW_APBH_CH3_CMD)

The APBH DMA Channel 3 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH3_CMD – 8000_4000h base + 270h offset = 8000_4270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨			▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SSP3 device. A value of 0 indicates a 64 KBytes trasnfer size.
15–12 CMDWORDS	This field indicates the number of command words to send to the SSP3, starting with the base PIO address of the SSP3 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the SSP3 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.32 APBH DMA Channel 3 Buffer Address Register (HW_APBH_CH3_BAR)

The APBH DMA Channel 3 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH3_BAR – 8000_4000h base + 280h offset = 8000_4280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.33 APBH DMA Channel 3 Semaphore Register (HW_APBH_CH3_SEMA)

The APBH DMA Channel 3 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH3_SEMA – 8000_4000h base + 290h offset = 8000_4290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.34 AHB to APBH DMA Channel 3 Debug Information (HW_APBH_CH3_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 3 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 3.

Address: HW_APBH_CH3_DEBUG1 – 8000_4000h base + 2A0h offset = 8000_42A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit is reserved for this DMA Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit is reserved for this Channel and always reads 0. For Channels 4-11, this bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflect the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflect the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflect the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflect the current state of the DMA Channel's Write FIFO Full signal.

HW_APBH_CH3_DEBUG1 field descriptions (continued)

Field	Description
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 3 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.35 AHB to APBH DMA Channel 3 Debug Information (HW_APBH_CH3_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

This register allows debug visibility of the APBH DMA Channel 3.

Address: HW_APBH_CH3_DEBUG2 – 8000_4000h base + 2B0h offset = 8000_42B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH3_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.36 APBH DMA Channel 4 Current Command Address Register (HW_APBH_CH4_CURCMDAR)

The APBH DMA Channel 4 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 4 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH4_CURCMDAR – 8000_4000h base + 2C0h offset = 8000_42C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 4.

6.5.37 APBH DMA Channel 4 Next Command Address Register (HW_APBH_CH4_NXTCMDAR)

The APBH DMA Channel 4 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 4 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH4_NXTCMDAR – 8000_4000h base + 2D0h offset = 8000_42D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 4.

6.5.38 APBH DMA Channel 4 Command Register (HW_APBH_CH4_CMD)

The APBH DMA Channel 4 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH4_CMD – 8000_4000h base + 2E0h offset = 8000_42E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨			▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI0 device. A value of 0 indicates a 64 KBytes trasnfer size.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI0, starting with the base PIO address of the GPMI0 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

HW_APBH_CH4_CMD field descriptions (continued)

Field	Description
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH1_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the NAND0(APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.39 APBH DMA Channel 4 Buffer Address Register (HW_APBH_CH4_BAR)

The APBH DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH4_BAR – 8000_4000h base + 2F0h offset = 8000_42F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.40 APBH DMA Channel 4 Semaphore Register (HW_APBH_CH4_SEMA)

The APBH DMA Channel 4 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBH_CHn_SEMA, 0, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBH_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Address: HW_APBH_CH4_SEMA – 8000_4000h base + 300h offset = 8000_4300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.

HW_APBH_CH4_SEMA field descriptions (continued)

Field	Description
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.41 AHB to APBH DMA Channel 4 Debug Information (HW_APBH_CH4_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 4 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 4.

Address: HW_APBH_CH4_DEBUG1 – 8000_4000h base + 310h offset = 8000_4310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved

HW_APBH_CH4_DEBUG1 field descriptions (continued)

Field	Description
4–0 STATEMACHINE	PIO Display of the DMA Channel 4 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.42 AHB to APBH DMA Channel 4 Debug Information (HW_APBH_CH4_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

This register allows debug visibility of the APBH DMA Channel 4.

Address: HW_APBH_CH4_DEBUG2 – 8000_4000h base + 320h offset = 8000_4320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH4_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.43 APBH DMA Channel 5 Current Command Address Register (HW_APBH_CH5_CURCMDAR)

The APBH DMA Channel 5 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 5 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH5_CURCMDAR – 8000_4000h base + 330h offset = 8000_4330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 5.

6.5.44 APBH DMA Channel 5 Next Command Address Register (HW_APBH_CH5_NXTCMDAR)

The APBH DMA Channel 5 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 5 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH5_NXTCMDAR – 8000_4000h base + 340h offset = 8000_4340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 5.

6.5.45 APBH DMA Channel 5 Command Register (HW_APBH_CH5_CMD)

The APBH DMA Channel 5 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH5_CMD – 8000_4000h base + 350h offset = 8000_4350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI1 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI1, starting with the base PIO address of the GPMI1 control register and incrementing from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

HW_APBH_CH5_CMD field descriptions (continued)

Field	Description
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH0_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from the NAND1 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.46 APBH DMA Channel 5 Buffer Address Register (HW_APBH_CH5_BAR)

The APBH DMA Channel 5 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH5_BAR – 8000_4000h base + 360h offset = 8000_4360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.47 APBH DMA Channel 5 Semaphore Register (HW_APBH_CH5_SEMA)

The APBH DMA Channel 5 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH5_SEMA – 8000_4000h base + 370h offset = 8000_4370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.48 AHB to APBH DMA Channel 5 Debug Information (HW_APBH_CH5_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 5 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 5.

Address: HW_APBH_CH5_DEBUG1 – 8000_4000h base + 380h offset = 8000_4380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 5 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

HW_APBH_CH5_DEBUG1 field descriptions (continued)

Field	Description
	0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.49 AHB to APBH DMA Channel 5 Debug Information (HW_APBH_CH5_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

This register allows debug visibility of the APBH DMA Channel 5.

Address: HW_APBH_CH5_DEBUG2 – 8000_4000h base + 390h offset = 8000_4390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH5_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.50 APBH DMA Channel 6 Current Command Address Register (HW_APBH_CH6_CURCMDAR)

The APBH DMA Channel 6 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 6 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH6_CURCMDAR – 8000_4000h base + 3A0h offset = 8000_43A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 6.

6.5.51 APBH DMA Channel 6 Next Command Address Register (HW_APBH_CH6_NXTCMDAR)

The APBH DMA Channel 6 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 6 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH6_NXTCMDAR – 8000_4000h base + 3B0h offset = 8000_43B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 6.

6.5.52 APBH DMA Channel 6 Command Register (HW_APBH_CH6_CMD)

The APBH DMA Channel 6 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH6_CMD – 8000_4000h base + 3C0h offset = 8000_43C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨			▨	▨	▨	▨		▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI2 device. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI2 starting with the base PIO address of the GPMI2 and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, that is, after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH3_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, that is, data sent from NAND2(APB PIO Read) to the system memory (AHB master write).

HW_APBH_CH6_CMD field descriptions (continued)

Field	Description
	10- read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.53 APBH DMA Channel 6 Buffer Address Register (HW_APBH_CH6_BAR)

The APBH DMA Channel 6 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH6_BAR – 8000_4000h base + 3D0h offset = 8000_43D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.54 APBH DMA Channel 6 Semaphore Register (HW_APBH_CH6_SEMA)

The APBH DMA Channel 6 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH6_SEMA – 8000_4000h base + 3E0h offset = 8000_43E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.55 AHB to APBH DMA Channel 6 Debug Information (HW_APBH_CH6_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 6 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 6.

Address: HW_APBH_CH6_DEBUG1 – 8000_4000h base + 3F0h offset = 8000_43F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH6_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 6 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete.

HW_APBH_CH6_DEBUG1 field descriptions (continued)

Field	Description
	0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.56 AHB to APBH DMA Channel 6 Debug Information (HW_APBH_CH6_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

This register allows debug visibility of the APBH DMA Channel 6.

Address: HW_APBH_CH6_DEBUG2 – 8000_4000h base + 400h offset = 8000_4400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH6_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.57 APBH DMA Channel 7 Current Command Address Register (HW_APBH_CH7_CURCMDAR)

The APBH DMA Channel 7 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 7 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH7_CURCMDAR – 8000_4000h base + 410h offset = 8000_4410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 7.

6.5.58 APBH DMA Channel 7 Next Command Address Register (HW_APBH_CH7_NXTCMDAR)

The APBH DMA Channel 7 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 7 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH7_NXTCMDAR – 8000_4000h base + 420h offset = 8000_4420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_NXTCMDAR field descriptions

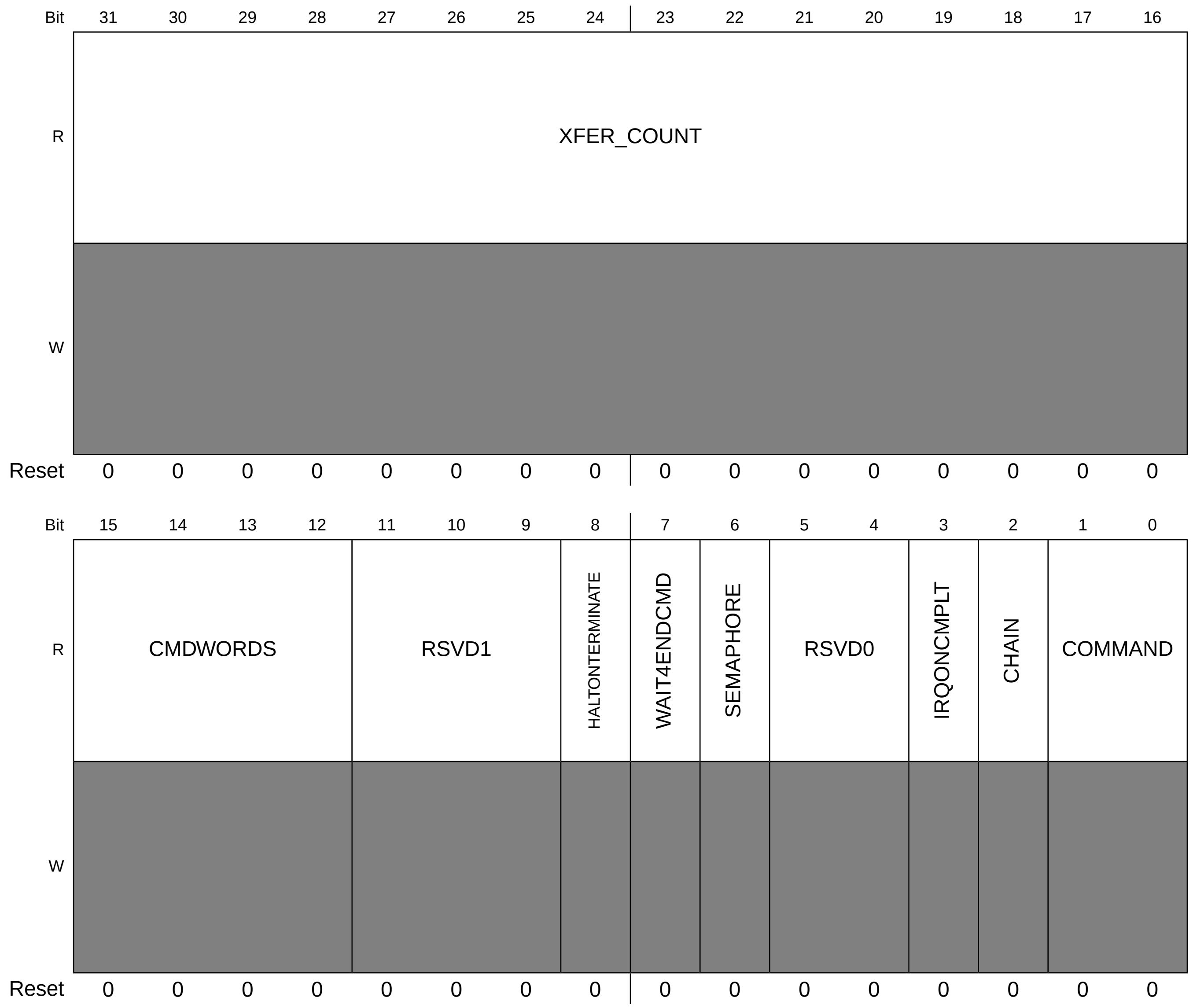
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 7.

6.5.59 APBH DMA Channel 7 Command Register (HW_APBH_CH7_CMD)

The APBH DMA Channel 7 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH7_CMD – 8000_4000h base + 430h offset = 8000_4430h



HW_APBH_CH7_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI3 device. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI3, starting with the base PIO address of the GPMI3 and increment from there. Zero means transfer NO command words

HW_APBH_CH7_CMD field descriptions (continued)

Field	Description
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH3_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, that is, data sent from NAND3 (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.60 APBH DMA Channel 7 Buffer Address Register (HW_APBH_CH7_BAR)

The APBH DMA Channel 7 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH7_BAR – 8000_4000h base + 440h offset = 8000_4440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.61 APBH DMA Channel 7 Semaphore Register (HW_APBH_CH7_SEMA)

The APBH DMA Channel 7 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH7_SEMA – 8000_4000h base + 450h offset = 8000_4450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.

HW_APBH_CH7_SEMA field descriptions (continued)

Field	Description
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.62 AHB to APBH DMA Channel 7 Debug Information (HW_APBH_CH7_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 7 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 7.

Address: HW_APBH_CH7_DEBUG1 – 8000_4000h base + 460h offset = 8000_4460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved

HW_APBH_CH7_DEBUG1 field descriptions (continued)

Field	Description
4–0 STATEMACHINE	PIO Display of the DMA Channel 7 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.63 AHB to APBH DMA Channel 7 Debug Information (HW_APBH_CH7_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

This register allows debug visibility of the APBH DMA Channel 7.

Address: HW_APBH_CH7_DEBUG2 – 8000_4000h base + 470h offset = 8000_4470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH7_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.64 APBH DMA Channel 8 Current Command Address Register (HW_APBH_CH8_CURCMDAR)

The APBH DMA Channel 8 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 8 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH8_CURCMDAR – 8000_4000h base + 480h offset = 8000_4480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 8.

6.5.65 APBH DMA Channel 8 Next Command Address Register (HW_APBH_CH8_NXTCMDAR)

The APBH DMA Channel 8 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 8 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 8 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH8_NXTCMDAR – 8000_4000h base + 490h offset = 8000_4490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_NXTCMDAR field descriptions

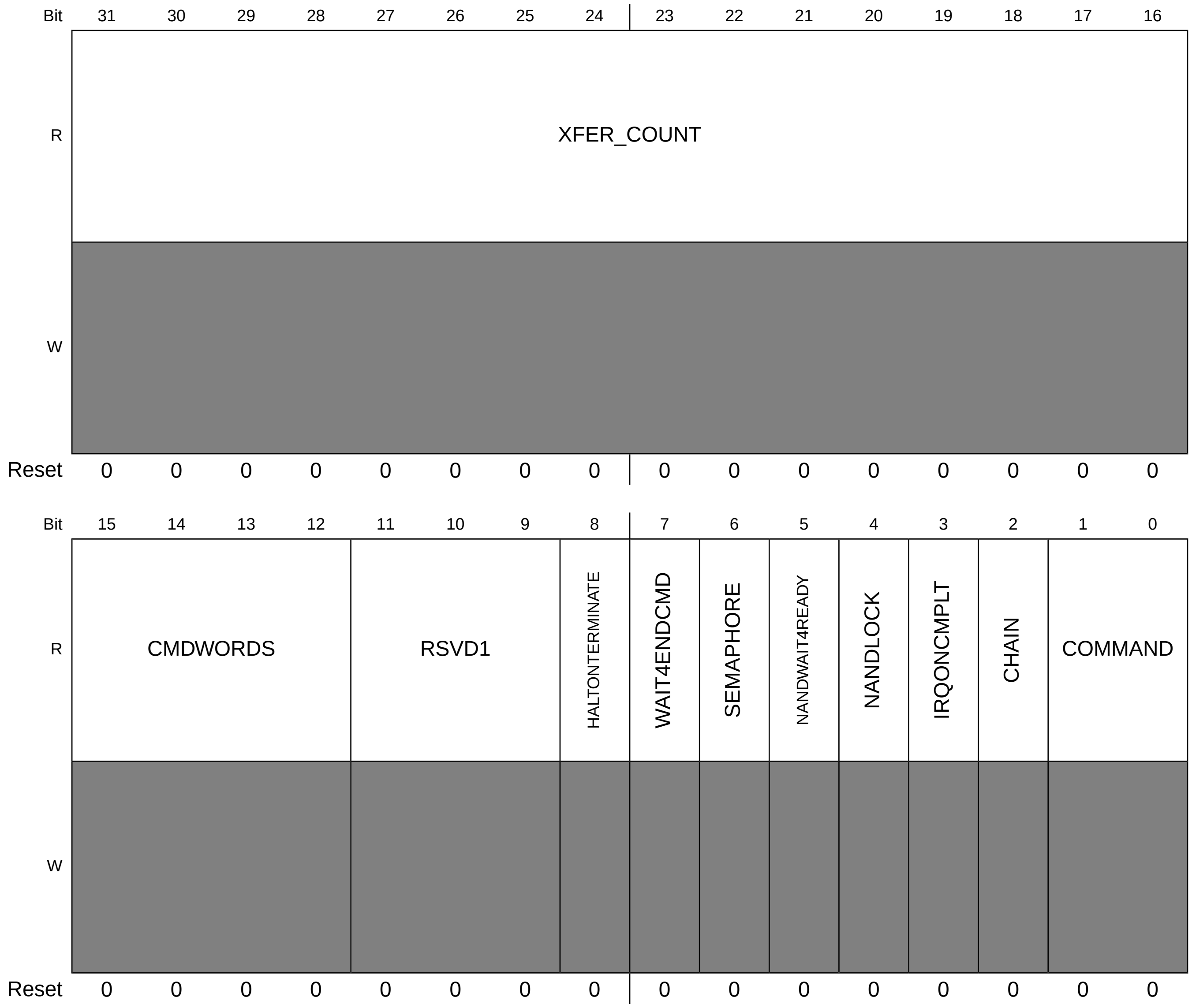
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 8.

6.5.66 APBH DMA Channel 8 Command Register (HW_APBH_CH8_CMD)

The APBH DMA Channel 8 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH8_CMD – 8000_4000h base + 4A0h offset = 8000_44A0h



HW_APBH_CH8_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI4 device HW_GPMI_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI4, starting with the base PIO address of the GPMI4 (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

HW_APBH_CH8_CMD field descriptions (continued)

Field	Description
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH7_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from NAND4 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.67 APBH DMA Channel 8 Buffer Address Register (HW_APBH_CH8_BAR)

The APBH DMA Channel 8 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH8_BAR – 8000_4000h base + 4B0h offset = 8000_44B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.68 APBH DMA Channel 8 Semaphore Register (HW_APBH_CH8_SEMA)

The APBH DMA Channel 8 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH8_SEMA – 8000_4000h base + 4C0h offset = 8000_44C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.69 AHB to APBH DMA Channel 8 Debug Information (HW_APBH_CH8_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 8 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 7.

Address: HW_APBH_CH8_DEBUG1 – 8000_4000h base + 4D0h offset = 8000_44D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 7 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

HW_APBH_CH8_DEBUG1 field descriptions (continued)

Field	Description
	0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.70 AHB to APBH DMA Channel 8 Debug Information (HW_APBH_CH8_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 8.

This register allows debug visibility of the APBH DMA Channel 8.

Address: HW_APBH_CH8_DEBUG2 – 8000_4000h base + 4E0h offset = 8000_44E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH8_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.71 APBH DMA Channel 9 Current Command Address Register (HW_APBH_CH9_CURCMDAR)

The APBH DMA Channel 9 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 9 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH9_CURCMDAR – 8000_4000h base + 4F0h offset = 8000_44F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 9.

6.5.72 APBH DMA Channel 9 Next Command Address Register (HW_APBH_CH9_NXTCMDAR)

The APBH DMA Channel 9 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 9 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 9 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH9_NXTCMDAR – 8000_4000h base + 500h offset = 8000_4500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 9.

6.5.73 APBH DMA Channel 9 Command Register (HW_APBH_CH9_CMD)

The APBH DMA Channel 9 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH9_CMD – 8000_4000h base + 510h offset = 8000_4510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI5 device HW_GPMI_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI5, starting with the base PIO address of the GPMI5 (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH9_CMDAR to find the next command.

HW_APBH_CH9_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from NAND5 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.74 APBH DMA Channel 9 Buffer Address Register (HW_APBH_CH9_BAR)

The APBH DMA Channel 9 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH9_BAR – 8000_4000h base + 520h offset = 8000_4520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.75 APBH DMA Channel 9 Semaphore Register (HW_APBH_CH9_SEMA)

The APBH DMA Channel 9 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH9_SEMA – 8000_4000h base + 530h offset = 8000_4530h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.76 AHB to APBH DMA Channel 9 Debug Information (HW_APBH_CH9_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 9 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 9.

Address: HW_APBH_CH9_DEBUG1 – 8000_4000h base + 540h offset = 8000_4540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH9_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 9 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete.

HW_APBH_CH9_DEBUG1 field descriptions (continued)

Field	Description
	0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.77 AHB to APBH DMA Channel 9 Debug Information (HW_APBH_CH9_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 9.

This register allows debug visibility of the APBH DMA Channel 9.

Address: HW_APBH_CH9_DEBUG2 – 8000_4000h base + 550h offset = 8000_4550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH9_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.78 APBH DMA channel 10 Current Command Address Register (HW_APBH_CH10_CURCMDAR)

The APBH DMA channel 10 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA channel 10 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH10_CURCMDAR – 8000_4000h base + 560h offset = 8000_4560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 10.

6.5.79 APBH DMA channel 10 Next Command Address Register (HW_APBH_CH10_NXTCMDAR)

The APBH DMA channel 10 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA channel 10 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the channel 10 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH10_NXTCMDAR – 8000_4000h base + 570h offset = 8000_4570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 10.

6.5.80 APBH DMA channel 10 Command Register (HW_APBH_CH10_CMD)

The APBH DMA channel 10 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH10_CMD – 8000_4000h base + 580h offset = 8000_4580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI6 device HW_GPMI_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI6, starting with the base PIO address of the GPMI6 (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words

HW_APBH_CH10_CMD field descriptions (continued)

Field	Description
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH10_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from NAND6 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.81 APBH DMA channel 10 Buffer Address Register (HW_APBH_CH10_BAR)

The APBH DMA channel 10 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH10_BAR – 8000_4000h base + 590h offset = 8000_4590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.82 APBH DMA channel 10 Semaphore Register (HW_APBH_CH10_SEMA)

The APBH DMA channel 10 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH10_SEMA – 8000_4000h base + 5A0h offset = 8000_45A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

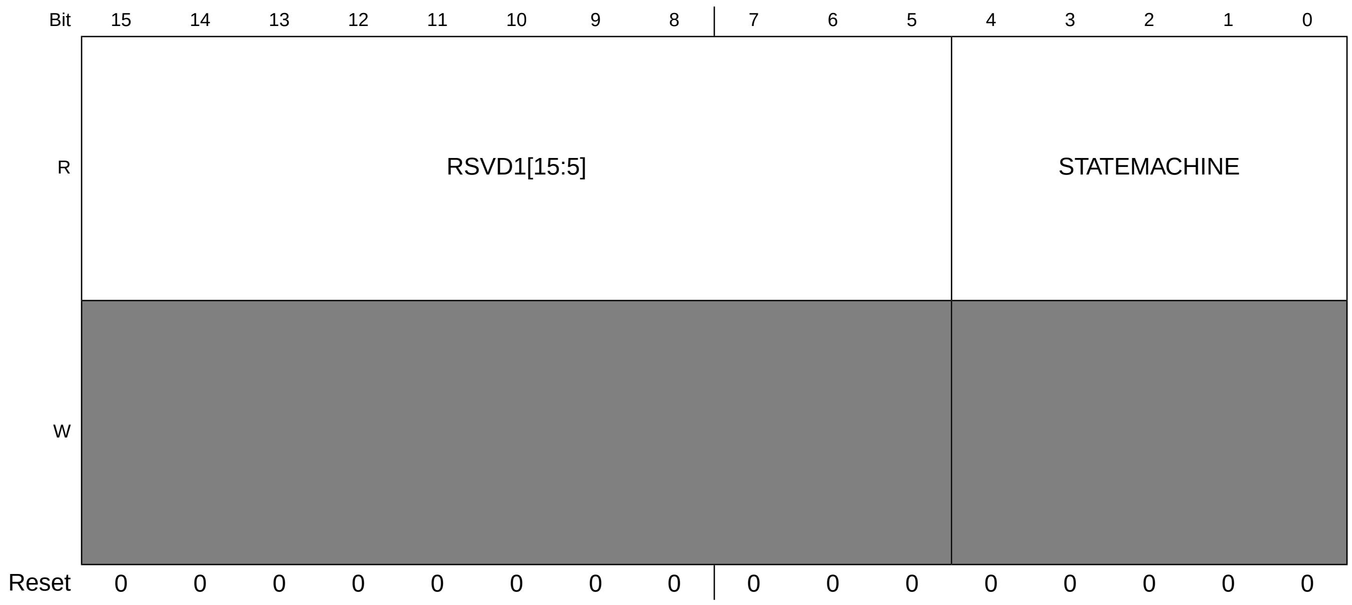
6.5.83 AHB to APBH DMA channel 10 Debug Information (HW_APBH_CH10_DEBUG1)

This register gives debug visibility into the APBH DMA channel 10 state machine and controls.

This register allows debug visibility of the APBH DMA channel 10.

Address: HW_APBH_CH10_DEBUG1 – 8000_4000h base + 5B0h offset = 8000_45B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0



HW_APBH_CH10_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved

HW_APBH_CH10_DEBUG1 field descriptions (continued)

Field	Description
4–0 STATEMACHINE	PIO Display of the DMA channel 10 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.84 AHB to APBH DMA channel 10 Debug Information (HW_APBH_CH10_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA channel 10.

This register allows debug visibility of the APBH DMA channel 10.

Address: HW_APBH_CH10_DEBUG2 – 8000_4000h base + 5C0h offset = 8000_45C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH10_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.85 APBH DMA Channel 11 Current Command Address Register (HW_APBH_CH11_CURCMDAR)

The APBH DMA Channel 11 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 11 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH11_CURCMDAR – 8000_4000h base + 5D0h offset = 8000_45D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 11.

6.5.86 APBH DMA Channel 11 Next Command Address Register (HW_APBH_CH11_NXTCMDAR)

The APBH DMA Channel 11 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 11 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 11 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH11_NXTCMDAR – 8000_4000h base + 5E0h offset = 8000_45E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_NXTCMDAR field descriptions

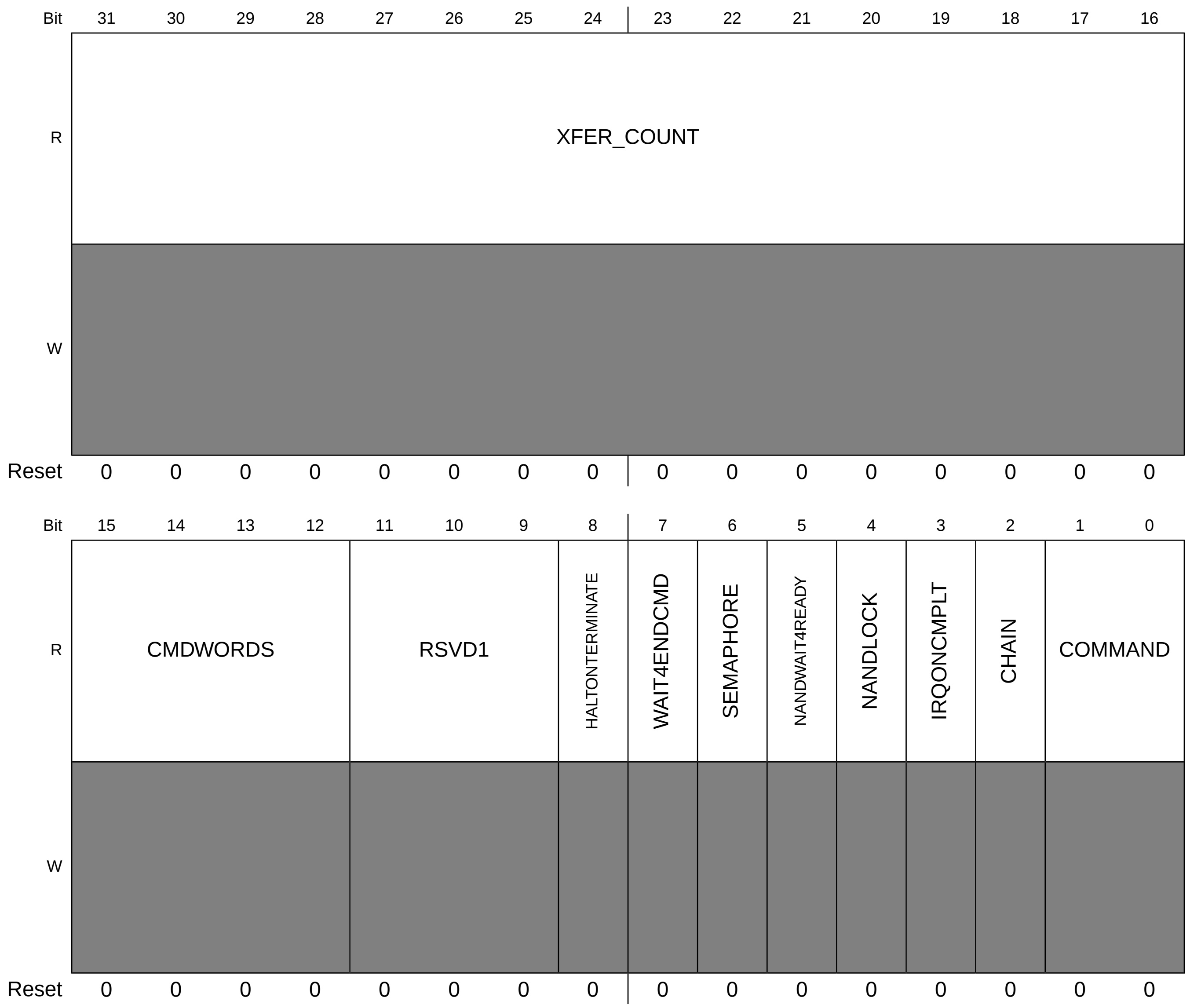
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 11.

6.5.87 APBH DMA Channel 11 Command Register (HW_APBH_CH11_CMD)

The APBH DMA Channel 11 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH11_CMD – 8000_4000h base + 5F0h offset = 8000_45F0h



HW_APBH_CH11_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the GPMI7 device HW_GPMI_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the GPMI7, starting with the base PIO address of the GPMI7 (HW_GPMI_CTRL0) and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.

HW_APBH_CH11_CMD field descriptions (continued)

Field	Description
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH11_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from NAND7 (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.88 APBH DMA Channel 11 Buffer Address Register (HW_APBH_CH11_BAR)

The APBH DMA Channel 11 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH11_BAR – 8000_4000h base + 600h offset = 8000_4600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.89 APBH DMA Channel 11 Semaphore Register (HW_APBH_CH11_SEMA)

The APBH DMA Channel 11 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH11_SEMA – 8000_4000h base + 610h offset = 8000_4610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.90 AHB to APBH DMA Channel 11 Debug Information (HW_APBH_CH11_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 11 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 11.

Address: HW_APBH_CH11_DEBUG1 – 8000_4000h base + 620h offset = 8000_4620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 11 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel.

HW_APBH_CH11_DEBUG1 field descriptions (continued)

Field	Description
	0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.91 AHB to APBH DMA Channel 11 Debug Information (HW_APBH_CH11_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 11.

This register allows debug visibility of the APBH DMA Channel 11.

Address: HW_APBH_CH11_DEBUG2 – 8000_4000h base + 630h offset = 8000_4630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH11_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.92 APBH DMA channel 12 Current Command Address Register (HW_APBH_CH12_CURCMDAR)

The APBH DMA channel 12 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA channel 12 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH12_CURCMDAR – 8000_4000h base + 640h offset = 8000_4640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 12.

6.5.93 APBH DMA channel 12 Next Command Address Register (HW_APBH_CH12_NXTCMDAR)

The APBH DMA channel 12 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA channel 12 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the channel 12 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH12_NXTCMDAR – 8000_4000h base + 650h offset = 8000_4650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 12.

6.5.94 APBH DMA channel 12 Command Register (HW_APBH_CH12_CMD)

The APBH DMA channel 12 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH12_CMD – 8000_4000h base + 660h offset = 8000_4660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨			▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the HSADC device HW_HSADC_FIFO_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the HSADC, starting with the base PIO address of the HSADC (HW_HSADC_CTRL0) and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels..
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH12_CMDAR to find the next command.

HW_APBH_CH12_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from HSADC (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.95 APBH DMA channel 12 Buffer Address Register (HW_APBH_CH12_BAR)

The APBH DMA channel 12 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH12_BAR – 8000_4000h base + 670h offset = 8000_4670h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.96 APBH DMA channel 12 Semaphore Register (HW_APBH_CH12_SEMA)

The APBH DMA channel 12 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH12_SEMA – 8000_4000h base + 680h offset = 8000_4680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.97 AHB to APBH DMA channel 12 Debug Information (HW_APBH_CH12_DEBUG1)

This register gives debug visibility into the APBH DMA channel 12 state machine and controls.

This register allows debug visibility of the APBH DMA channel 12.

Address: HW_APBH_CH12_DEBUG1 – 8000_4000h base + 690h offset = 8000_4690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH12_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA channel 12 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete.

HW_APBH_CH12_DEBUG1 field descriptions (continued)

Field	Description
	0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.98 AHB to APBH DMA channel 12 Debug Information (HW_APBH_CH12_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA channel 12.

This register allows debug visibility of the APBH DMA channel 12.

Address: HW_APBH_CH12_DEBUG2 – 8000_4000h base + 6A0h offset = 8000_46A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH12_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.99 APBH DMA Channel 13 Current Command Address Register (HW_APBH_CH13_CURCMDAR)

The APBH DMA Channel 13 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBH DMA Channel 13 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBH_CH13_CURCMDAR – 8000_4000h base + 6B0h offset = 8000_46B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 13.

6.5.100 APBH DMA Channel 13 Next Command Address Register (HW_APBH_CH13_NXTCMDAR)

The APBH DMA Channel 13 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBH DMA Channel 13 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 13 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBH_CH13_NXTCMDAR – 8000_4000h base + 6C0h offset = 8000_46C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_NXTCMDAR field descriptions

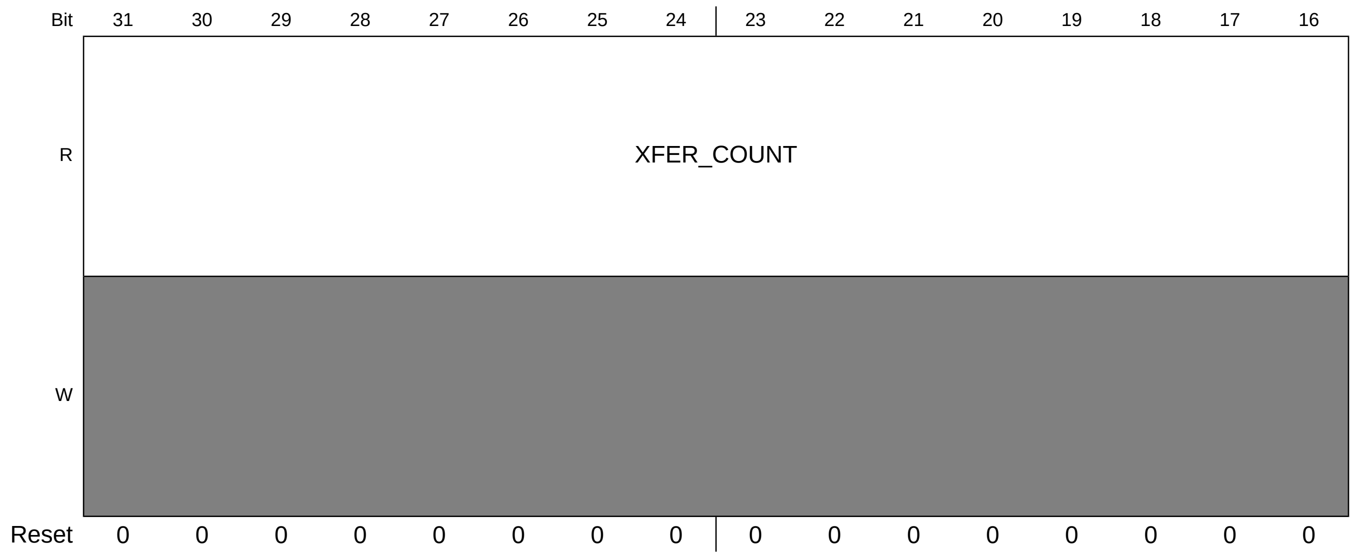
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 13.

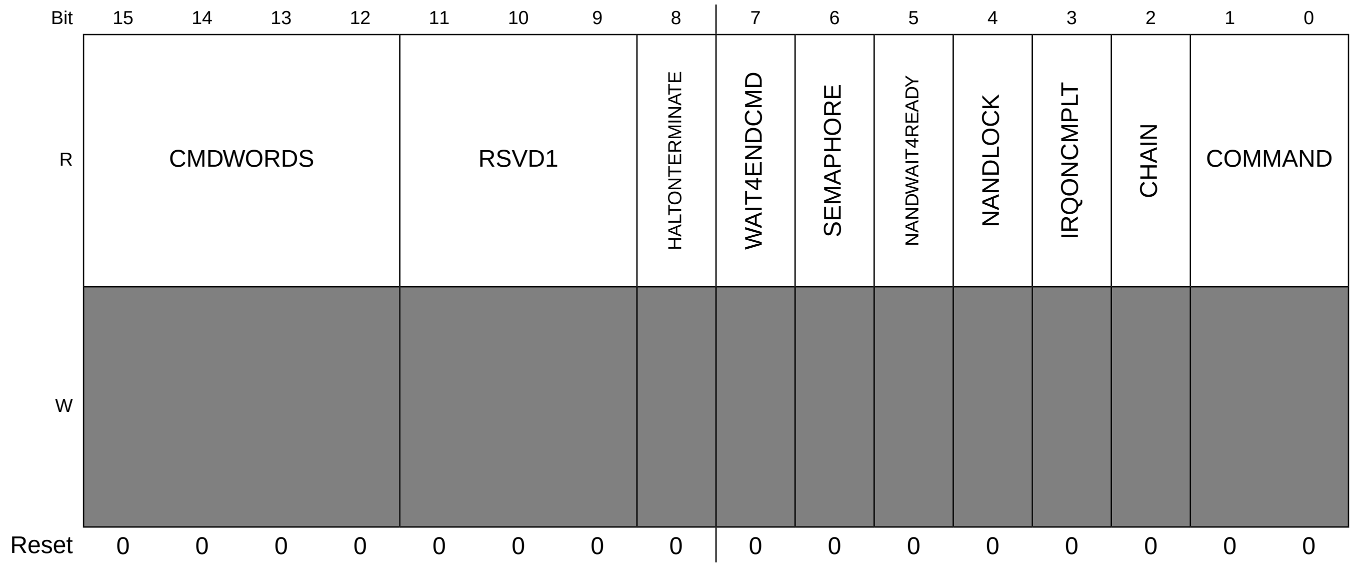
6.5.101 APBH DMA Channel 13 Command Register (HW_APBH_CH13_CMD)

The APBH DMA Channel 13 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBH_CH13_CMD – 8000_4000h base + 6D0h offset = 8000_46D0h





HW_APBH_CH13_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the LCDIF device HW_LCDIF_DATA register. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the LCDIF, starting with the base PIO address of the LCDIF (HW_LCDIF_CTRL) and increment from there. Zero means transfer NO command words

HW_APBH_CH13_CMD field descriptions (continued)

Field	Description
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH13_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from LCDIF (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.102 APBH DMA Channel 13 Buffer Address Register (HW_APBH_CH13_BAR)

The APBH DMA Channel 13 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBH_CH13_BAR – 8000_4000h base + 6E0h offset = 8000_46E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.103 APBH DMA Channel 13 Semaphore Register (HW_APBH_CH13_SEMA)

The APBH DMA Channel 13 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBH_CH13_SEMA – 8000_4000h base + 6F0h offset = 8000_46F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.104 AHB to APBH DMA Channel 13 Debug Information (HW_APBH_CH13_DEBUG1)

This register gives debug visibility into the APBH DMA Channel 13 state machine and controls.

This register allows debug visibility of the APBH DMA Channel 13.

Address: HW_APBH_CH13_DEBUG1 – 8000_4000h base + 700h offset = 8000_4700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of the GPMI Sense Signal sent from the APB GPMI Device
26 READY	This bit reflects the current state of the GPMI Ready Signal sent from the APB GPMI Device
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved

HW_APBH_CH13_DEBUG1 field descriptions (continued)

Field	Description
4–0 STATEMACHINE	PIO Display of the DMA Channel 13 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.105 AHB to APBH DMA Channel 13 Debug Information (HW_APBH_CH13_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 13.

This register allows debug visibility of the APBH DMA Channel 13.

Address: HW_APBH_CH13_DEBUG2 – 8000_4000h base + 710h offset = 8000_4710h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH13_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.106 APBH DMA channel 14 Current Command Address Register (HW_APBH_CH14_CURCMDAR)

The APBH DMA channel 14 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_CURCMDAR – 8000_4000h base + 720h offset = 8000_4720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 14.

6.5.107 APBH DMA channel 14 Next Command Address Register (HW_APBH_CH14_NXTCMDAR)

The APBH DMA channel 14 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_NXTCMDAR – 8000_4000h base + 730h offset = 8000_4730h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 14.

6.5.108 APBH DMA channel 14 Command Register (HW_APBH_CH14_CMD)

The APBH DMA channel 14 command register specifies the cycle to perform for the current command chain item.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_CMD – 8000_4000h base + 740h offset = 8000_4740h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in assigned peripheral. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to assigned peripheral, starting with the base PIO address of assigned peripheral and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH14_CMDAR to find the next command.

HW_APBH_CH14_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, that is, data sent from assigned peripheral (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.109 APBH DMA channel 14 Buffer Address Register (HW_APBH_CH14_BAR)

The APBH DMA channel 14 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_BAR – 8000_4000h base + 750h offset = 8000_4750h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.110 APBH DMA channel 14 Semaphore Register (HW_APBH_CH14_SEMA)

The APBH DMA channel 14 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_SEMA – 8000_4000h base + 760h offset = 8000_4760h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.111 AHB to APBH DMA channel 14 Debug Information (HW_APBH_CH14_DEBUG1)

This register gives debug visibility into the APBH DMA channel 14 state machine and controls.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_DEBUG1 – 8000_4000h base + 770h offset = 8000_4770h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBH_CH14_DEBUG1 field descriptions (continued)

Field	Description
27 SENSE	This bit reflects the current state of the Sense Signal sent from assigned peripheral
26 READY	This bit reflects the current state of the Ready Signal sent from assigned peripheral
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA channel 14 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete.

HW_APBH_CH14_DEBUG1 field descriptions (continued)

Field	Description
	0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.112 AHB to APBH DMA channel 14 Debug Information (HW_APBH_CH14_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA channel 14.

Because channel 14 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH14_DEBUG2 – 8000_4000h base + 780h offset = 8000_4780h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH14_DEBUG2 field descriptions

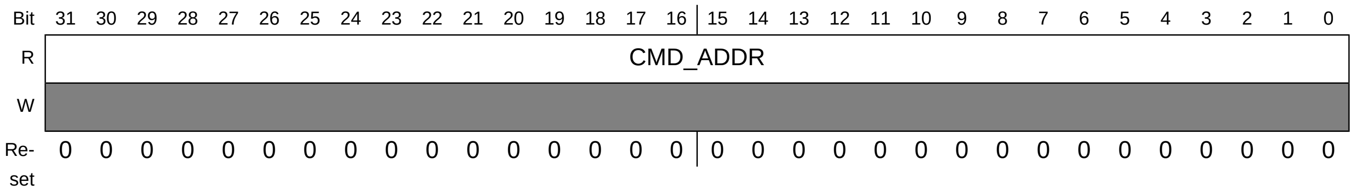
Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.113 APBH DMA channel 15 Current Command Address Register (HW_APBH_CH15_CURCMDAR)

The APBH DMA channel 15 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_CURCMDAR – 8000_4000h base + 790h offset = 8000_4790h



HW_APBH_CH15_CURCMDAR field descriptions

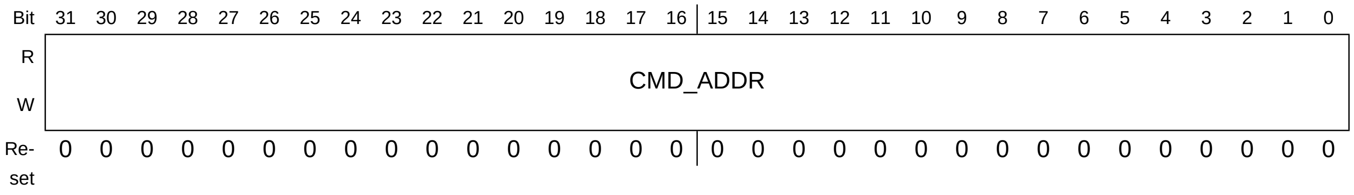
Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 15.

6.5.114 APBH DMA channel 15 Next Command Address Register (HW_APBH_CH15_NXTCMDAR)

The APBH DMA channel 15 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_NXTCMDAR – 8000_4000h base + 7A0h offset = 8000_47A0h



HW_APBH_CH15_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 15.

6.5.115 APBH DMA channel 15 Command Register (HW_APBH_CH15_CMD)

The APBH DMA channel 15 command register specifies the cycle to perform for the current command chain item.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_CMD – 8000_4000h base + 7B0h offset = 8000_47B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1			HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	NANDWAIT4READY	NANDLOCK	IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH15_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in assigned peripheral. A value of 0 indicates a 64 Kbytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to assigned peripheral, starting with the base PIO address of assigned peripheral and increment from there. Zero means transfer NO command words
11–9 RSVD1	Reserved, always set to zero.
8 HALTONTERMINATE	A value of one indicates that the channel immediately terminates the current descriptor and halts the DMA channel if a terminate signal is set. A value of 0 still causes an immediate terminate of the channel if the terminate signal is set, but the channel continues as if the count had been exhausted, meaning it honors IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel waits for the end of command signal to be sent from the APBH device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel decrements its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5 NANDWAIT4READY	A value of one indicates that the NAND DMA channel waits until the NAND device reports \'ready\' before executing the command. It is ignored for non-NAND DMA channels.
4 NANDLOCK	A value of one indicates that the NAND DMA channel remains locked in the arbiter at the expense of other NAND DMA channels. It is ignored for non-NAND DMA channels.
3 IRQONCMPLT	A value of one indicates that the channel will cause the interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBH_CH15_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- Write transfers, i.e. data sent from assigned peripheral (APB PIO Read) to the system memory (AHB master write). 10- Read transfer 11- SENSE 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes. 0x3 **DMA_SENSE** — Perform any requested PIO word transfers and then perform a conditional branch to the next chained device. Follow the NEXCMD_ADDR pointer if the perpheral sense is false. Follow the BUFFER_ADDRESS as a chain pointer if the peripheral sense line is true.

6.5.116 APBH DMA channel 15 Buffer Address Register (HW_APBH_CH15_BAR)

The APBH DMA channel 15 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_BAR – 8000_4000h base + 7C0h offset = 8000_47C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH15_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

6.5.117 APBH DMA channel 15 Semaphore Register (HW_APBH_CH15_SEMA)

The APBH DMA channel 15 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_SEMA – 8000_4000h base + 7D0h offset = 8000_47D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH15_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.

HW_APBH_CH15_SEMA field descriptions (continued)

Field	Description
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

6.5.118 AHB to APBH DMA channel 15 Debug Information (HW_APBH_CH15_DEBUG1)

This register gives debug visibility into the APBH DMA channel 15 state machine and controls.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_DEBUG1 – 8000_4000h base + 7E0h offset = 8000_47E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	SENSE	READY	LOCK	NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH15_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27 SENSE	This bit reflects the current state of Sense Signal sent from assigned peripheral
26 READY	This bit reflects the current state of Ready Signal sent from assigned peripheral
25 LOCK	This bit reflects the current state of the DMA Channel Lock for a GPMI Channel.
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved

HW_APBH_CH15_DEBUG1 field descriptions (continued)

Field	Description
4–0 STATEMACHINE	PIO Display of the DMA channel 15 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x14 **TERMINATE** — When a terminate signal is set, the state machine enters this state until the current AHB transfer is completed. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1D **HALT_AFTER_TERM** — If HALTONTERMINATE is set and a terminate signal is set, the state machine enters this state and effectively halts. A channel reset is required to exit this state 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts. 0x1F **WAIT_READY** — When the NAND Wait for Ready bit is set, the state machine enters this state until the GPMI device indicates that the external device is ready.

6.5.119 AHB to APBH DMA channel 15 Debug Information (HW_APBH_CH15_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA channel 15.

Because channel 15 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBH_CH15_DEBUG2 – 8000_4000h base + 7F0h offset = 8000_47F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_CH15_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

6.5.120 APBH Bridge Version Register (HW_APBH_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_APBH_VERSION.B.MAJOR != 3)
   Error();
```

Address: HW_APBH_VERSION – 8000_4000h base + 800h offset = 8000_4800h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Re-set	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBH_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 7
AHB-to-APBX Bridge with DMA (APBX-Bridge-DMA)

7.1 AHB-to-APBX Bridge Overview

The AHB-to-APBX bridge provides the device with an inexpensive peripheral attachment bus running on the AHB's XCLK. (The X in APB*X* denotes that the APBX runs on a crystal-derived clock, as compared to APB*H*, which is synchronous to HCLK.)

As shown in the following figure, the AHB-to-APBX bridge includes the AHB-to-APB PIO bridge for a memory-mapped I/O to the APB devices, as well a central DMA facility for devices on this bus. Each one of the APB peripherals is documented in their own chapters elsewhere in this document.

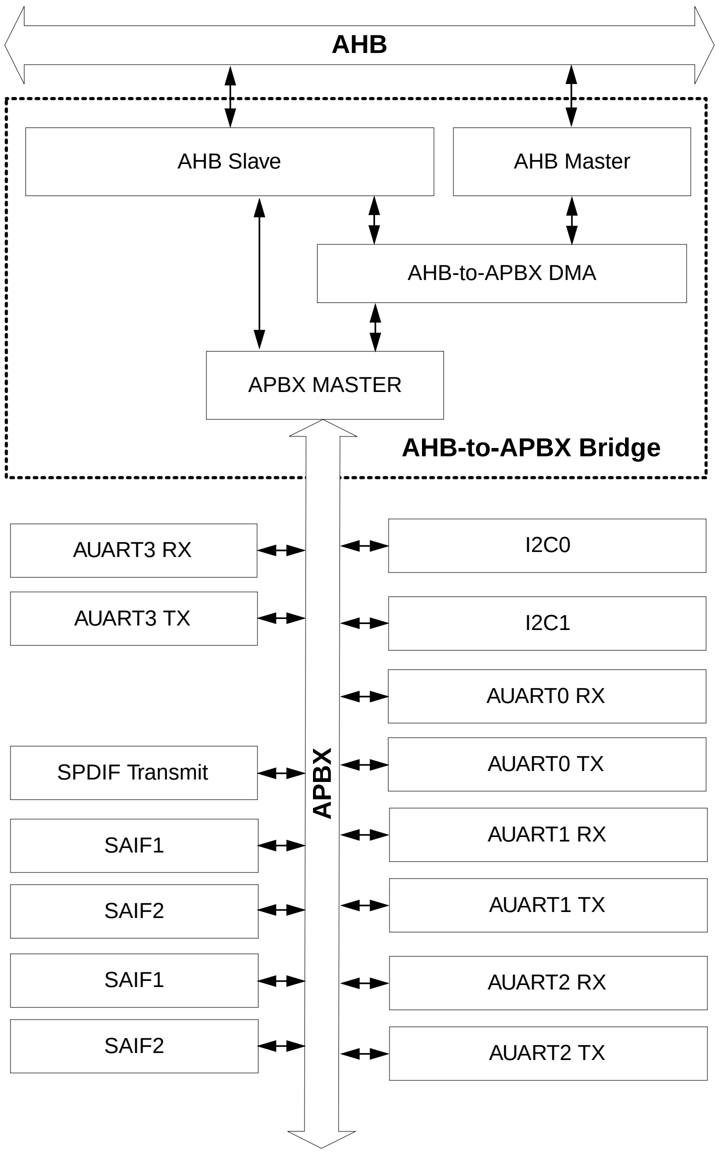


Figure 7-1. AHB-to-APBX Bridge DMA Block Diagram

The DMA controller uses the APBX bus to transfer read and write data to and from each peripheral. There is no separate DMA bus for these devices. Contention between the DMA's use of the APBX bus and AHB-to-APB bridge functions' use of the APBX is mediated by an internal arbitration logic. For contention between these two units, the DMA is favored and the AHB slave will report not ready through its HREADY output until the bridge transfer completes. The arbiter tracks repeated lockouts and inverts the priority, so that the CPU is guaranteed every fourth transfer on the APB.

7.2 APBX DMA

The DMA supports sixteen channels of DMA services, as shown in the following table. The shared DMA resource allows each independent channel to follow a simple chained command list. Command chains are built up using the DMA command structure, as shown in Figure 7-2.

Table 7-1. APBX DMA Channel Assignments

APBX DMA Channel #	USAGE
0	AUART4 RX
1	AUART4 TX
2	SPDIF TX
3	EMPTY
4	SAIF0
5	SAIF1
6	I2C0
7	I2C1
8	AUART0 RX
9	AUART0 TX
10	AUART1 RX
11	AUART1 TX
12	AUART2 RX
13	AUART2 TX
14	AUART3 RX
15	AUART3 TX

A single command structure or channel command word specifies a number of operations to be performed by the DMA in support of a given device. Thus, the CPU can set up large units of work, chaining together many DMA channel command words, pass them off to the DMA and have no further concern for the device until the DMA completion interrupt occurs. The i.MX28 is designed to have enough intelligence in the DMA and the devices to keep the interrupt frequency from any device below 1 KHz (arrival intervals longer than 1 ms).

Thus, a single command structure can issue 32-bit PIO write operations to key registers in the associated device using the same APB bus and the controls, it uses to write DMA data bytes to the device. For example, this allows a chain of operations to be issued to the serial audio interface to send command bytes, address bytes, and data transfers, where the command and the address structure is completely under software control, but the administration of that transfer is handled autonomously by the DMA.

Each DMA structure can have from 0 to 15 PIO words appended to it. The #PIOWORDs field, if non-zero, instructs the DMA engine to copy these words to the APB, beginning at PADDR = 0x0000 and incrementing its PADDR for each cycle. (Note that for APBX DMA Channel 8, which is the UART RX channel, the first PIO word in the DMA command is CTRL0. However, for APBX DMA Channel 9, which is the UART TX, the first PIO word in a DMA command is CTRL1.)

The DMA master generates only normal read/write transfers to the APBX. It does not generate set, clear, or toggle SCT transfers.

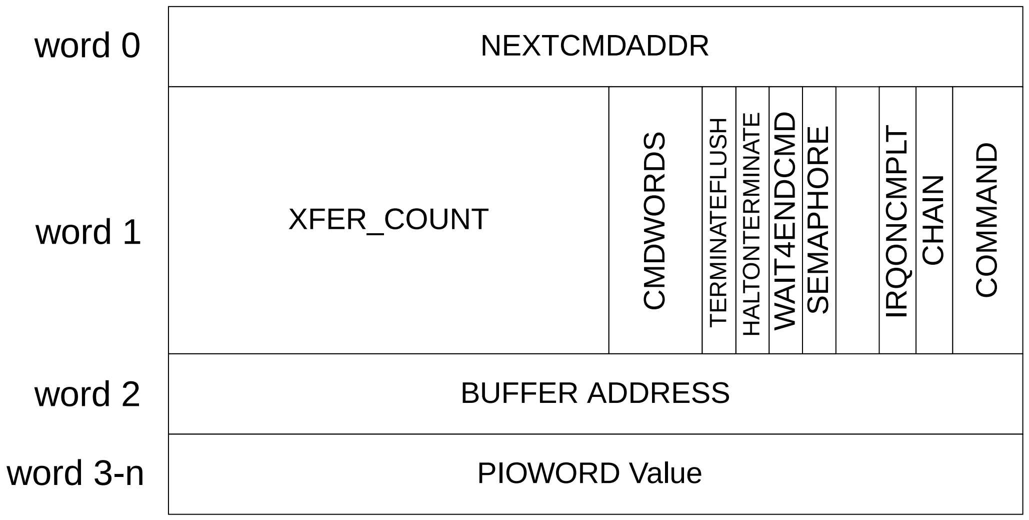


Figure 7-2. AHB-to-APBX Bridge DMA Channel Command Structure

After any requested PIO words have been transferred to the peripheral, the DMA examines the two-bit command field in the channel command structure. The following table shows the four commands implemented by the DMA.

Table 7-2. APBX DMA Commands

DMA COMMAND	Usage
00	NO_DMA_XFER. Perform any requested PIO word transfers, but terminate the command before any DMA transfer.
01	DMA_WRITE. Perform any requested PIO word transfers, and then perform a DMA transfer from the peripheral for the specified number of bytes.

DMA COMMAND	Usage
10	DMA_READ. Perform any requested PIO word transfers, and then perform a DMA transfer to the peripheral for the specified number of bytes.
11	Reserved

DMA_WRITE operations copy data bytes to the system memory (on-chip RAM or SDRAM) from the associated peripheral. Each peripheral has a target PADDR value that it expects to receive DMA bytes. This association is synthesized in the DMA. The DMA_WRITE transfer uses the BUFFER_ADDRESS word in the command structure to point to the beginning byte to write data from the peripheral.

DMA_READ operations copy data bytes to the APB peripheral from the system memory. The DMA engine contains a shared byte aligner that aligns bytes from system memory to or from the peripherals. Peripherals always assume little-endian-aligned data arrives or departs on their 32-bit APB. The DMA_READ transfer uses the BUFFER_ADDRESS word in the command structure to point to the DMA data buffer to be read by the DMA_READ command.

The NO_DMA_XFER command is used to write PIO words to a device without performing any DMA data byte transfers.

As each DMA command completes, it triggers the DMA to load the next DMA command structure in the chain. The normal flow list of DMA commands is found by following the NEXTCMD_ADDR pointer in the DMA command structure. If the wait-for-end-command bit (WAIT4ENDCMD) is set in a command structure, then the DMA channel waits for the device to signal completion of a command by toggling the apx_endcmcd signal before proceeding to load and execute the next command structure. Then, if DECREMENT_SEMAPHORE is set, the semaphore is decremented after the end command is seen.

A detailed bit-field view of the DMA command structure is shown in the following table, which shows a field that specifies the number of bytes to be transferred by this DMA command. The transfer count mechanism is duplicated in the associated peripheral, either as an implied or as a specified count in the peripheral.

Table 7-3. DMA Channel Command Word in System Memory

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Number DMA Bytes to Transfer																Number PIO Words to Write						TERMINATEFLUSH	HALTONTERMINATE	WAIT4ENDCMD	DECREMENT SEMAPHORE	Reserved		IRQ_FINISH	CHAIN	COMMAND	
DMA Buffer or Alternate CCW																															
Zero or More PIO Words to Write to the Associated Peripheral Starting at its Base Address on the APBX Bus																															

Figure 7-3 shows the CHAIN bit in bit 2 of the second word of the command structure. This bit is set to 1, if the NEXT_COMMAND_ADDRESS contains a pointer to another DMA command structure. If a null pointer (0) is loaded into the NEXT_COMMAND_ADDRESS, it will not be detected by the DMA hardware. Only the CHAIN bit indicates whether a valid list exists beyond the current structure.

If the IRQ_COMPLETE bit is set in the command structure, then the last act of the DMA before loading the next command is to set the interrupt status bit corresponding to the current channel. The sticky interrupt request bit in the DMA CSR remains set until cleared by the software. It can be used to interrupt the CPU.

Each channel has an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore is non-zero, the channel is ready to run and process commands and DMA transfers. Whenever a command finishes its DMA transfer, it checks the DECREMENT_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the IDLE state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its IDLE state, then it uses the value in the HW_APBX_CHn_NXTCMDAR (next command address register) to fetch a pointer to the next command to process. NOTE: this is a double indirect case. This method allows software to append to a running command list under the protection of the counting semaphore.

Receiving an IRQ for HALTONTERMINATE (HOT) is a new feature in the APBH/X DMA descriptor that allows certain peripheral block (for example, GPMI, SSP, I2C) to signal to the DMA engine that an error has occurred. In prior chips, if a block stalled due to an error, the only practical way to discover this in software was through a timer of some sort, or to poll the block. Now, an HOT signal is sent from the peripheral to the DMA engine and causes an IRQ after terminating the DMA descriptor being executed. Note not all peripheral block support this termination feature.

Under a normal operation mode, when termination occurs, DMA would stop the current descriptor and discard any data remaining in the FIFO. By setting TERMINATEFLUSH bit in the descriptor command word, DMA will flush out all the remainder data in the FIFO to system memory for write DMA operation before issue HOT IRQ.

Therefore, it is recommended that software use this signal as follows:

- Always set HALTONTERMINATE to 1 in a DMA descriptor. That way, if a peripheral signals HOT, the transfer will end, leaving the peripheral block and the DMA engine synchronized (but at the end of a command).
- When an IRQ from an APBH/X channel is received, and the IRQ is determined to be due to an error (as opposed to an IRQONCOMPLETE interrupt) the software should:

1. Reset the channel.
2. Determine the error from error reporting in the peripheral block, then manage the error in the peripheral that is attached to that channel in whatever appropriate way exists for that device (software recovery, device reset, block reset, and so on).

To start processing the first time, software creates the command list to be processed. It writes the address of the first command into the HW_APBX_CHn_NXTCMDAR register, and then writes a 1 to the counting semaphore in HW_APBX_CHn_SEMA. The DMA channel loads HW_APBX_CHn_CURCMDAR register and then enters the normal state machine processing for the next command. When the software writes a value to the counting semaphore, it is added to the semaphore count by hardware, protecting the case where both hardware and software are trying to change the semaphore on the same clock edge.

Software can examine the value of HW_APBX_CHn_CURCMDAR at any time to determine the location of the command structure that is currently being processed.

7.3 DMA Chain Example

The example in the following figure shows how to bring the basic items together to make a simple DMA chain to read PCM samples and send them out the Audio Output (DAC) using one DMA channel. This example shows three command structures linked together using their normal command list pointers. The first command writes a single PIO word to the HW_AUDIOOUT_CTRL0 register with a new word count for the DAC. This first command also performs a 512 byte DMA_READ operation to read the data block bytes into the DAC. A second and a third DMA command structure also performs a DMA_READ operation to handle circular buffer style outputs. The completion of each command structure generates an interrupt request. In addition, each command structure decrements the semaphore. If the decompression software has not provided a buffer in a timely fashion, then the DMA will stall. Without the decrement semaphore interlocking, then the DMA will continue to output a stream of samples. In this mode, it is up to software to use the interrupts to synchronize outputs so that underruns do not occur.

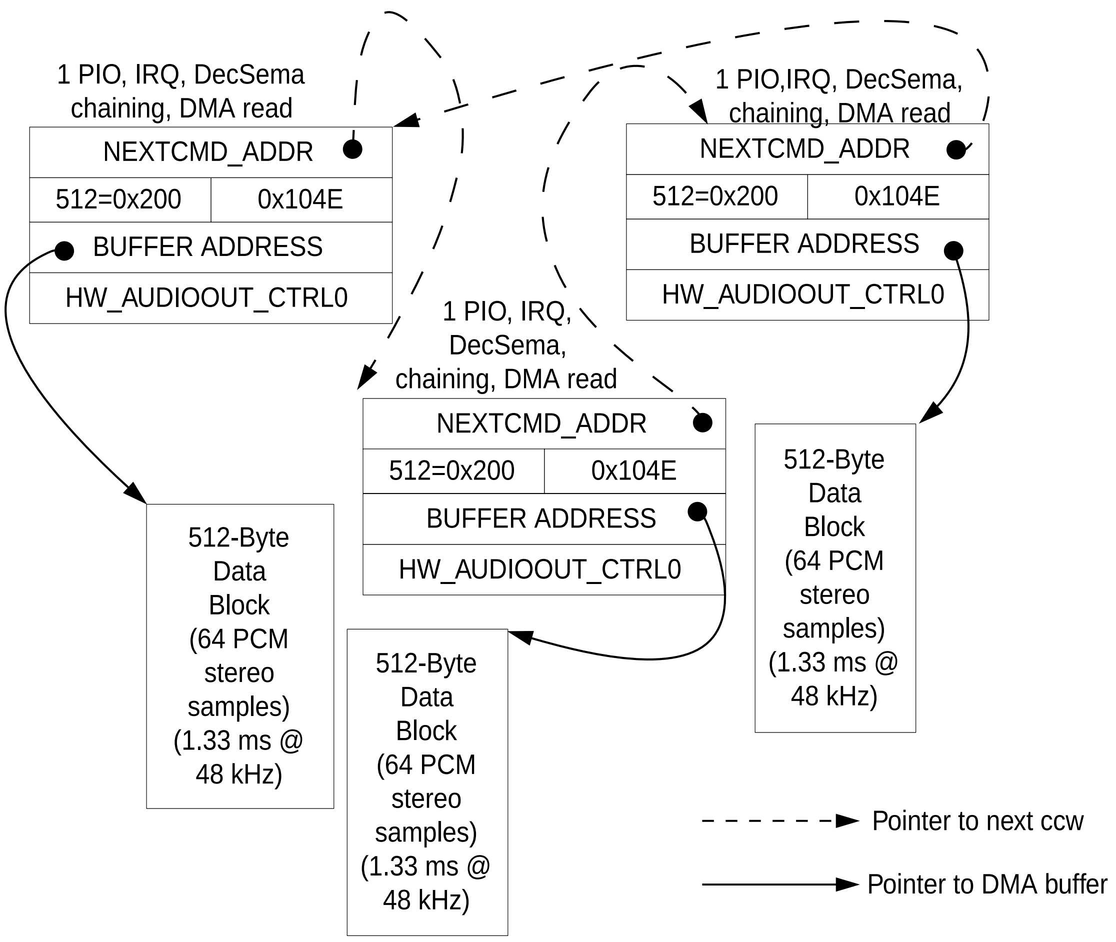

1 PIO, IRQ, DecSema chaining, DMA read
NEXTCMD_ADDR
512=0x200
0x104E
BUFFER ADDRESS
HW_AUDIOOUT_CTRL0
1 PIO,IRQ, DecSema, chaining, DMA read
NEXTCMD_ADDR
512=0x200
0x104E
BUFFER ADDRESS
HW_AUDIOOUT_CTRL0
1 PIO, IRQ, DecSema, chaining, DMA read
NEXTCMD_ADDR
512=0x200
0x104E
BUFFER ADDRESS
HW_AUDIOOUT_CTRL0
512-Byte Data Block (64 PCM stereo samples) (1.33 ms @ 48 kHz)
512-Byte Data Block (64 PCM stereo samples) (1.33 ms @ 48 kHz)
512-Byte Data Block (64 PCM stereo samples) (1.33 ms @ 48 kHz)
Pointer to next ccw
Pointer to DMA buffer

Figure 7-3. AHB-to-APBX Bridge DMA AUDIOOUT (DAC) Example Command Chain

Note that each word of the three-word DMA command structure corresponds to a PIO register of the DMA that is accessible on the APBX bus. Normally, the DMA copies the next command structure onto these registers for processing at the start of each command by following the value of the pointer previously loaded into the NEXTCMD_ADDR register. To start DMA processing, for the first command, one must initialize the PIO registers of the desired channel. First load the next command address register with a pointer to the first command to be loaded. Then, write a one to the counting semaphore register. This causes the DMA to schedule the targeted channel for DMA command structure load, as if it just finished its previous command.

7.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

7.5 Programmable Registers

APBX Hardware Register Format Summary

HW_APBX memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_4000	AHB to APBX Bridge Control Register 0 (HW_APBX_CTRL0)	32	R/W	C000_0000h	7.5.1/511
8002_4010	AHB to APBX Bridge Control Register 1 (HW_APBX_CTRL1)	32	R/W	0000_0000h	7.5.2/511
8002_4020	AHB to APBX Bridge Control and Status Register 2 (HW_APBX_CTRL2)	32	R/W	0000_0000h	7.5.3/515
8002_4030	AHB to APBX Bridge Channel Register (HW_APBX_CHANNEL_CTRL)	32	R/W	0000_0000h	7.5.4/519
8002_4040	AHB to APBX DMA Device Assignment Register (HW_APBX_DEVSEL)	32	R	0000_0000h	7.5.5/521
8002_4100	APBX DMA Channel 0 Current Command Address Register (HW_APBX_CH0_CURCMDAR)	32	R	0000_0000h	7.5.6/522
8002_4110	APBX DMA Channel 0 Next Command Address Register (HW_APBX_CH0_NXTCMDAR)	32	R/W	0000_0000h	7.5.7/523
8002_4120	APBX DMA Channel 0 Command Register (HW_APBX_CH0_CMD)	32	R	0000_0000h	7.5.8/523
8002_4130	APBX DMA Channel 0 Buffer Address Register (HW_APBX_CH0_BAR)	32	R	0000_0000h	7.5.9/525
8002_4140	APBX DMA Channel 0 Semaphore Register (HW_APBX_CH0_SEMA)	32	R/W	0000_0000h	7.5.10/526
8002_4150	AHB to APBX DMA Channel 0 Debug Information (HW_APBX_CH0_DEBUG1)	32	R	00A0_0000h	7.5.11/527
8002_4160	AHB to APBX DMA Channel 0 Debug Information (HW_APBX_CH0_DEBUG2)	32	R	0000_0000h	7.5.12/529
8002_4170	APBX DMA Channel 1 Current Command Address Register (HW_APBX_CH1_CURCMDAR)	32	R	0000_0000h	7.5.13/530
8002_4180	APBX DMA Channel 1 Next Command Address Register (HW_APBX_CH1_NXTCMDAR)	32	R/W	0000_0000h	7.5.14/531
8002_4190	APBX DMA Channel 1 Command Register (HW_APBX_CH1_CMD)	32	R	0000_0000h	7.5.15/531

HW_APBX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_41A0	APBX DMA Channel 1 Buffer Address Register (HW_APBX_CH1_BAR)	32	R	0000_0000h	7.5.16/533
8002_41B0	APBX DMA Channel 1 Semaphore Register (HW_APBX_CH1_SEMA)	32	R/W	0000_0000h	7.5.17/534
8002_41C0	AHB to APBX DMA Channel 1 Debug Information (HW_APBX_CH1_DEBUG1)	32	R	00A0_0000h	7.5.18/535
8002_41D0	AHB to APBX DMA Channel 1 Debug Information (HW_APBX_CH1_DEBUG2)	32	R	0000_0000h	7.5.19/537
8002_41E0	APBX DMA Channel 2 Current Command Address Register (HW_APBX_CH2_CURCMDAR)	32	R	0000_0000h	7.5.20/537
8002_41F0	APBX DMA Channel 2 Next Command Address Register (HW_APBX_CH2_NXTCMDAR)	32	R/W	0000_0000h	7.5.21/538
8002_4200	APBX DMA Channel 2 Command Register (HW_APBX_CH2_CMD)	32	R	0000_0000h	7.5.22/538
8002_4210	APBX DMA Channel 2 Buffer Address Register (HW_APBX_CH2_BAR)	32	R	0000_0000h	7.5.23/540
8002_4220	APBX DMA Channel 2 Semaphore Register (HW_APBX_CH2_SEMA)	32	R/W	0000_0000h	7.5.24/541
8002_4230	AHB to APBX DMA Channel 2 Debug Information (HW_APBX_CH2_DEBUG1)	32	R	00A0_0000h	7.5.25/541
8002_4240	AHB to APBX DMA Channel 2 Debug Information (HW_APBX_CH2_DEBUG2)	32	R	0000_0000h	7.5.26/544
8002_4250	APBX DMA Channel 3 Current Command Address Register (HW_APBX_CH3_CURCMDAR)	32	R	0000_0000h	7.5.27/544
8002_4260	APBX DMA Channel 3 Next Command Address Register (HW_APBX_CH3_NXTCMDAR)	32	R/W	0000_0000h	7.5.28/545
8002_4270	APBX DMA Channel 3 Command Register (HW_APBX_CH3_CMD)	32	R	0000_0000h	7.5.29/545
8002_4280	APBX DMA Channel 3 Buffer Address Register (HW_APBX_CH3_BAR)	32	R	0000_0000h	7.5.30/547
8002_4290	APBX DMA Channel 3 Semaphore Register (HW_APBX_CH3_SEMA)	32	R/W	0000_0000h	7.5.31/548
8002_42A0	AHB to APBX DMA Channel 3 Debug Information (HW_APBX_CH3_DEBUG1)	32	R	0000_0000h	7.5.32/548
8002_42B0	AHB to APBX DMA Channel 3 Debug Information (HW_APBX_CH3_DEBUG2)	32	R	0000_0000h	7.5.33/551
8002_42C0	APBX DMA Channel 4 Current Command Address Register (HW_APBX_CH4_CURCMDAR)	32	R	0000_0000h	7.5.34/551
8002_42D0	APBX DMA Channel 4 Next Command Address Register (HW_APBX_CH4_NXTCMDAR)	32	R/W	0000_0000h	7.5.35/552
8002_42E0	APBX DMA Channel 4 Command Register (HW_APBX_CH4_CMD)	32	R	0000_0000h	7.5.36/552

HW_APBX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_42F0	APBX DMA Channel 4 Buffer Address Register (HW_APBX_CH4_BAR)	32	R	0000_0000h	7.5.37/554
8002_4300	APBX DMA Channel 4 Semaphore Register (HW_APBX_CH4_SEMA)	32	R/W	0000_0000h	7.5.38/555
8002_4310	AHB to APBX DMA Channel 4 Debug Information (HW_APBX_CH4_DEBUG1)	32	R	00A0_0000h	7.5.39/555
8002_4320	AHB to APBX DMA Channel 4 Debug Information (HW_APBX_CH4_DEBUG2)	32	R	0000_0000h	7.5.40/558
8002_4330	APBX DMA Channel 5 Current Command Address Register (HW_APBX_CH5_CURCMDAR)	32	R	0000_0000h	7.5.41/558
8002_4340	APBX DMA Channel 5 Next Command Address Register (HW_APBX_CH5_NXTCMDAR)	32	R/W	0000_0000h	7.5.42/559
8002_4350	APBX DMA Channel 5 Command Register (HW_APBX_CH5_CMD)	32	R	0000_0000h	7.5.43/559
8002_4360	APBX DMA Channel 5 Buffer Address Register (HW_APBX_CH5_BAR)	32	R	0000_0000h	7.5.44/561
8002_4370	APBX DMA Channel 5 Semaphore Register (HW_APBX_CH5_SEMA)	32	R/W	0000_0000h	7.5.45/562
8002_4380	AHB to APBX DMA Channel 5 Debug Information (HW_APBX_CH5_DEBUG1)	32	R	00A0_0000h	7.5.46/562
8002_4390	AHB to APBX DMA Channel 5 Debug Information (HW_APBX_CH5_DEBUG2)	32	R	0000_0000h	7.5.47/565
8002_43A0	APBX DMA Channel 6 Current Command Address Register (HW_APBX_CH6_CURCMDAR)	32	R	0000_0000h	7.5.48/565
8002_43B0	APBX DMA Channel 6 Next Command Address Register (HW_APBX_CH6_NXTCMDAR)	32	R/W	0000_0000h	7.5.49/566
8002_43C0	APBX DMA Channel 6 Command Register (HW_APBX_CH6_CMD)	32	R	0000_0000h	7.5.50/566
8002_43D0	APBX DMA Channel 6 Buffer Address Register (HW_APBX_CH6_BAR)	32	R	0000_0000h	7.5.51/568
8002_43E0	APBX DMA Channel 6 Semaphore Register (HW_APBX_CH6_SEMA)	32	R/W	0000_0000h	7.5.52/569
8002_43F0	AHB to APBX DMA Channel 6 Debug Information (HW_APBX_CH6_DEBUG1)	32	R	00A0_0000h	7.5.53/570
8002_4400	AHB to APBX DMA Channel 6 Debug Information (HW_APBX_CH6_DEBUG2)	32	R	0000_0000h	7.5.54/572
8002_4410	APBX DMA Channel 7 Current Command Address Register (HW_APBX_CH7_CURCMDAR)	32	R	0000_0000h	7.5.55/572
8002_4420	APBX DMA Channel 7 Next Command Address Register (HW_APBX_CH7_NXTCMDAR)	32	R/W	0000_0000h	7.5.56/573
8002_4430	APBX DMA Channel 7 Command Register (HW_APBX_CH7_CMD)	32	R	0000_0000h	7.5.57/573

HW_APBX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_4440	APBX DMA Channel 7 Buffer Address Register (HW_APBX_CH7_BAR)	32	R	0000_0000h	7.5.58/575
8002_4450	APBX DMA Channel 7 Semaphore Register (HW_APBX_CH7_SEMA)	32	R/W	0000_0000h	7.5.59/576
8002_4460	AHB to APBX DMA Channel 7 Debug Information (HW_APBX_CH7_DEBUG1)	32	R	00A0_0000h	7.5.60/577
8002_4470	AHB to APBX DMA Channel 7 Debug Information (HW_APBX_CH7_DEBUG2)	32	R	0000_0000h	7.5.61/579
8002_4480	APBX DMA Channel 8 Current Command Address Register (HW_APBX_CH8_CURCMDAR)	32	R	0000_0000h	7.5.62/580
8002_4490	APBX DMA Channel 8 Next Command Address Register (HW_APBX_CH8_NXTCMDAR)	32	R/W	0000_0000h	7.5.63/580
8002_44A0	APBX DMA Channel 8 Command Register (HW_APBX_CH8_CMD)	32	R	0000_0000h	7.5.64/581
8002_44B0	APBX DMA Channel 8 Buffer Address Register (HW_APBX_CH8_BAR)	32	R	0000_0000h	7.5.65/583
8002_44C0	APBX DMA Channel 8 Semaphore Register (HW_APBX_CH8_SEMA)	32	R/W	0000_0000h	7.5.66/584
8002_44D0	AHB to APBX DMA Channel 8 Debug Information (HW_APBX_CH8_DEBUG1)	32	R	00A0_0000h	7.5.67/584
8002_44E0	AHB to APBX DMA Channel 8 Debug Information (HW_APBX_CH8_DEBUG2)	32	R	0000_0000h	7.5.68/587
8002_44F0	APBX DMA Channel 9 Current Command Address Register (HW_APBX_CH9_CURCMDAR)	32	R	0000_0000h	7.5.69/587
8002_4500	APBX DMA Channel 9 Next Command Address Register (HW_APBX_CH9_NXTCMDAR)	32	R/W	0000_0000h	7.5.70/588
8002_4510	APBX DMA Channel 9 Command Register (HW_APBX_CH9_CMD)	32	R	0000_0000h	7.5.71/588
8002_4520	APBX DMA Channel 9 Buffer Address Register (HW_APBX_CH9_BAR)	32	R	0000_0000h	7.5.72/590
8002_4530	APBX DMA Channel 9 Semaphore Register (HW_APBX_CH9_SEMA)	32	R/W	0000_0000h	7.5.73/591
8002_4540	AHB to APBX DMA Channel 9 Debug Information (HW_APBX_CH9_DEBUG1)	32	R	00A0_0000h	7.5.74/592
8002_4550	AHB to APBX DMA Channel 9 Debug Information (HW_APBX_CH9_DEBUG2)	32	R	0000_0000h	7.5.75/594
8002_4560	APBX DMA Channel 10 Current Command Address Register (HW_APBX_CH10_CURCMDAR)	32	R	0000_0000h	7.5.76/594
8002_4570	APBX DMA Channel 10 Next Command Address Register (HW_APBX_CH10_NXTCMDAR)	32	R/W	0000_0000h	7.5.77/595
8002_4580	APBX DMA Channel 10 Command Register (HW_APBX_CH10_CMD)	32	R	0000_0000h	7.5.78/595

HW_APBX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_4590	APBX DMA Channel 10 Buffer Address Register (HW_APBX_CH10_BAR)	32	R	0000_0000h	7.5.79/597
8002_45A0	APBX DMA Channel 10 Semaphore Register (HW_APBX_CH10_SEMA)	32	R/W	0000_0000h	7.5.80/598
8002_45B0	AHB to APBX DMA Channel 10 Debug Information (HW_APBX_CH10_DEBUG1)	32	R	00A0_0000h	7.5.81/599
8002_45C0	AHB to APBX DMA Channel 10 Debug Information (HW_APBX_CH10_DEBUG2)	32	R	0000_0000h	7.5.82/601
8002_45D0	APBX DMA Channel 11 Current Command Address Register (HW_APBX_CH11_CURCMDAR)	32	R	0000_0000h	7.5.83/602
8002_45E0	APBX DMA Channel 11 Next Command Address Register (HW_APBX_CH11_NXTCMDAR)	32	R/W	0000_0000h	7.5.84/602
8002_45F0	APBX DMA Channel 11 Command Register (HW_APBX_CH11_CMD)	32	R	0000_0000h	7.5.85/603
8002_4600	APBX DMA Channel 11 Buffer Address Register (HW_APBX_CH11_BAR)	32	R	0000_0000h	7.5.86/605
8002_4610	APBX DMA Channel 11 Semaphore Register (HW_APBX_CH11_SEMA)	32	R/W	0000_0000h	7.5.87/605
8002_4620	AHB to APBX DMA Channel 11 Debug Information (HW_APBX_CH11_DEBUG1)	32	R	00A0_0000h	7.5.88/606
8002_4630	AHB to APBX DMA Channel 11 Debug Information (HW_APBX_CH11_DEBUG2)	32	R	0000_0000h	7.5.89/609
8002_4640	APBX DMA Channel 12 Current Command Address Register (HW_APBX_CH12_CURCMDAR)	32	R	0000_0000h	7.5.90/609
8002_4650	APBX DMA Channel 12 Next Command Address Register (HW_APBX_CH12_NXTCMDAR)	32	R/W	0000_0000h	7.5.91/610
8002_4660	APBX DMA Channel 12 Command Register (HW_APBX_CH12_CMD)	32	R	0000_0000h	7.5.92/610
8002_4670	APBX DMA Channel 12 Buffer Address Register (HW_APBX_CH12_BAR)	32	R	0000_0000h	7.5.93/612
8002_4680	APBX DMA Channel 12 Semaphore Register (HW_APBX_CH12_SEMA)	32	R/W	0000_0000h	7.5.94/613
8002_4690	AHB to APBX DMA Channel 12 Debug Information (HW_APBX_CH12_DEBUG1)	32	R	00A0_0000h	7.5.95/614
8002_46A0	AHB to APBX DMA Channel 12 Debug Information (HW_APBX_CH12_DEBUG2)	32	R	0000_0000h	7.5.96/616
8002_46B0	APBX DMA Channel 13 Current Command Address Register (HW_APBX_CH13_CURCMDAR)	32	R	0000_0000h	7.5.97/616
8002_46C0	APBX DMA Channel 13 Next Command Address Register (HW_APBX_CH13_NXTCMDAR)	32	R/W	0000_0000h	7.5.98/617
8002_46D0	APBX DMA Channel 13 Command Register (HW_APBX_CH13_CMD)	32	R	0000_0000h	7.5.99/617

HW_APBX memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_46E0	APBX DMA Channel 13 Buffer Address Register (HW_APBX_CH13_BAR)	32	R	0000_0000h	7.5.100/619
8002_46F0	APBX DMA Channel 13 Semaphore Register (HW_APBX_CH13_SEMA)	32	R/W	0000_0000h	7.5.101/620
8002_4700	AHB to APBX DMA Channel 13 Debug Information (HW_APBX_CH13_DEBUG1)	32	R	00A0_0000h	7.5.102/620
8002_4710	AHB to APBX DMA Channel 13 Debug Information (HW_APBX_CH13_DEBUG2)	32	R	0000_0000h	7.5.103/623
8002_4720	APBX DMA Channel 14 Current Command Address Register (HW_APBX_CH14_CURCMDAR)	32	R	0000_0000h	7.5.104/623
8002_4730	APBX DMA Channel 14 Next Command Address Register (HW_APBX_CH14_NXTCMDAR)	32	R/W	0000_0000h	7.5.105/624
8002_4740	APBX DMA Channel 14 Command Register (HW_APBX_CH14_CMD)	32	R	0000_0000h	7.5.106/624
8002_4750	APBX DMA Channel 14 Buffer Address Register (HW_APBX_CH14_BAR)	32	R	0000_0000h	7.5.107/626
8002_4760	APBX DMA Channel 14 Semaphore Register (HW_APBX_CH14_SEMA)	32	R/W	0000_0000h	7.5.108/627
8002_4770	AHB to APBX DMA Channel 14 Debug Information (HW_APBX_CH14_DEBUG1)	32	R	00A0_0000h	7.5.109/628
8002_4780	AHB to APBX DMA Channel 14 Debug Information (HW_APBX_CH14_DEBUG2)	32	R	0000_0000h	7.5.110/630
8002_4790	APBX DMA Channel 15 Current Command Address Register (HW_APBX_CH15_CURCMDAR)	32	R	0000_0000h	7.5.111/630
8002_47A0	APBX DMA Channel 15 Next Command Address Register (HW_APBX_CH15_NXTCMDAR)	32	R/W	0000_0000h	7.5.112/631
8002_47B0	APBX DMA Channel 15 Command Register (HW_APBX_CH15_CMD)	32	R	0000_0000h	7.5.113/631
8002_47C0	APBX DMA Channel 15 Buffer Address Register (HW_APBX_CH15_BAR)	32	R	0000_0000h	7.5.114/633
8002_47D0	APBX DMA Channel 15 Semaphore Register (HW_APBX_CH15_SEMA)	32	R/W	0000_0000h	7.5.115/634
8002_47E0	AHB to APBX DMA Channel 15 Debug Information (HW_APBX_CH15_DEBUG1)	32	R	00A0_0000h	7.5.116/634
8002_47F0	AHB to APBX DMA Channel 15 Debug Information (HW_APBX_CH15_DEBUG2)	32	R	0000_0000h	7.5.117/637
8002_4800	APBX Bridge Version Register (HW_APBX_VERSION)	32	R	0202_0000h	7.5.118/637

7.5.1 AHB to APBX Bridge Control Register 0 (HW_APBX_CTRL0)

The APBX CTRL 0 provides overall control and IRQ status of the AHB to APBX bridge and DMA.

HW_APBX_CTRL0: 0x000

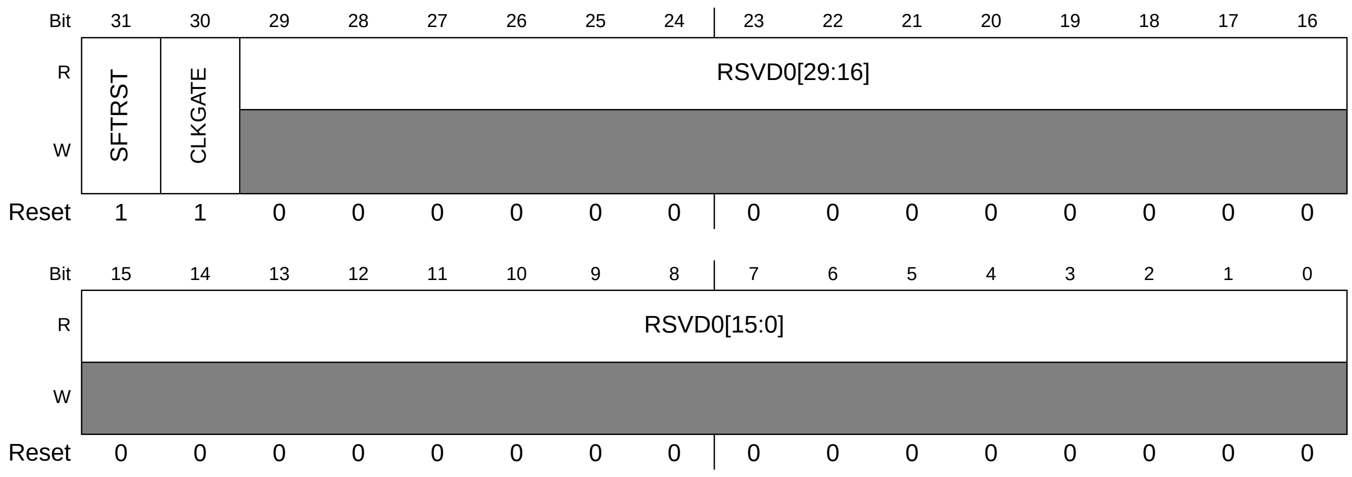
HW_APBX_CTRL0_SET: 0x004

HW_APBX_CTRL0_CLR: 0x008

HW_APBX_CTRL0_TOG: 0x00C

This register contains softreset, clock gating bits.

Address: HW_APBX_CTRL0 – 8002_4000h base + 0h offset = 8002_4000h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
SFTRST
CLKGATE
RSVD0[29:16]
Reset 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSVD0[15:0]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CTRL0 field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal APBX DMA operation. Set this bit to one (default) to disable clocking with the APBX DMA and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the APBX DMA block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29–0 RSVD0	Reserved, always set to zero.

7.5.2 AHB to APBX Bridge Control Register 1 (HW_APBX_CTRL1)

The APBX CTRL 1 provides channel complete IRQ status of the AHB to APBX bridge and DMA.

HW_APBX_CTRL1: 0x010

HW_APBX_CTRL1_SET: 0x014

HW_APBX_CTRL1_CLR: 0x018

HW_APBX_CTRL1_TOG: 0x01C

This register contains the per channel interrupt status bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

EXAMPLE

```
BF_WR(APBX_CTRL1, CH5_CMDCMPLT_IRQ, 0);  // use bitfield write macro
BF_APBX_CTRL1.CH5_CMDCMPLT_IRQ = 0;      // or, assign to register struct's bitfield
```

Address: HW_APBX_CTRL1 – 8002_4000h base + 10h offset = 8002_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CH15_CMDCMPLT_IRQ_EN	CH14_CMDCMPLT_IRQ_EN	CH13_CMDCMPLT_IRQ_EN	CH12_CMDCMPLT_IRQ_EN	CH11_CMDCMPLT_IRQ_EN	CH10_CMDCMPLT_IRQ_EN	CH9_CMDCMPLT_IRQ_EN	CH8_CMDCMPLT_IRQ_EN	CH7_CMDCMPLT_IRQ_EN	CH6_CMDCMPLT_IRQ_EN	CH5_CMDCMPLT_IRQ_EN	CH4_CMDCMPLT_IRQ_EN	CH3_CMDCMPLT_IRQ_EN	CH2_CMDCMPLT_IRQ_EN	CH1_CMDCMPLT_IRQ_EN	CH0_CMDCMPLT_IRQ_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH15_CMDCMPLT_IRQ	CH14_CMDCMPLT_IRQ	CH13_CMDCMPLT_IRQ	CH12_CMDCMPLT_IRQ	CH11_CMDCMPLT_IRQ	CH10_CMDCMPLT_IRQ	CH9_CMDCMPLT_IRQ	CH8_CMDCMPLT_IRQ	CH7_CMDCMPLT_IRQ	CH6_CMDCMPLT_IRQ	CH5_CMDCMPLT_IRQ	CH4_CMDCMPLT_IRQ	CH3_CMDCMPLT_IRQ	CH2_CMDCMPLT_IRQ	CH1_CMDCMPLT_IRQ	CH0_CMDCMPLT_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CTRL1 field descriptions

Field	Description
31 CH15_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 15.
30 CH14_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 14.
29 CH13_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 13.

HW_APBX_CTRL1 field descriptions (continued)

Field	Description
28 CH12_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 12.
27 CH11_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 11.
26 CH10_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 10.
25 CH9_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 9.
24 CH8_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 8.
23 CH7_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 7.
22 CH6_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 6.
21 CH5_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 5.
20 CH4_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 4.
19 CH3_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 3.
18 CH2_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 2.
17 CH1_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 1.

HW_APBX_CTRL1 field descriptions (continued)

Field	Description
16 CH0_ CMDCMPLT_ IRQ_EN	Setting this bit enables the generation of an interrupt request for APBX DMA Channel 0.
15 CH15_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
14 CH14_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
13 CH13_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
12 CH12_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
11 CH11_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
10 CH10_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
9 CH9_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
8 CH8_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
7 CH7_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
6 CH6_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
5 CH5_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
4 CH4_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
3 CH3_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
2 CH2_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

HW_APBX_CTRL1 field descriptions (continued)

Field	Description
1 CH1_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.
0 CH0_ CMDCMPLT_IRQ	Interrupt request status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ANDed with its corresponding enable bit to generate an interrupt.

7.5.3 AHB to APBX Bridge Control and Status Register 2 (HW_APBX_CTRL2)

The APBX CTRL 2 provides channel error interrupts generated by the AHB to APBX DMA.

HW_APBX_CTRL2: 0x020

HW_APBX_CTRL2_SET: 0x024

HW_APBX_CTRL2_CLR: 0x028

HW_APBX_CTRL2_TOG: 0x02C

This register contains the per channel bus error interrupt status bits and the per channel completion interrupt enable bits. Each channel has a dedicated interrupt vector in the vectored interrupt controller.

Address: HW_APBX_CTRL2 – 8002_4000h base + 20h offset = 8002_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CH15_ERROR_STATUS	CH14_ERROR_STATUS	CH13_ERROR_STATUS	CH12_ERROR_STATUS	CH11_ERROR_STATUS	CH10_ERROR_STATUS	CH9_ERROR_STATUS	CH8_ERROR_STATUS	CH7_ERROR_STATUS	CH6_ERROR_STATUS	CH5_ERROR_STATUS	CH4_ERROR_STATUS	CH3_ERROR_STATUS	CH2_ERROR_STATUS	CH1_ERROR_STATUS	CH0_ERROR_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CH15_ERROR_IRQ	CH14_ERROR_IRQ	CH13_ERROR_IRQ	CH12_ERROR_IRQ	CH11_ERROR_IRQ	CH10_ERROR_IRQ	CH9_ERROR_IRQ	CH8_ERROR_IRQ	CH7_ERROR_IRQ	CH6_ERROR_IRQ	CH5_ERROR_IRQ	CH4_ERROR_IRQ	CH3_ERROR_IRQ	CH2_ERROR_IRQ	CH1_ERROR_IRQ	CH0_ERROR_IRQ
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CTRL2 field descriptions

Field	Description
31 CH15_ERROR_ STATUS	Error status bit for APBX DMA Channel 15. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
30 CH14_ERROR_ STATUS	Error status bit for APBX DMA Channel 14. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
29 CH13_ERROR_ STATUS	Error status bit for APBX DMA Channel 13. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
28 CH12_ERROR_ STATUS	Error status bit for APBX DMA Channel 12. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
27 CH11_ERROR_ STATUS	Error status bit for APBX DMA Channel 11. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
26 CH10_ERROR_ STATUS	Error status bit for APBX DMA Channel 10. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.

HW_APBX_CTRL2 field descriptions (continued)

Field	Description
25 CH9_ERROR_ STATUS	Error status bit for APBX DMA Channel 9. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
24 CH8_ERROR_ STATUS	Error status bit for APBX DMA Channel 8. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
23 CH7_ERROR_ STATUS	Error status bit for APBX DMA Channel 7. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
22 CH6_ERROR_ STATUS	Error status bit for APBX DMA Channel 6. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
21 CH5_ERROR_ STATUS	Error status bit for APBX DMA Channel 5. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
20 CH4_ERROR_ STATUS	Error status bit for APBX DMA Channel 4. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
19 CH3_ERROR_ STATUS	Error status bit for APBX DMA Channel 3. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
18 CH2_ERROR_ STATUS	Error status bit for APBX DMA Channel 2. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination.

HW_APBX_CTRL2 field descriptions (continued)

Field	Description
	0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
17 CH1_ERROR_ STATUS	Error status bit for APBX DMA Channel 1. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
16 CH0_ERROR_ STATUS	Error status bit for APBX DMA Channel 0. Valid when corresponding Error IRQ is set. 1 - AHB bus error 0 - channel early termination. 0x0 **TERMINATION** — An early termination from the device causes error IRQ. 0x1 **BUS_ERROR** — An AHB bus error causes error IRQ.
15 CH15_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 15. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
14 CH14_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 14. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
13 CH13_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 13. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
12 CH12_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 12. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
11 CH11_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 11. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
10 CH10_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 10. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
9 CH9_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 9. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
8 CH8_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 8. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
7 CH7_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 7. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
6 CH6_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 6. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

HW_APBX_CTRL2 field descriptions (continued)

Field	Description
5 CH5_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 5. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
4 CH4_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 4. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
3 CH3_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 3. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
2 CH2_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 2. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
1 CH1_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 1. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.
0 CH0_ERROR_ IRQ	Error interrupt status bit for APBX DMA Channel 0. This sticky bit is set by DMA hardware and reset by software. It is ORed with the corresponding cmdcmplt irq to generate an irq to ARM.

7.5.4 AHB to APBX Bridge Channel Register (HW_APBX_CHANNEL_CTRL)

The APBX CHANNEL CTRL provides reset/freeze control of each DMA channel.

HW_APBX_CHANNEL_CTRL: 0x030

HW_APBX_CHANNEL_CTRL_SET: 0x034

HW_APBX_CHANNEL_CTRL_CLR: 0x038

HW_APBX_CHANNEL_CTRL_TOG: 0x03C

This register contains individual channel reset/freeze bits.

Address: HW_APBX_CHANNEL_CTRL – 8002_4000h base + 30h offset = 8002_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RESET_CHANNEL																FREEZE_CHANNEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CHANNEL_CTRL field descriptions

Field	Description
31–16 RESET_ CHANNEL	Setting a bit in this field causes the DMA controller to take the corresponding channel through its reset state. The bit is reset after the channel resources are cleared. 0x0001 **UART4_RX** — 0x0002 **UART4_TX** — 0x0004 **SPDIF_TX** — 0x0010 **SAIF0** — 0x0020 **SAIF1** — 0x0040 **I2C0** — 0x0080 **I2C1** — 0x0100 **UART0_RX** — 0x0200 **UART0_TX** — 0x0400 **UART1_RX** — 0x0800 **UART1_TX** — 0x1000 **UART2_RX** — 0x2000 **UART2_TX** — 0x4000 **UART3_RX** — 0x8000 **UART3_TX** —
15–0 FREEZE_ CHANNEL	Setting a bit in this field will freeze the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When frozen, the channel is deined access to the central DMA resources. Note: 1. DMA PIO write to associated peripheral is not supported when Freeze bit is to set, and use ARM instead to configure peripheral. 2. After FREEZE bit is set, no more access, neither AHB access to memory nor APB access to peripherals, will be allowed by arbiter. But, there might be on-going channel access exactly when FREEZE bit is set, either INCR8/INCR4/SINGLE AHB access or APB peripheral access, this on-going access will not be affected by FREEZE bit and will finish as normal. That is to say, setting FREEZE bit might not freeze channel access immediately, it only freezes further channel access, and you have to wait a while to freeze channel access completely. To make sure that there is no more access from freezed channel, channel state machine should be checked by reading channel DEBUG1 register, wait till state stunk at any of IDLE, READ_REQ, WRITE, or CHAIN_WAIT. 0x0001 **UART4_RX** — 0x0002 **UART4_TX** — 0x0004 **SPDIF_TX** — 0x0010 **SAIF0** — 0x0020 **SAIF1** — 0x0040 **I2C0** — 0x0080 **I2C1** — 0x0100 **UART0_RX** — 0x0200 **UART0_TX** — 0x0400 **UART1_RX** — 0x0800 **UART1_TX** — 0x1000 **UART2_RX** — 0x2000 **UART2_TX** — 0x4000 **UART3_RX** — 0x8000 **UART3_TX** —

7.5.5 AHB to APBX DMA Device Assignment Register (HW_APBX_DEVSEL)

This register allows reassignment of the APBX device connected to the DMA Channels.

HW_APBX_DEVSEL: 0x040

HW_APBX_DEVSEL_SET: 0x044

HW_APBX_DEVSEL_CLR: 0x048

HW_APBX_DEVSEL_TOG: 0x04C

In this chip, apbxdma channel resource is enough for peripherals, so this register is of no use and reserved.

Address: HW_APBX_DEVSEL – 8002_4000h base + 40h offset = 8002_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CH15		CH14		CH13		CH12		CH11		CH10		CH9		CH8		CH7		CH6		CH5		CH4		CH3		CH2		CH1		CH0	
W	▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨		▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_DEVSEL field descriptions

Field	Description
31–30 CH15	Reserved.
29–28 CH14	Reserved.
27–26 CH13	Reserved.
25–24 CH12	Reserved.
23–22 CH11	Reserved.
21–20 CH10	Reserved.
19–18 CH9	Reserved.
17–16 CH8	Reserved.
15–14 CH7	Reserved.
13–12 CH6	Reserved.

HW_APBX_DEVSEL field descriptions (continued)

Field	Description
11–10 CH5	Reserved.
9–8 CH4	Reserved.
7–6 CH3	Reserved.
5–4 CH2	Reserved.
3–2 CH1	Reserved.
1–0 CH0	Reserved.

7.5.6 APBX DMA Channel 0 Current Command Address Register (HW_APBX_CH0_CURCMDAR)

The APBX DMA Channel 0 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 0 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (hw_apbx_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(0);              // read the whole
register, since there is only one field
pCurCmd = (hw_apbx_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 0, CMD_ADDR);  // or, use
multi-register bitfield read macro
pCurCmd = (hw_apbx_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(0).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: HW_APBX_CH0_CURCMDAR – 8002_4000h base + 100h offset = 8002_4100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH0_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 0.

7.5.7 APBX DMA Channel 0 Next Command Address Register (HW_APBX_CH0_NXTCMDAR)

The APBX DMA Channel 0 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 0 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
HW_APBX_CHn_NXTCMDAR_WR(0, (reg32_t) pCommandTwoStructure);        // write the entire register,
 since there is only one field
BF_WRn(APBX_CHn_NXTCMDAR, 0, (reg32_t) pCommandTwoStructure);       // or, use multi-register
 bitfield write macro
HW_APBX_CHn_NXTCMDAR(0).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
 of indexed register's struct
```

Address: HW_APBX_CH0_NXTCMDAR – 8002_4000h base + 110h offset = 8002_4110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH0_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 0.

7.5.8 APBX DMA Channel 0 Command Register (HW_APBX_CH0_CMD)

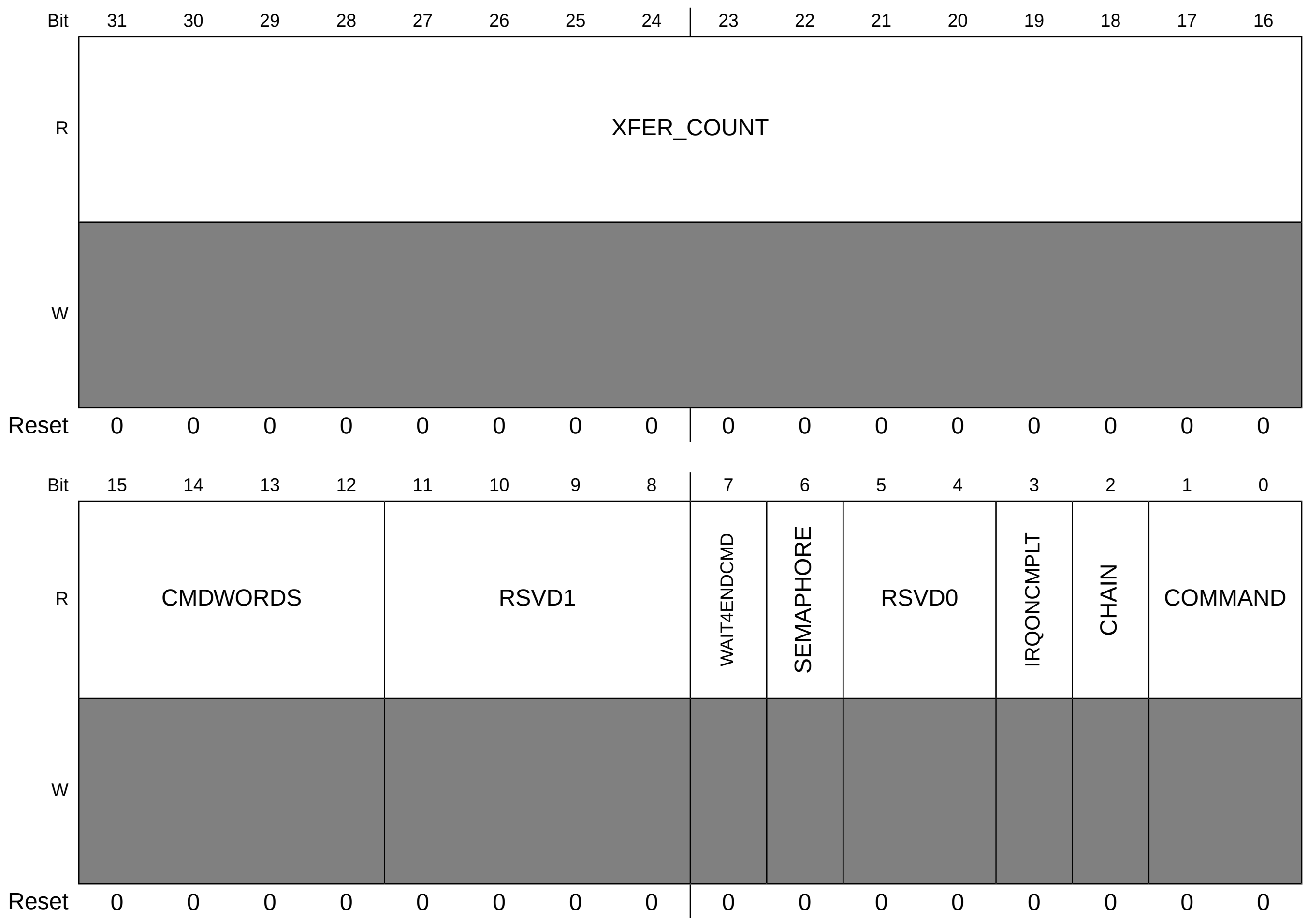
The APBX DMA Channel 0 command register specifies the DMA transaction to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

EXAMPLE

```
hw_apbx_chn_cmd_t  dma_cmd;
dma_cmd.XFER_COUNT = 512;                                 // transfer 512 bytes
dma_cmd.COMMAND = BV_APBX_CHn_CMD_COMMAND__DMA_WRITE;  // transfer to system memory from
peripheral device
dma_cmd.CHAIN = 1;                                     // chain an additional command structure
 on to the list
dma_cmd.IRQONCMPLT = 1;                                  // generate an interrupt on completion
 of this command structure
```

Address: HW_APBX_CH0_CMD – 8002_4000h base + 120h offset = 8002_4120h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
XFER_COUNT
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
CMDWORDS
RSVD1
WAIT4ENDCMD
SEMAPHORE
RSVD0
IRQONCMPLT
CHAIN
COMMAND
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH0_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART4 device HW_UARTAPP_DATA. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART, starting with the base PIO address of the UART4 (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words

HW_APBX_CH0_CMD field descriptions (continued)

Field	Description
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the ABPX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH0_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.9 APBX DMA Channel 0 Buffer Address Register (HW_APBX_CH0_BAR)

The APBX DMA Channel 0 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
hw_apbx_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: HW_APBX_CH0_BAR – 8002_4000h base + 130h offset = 8002_4130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH0_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.10 APBX DMA Channel 0 Semaphore Register (HW_APBX_CH0_SEMA)

The APBX DMA Channel 0 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBX_CHn_SEMA, 0, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 0, PHORE);  // get instantaneous value
```

Address: HW_APBX_CH0_SEMA – 8002_4000h base + 140h offset = 8002_4140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH0_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.

HW_APBX_CH0_SEMA field descriptions (continued)

Field	Description
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

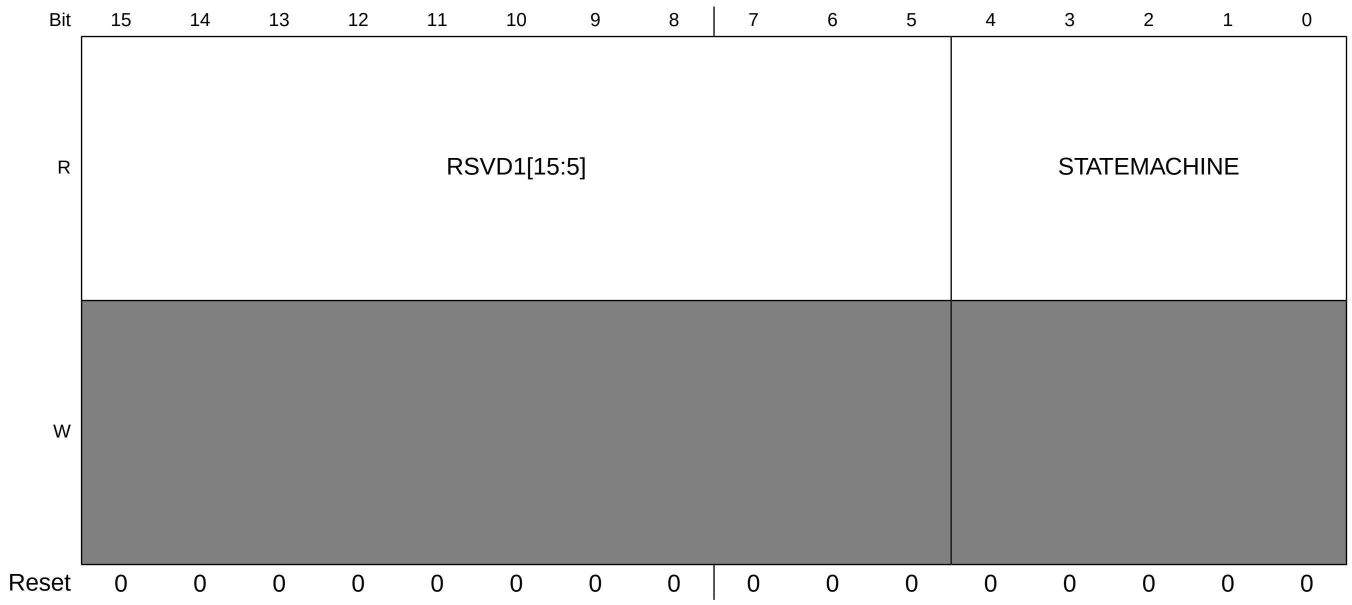
7.5.11 AHB to APBX DMA Channel 0 Debug Information (HW_APBX_CH0_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 0 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 0.

Address: HW_APBX_CH0_DEBUG1 – 8002_4000h base + 150h offset = 8002_4150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W	▨	▨	▨	▨	▨			▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0


Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSVD1[15:5]
STATEMACHINE
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH0_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 0 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command.

HW_APBX_CH0_DEBUG1 field descriptions (continued)

Field	Description
	0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.12 AHB to APBX DMA Channel 0 Debug Information (HW_APBX_CH0_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 0.

This register allows debug visibility of the APBX DMA Channel 0.

Address: HW_APBX_CH0_DEBUG2 – 8002_4000h base + 160h offset = 8002_4160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH0_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.13 APBX DMA Channel 1 Current Command Address Register (HW_APBX_CH1_CURCMDAR)

The APBX DMA Channel 1 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 1 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

EXAMPLE

```
pCurCmd = (hw_apbx_chn_cmd_t *) HW_APBX_CHn_CURCMDAR_RD(1);              // read the whole
register, since there is only one field
pCurCmd = (hw_apbx_chn_cmd_t *) BF_RDn(APBX_CHn_CURCMDAR, 1, CMD_ADDR);  // or, use
multi-register bitfield read macro
pCurCmd = (hw_apbx_chn_cmd_t *) HW_APBX_CHn_CURCMDAR(1).CMD_ADDR;        // or, assign from
bitfield of indexed register's struct
```

Address: HW_APBX_CH1_CURCMDAR – 8002_4000h base + 170h offset = 8002_4170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 1.

7.5.14 APBX DMA Channel 1 Next Command Address Register (HW_APBX_CH1_NXTCMDAR)

The APBX DMA Channel 1 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 1 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 1 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

EXAMPLE

```
HW_APBX_CHn_NXTCMDAR_WR(1, (reg32_t) pCommandTwoStructure);        // write the entire register,
 since there is only one field
BF_WRn(APBX_CHn_NXTCMDAR, 1, (reg32_t) pCommandTwoStructure);       // or, use multi-register
 bitfield write macro
HW_APBX_CHn_NXTCMDAR(1).CMD_ADDR = (reg32_t) pCommandTwoStructure;  // or, assign to bitfield
 of indexed register's struct
```

Address: HW_APBX_CH1_NXTCMDAR – 8002_4000h base + 180h offset = 8002_4180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_NXTCMDAR field descriptions

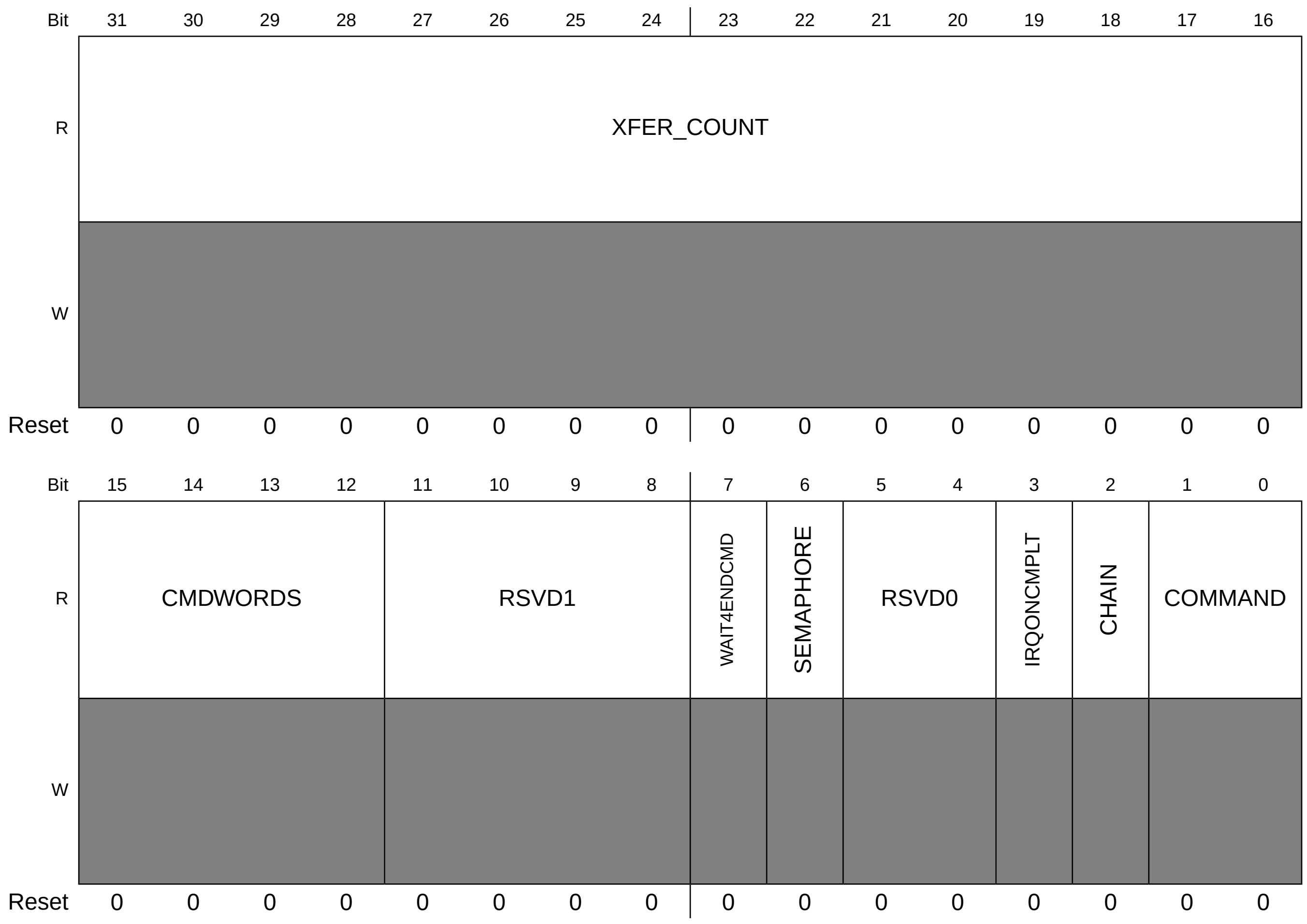
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 1.

7.5.15 APBX DMA Channel 1 Command Register (HW_APBX_CH1_CMD)

The APBX DMA Channel 1 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH1_CMD – 8002_4000h base + 190h offset = 8002_4190h



HW_APBX_CH1_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART4 device HW_UARTAPP_DATA. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the DAC, starting with the base PIO address of the UART4 (HW_UARTAPP_CTRL1) and increment from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.

HW_APBX_CH1_CMD field descriptions (continued)

Field	Description
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH1_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.16 APBX DMA Channel 1 Buffer Address Register (HW_APBX_CH1_BAR)

The APBX DMA Channel 1 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

EXAMPLE

```
hw_apbx_chn_bar_t  dma_data;
dma_data.ADDRESS = (reg32_t) pDataBuffer;
```

Address: HW_APBX_CH1_BAR – 8002_4000h base + 1A0h offset = 8002_41A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.17 APBX DMA Channel 1 Semaphore Register (HW_APBX_CH1_SEMA)

The APBX DMA Channel 1 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

EXAMPLE

```
BF_WR(APBX_CHn_SEMA, 1, INCREMENT_SEMA, 2);     // increment semaphore by two
current_sema = BF_RD(APBX_CHn_SEMA, 1, PHORE);  // get instantaneous value
```

Address: HW_APBX_CH1_SEMA – 8002_4000h base + 1B0h offset = 8002_41B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.18 AHB to APBX DMA Channel 1 Debug Information (HW_APBX_CH1_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 1 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 1.

Address: HW_APBX_CH1_DEBUG1 – 8002_4000h base + 1C0h offset = 8002_41C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 1 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.

HW_APBX_CH1_DEBUG1 field descriptions (continued)

Field	Description
	0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.19 AHB to APBX DMA Channel 1 Debug Information (HW_APBX_CH1_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 1.

This register allows debug visibility of the APBX DMA Channel 1.

Address: HW_APBX_CH1_DEBUG2 – 8002_4000h base + 1D0h offset = 8002_41D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH1_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.20 APBX DMA Channel 2 Current Command Address Register (HW_APBX_CH2_CURCMDAR)

The APBX DMA Channel 2 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 2 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH2_CURCMDAR – 8002_4000h base + 1E0h offset = 8002_41E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 2.

7.5.21 APBX DMA Channel 2 Next Command Address Register (HW_APBX_CH2_NXTCMDAR)

The APBX DMA Channel 2 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 2 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 0 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH2_NXTCMDAR – 8002_4000h base + 1F0h offset = 8002_41F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_NXTCMDAR field descriptions

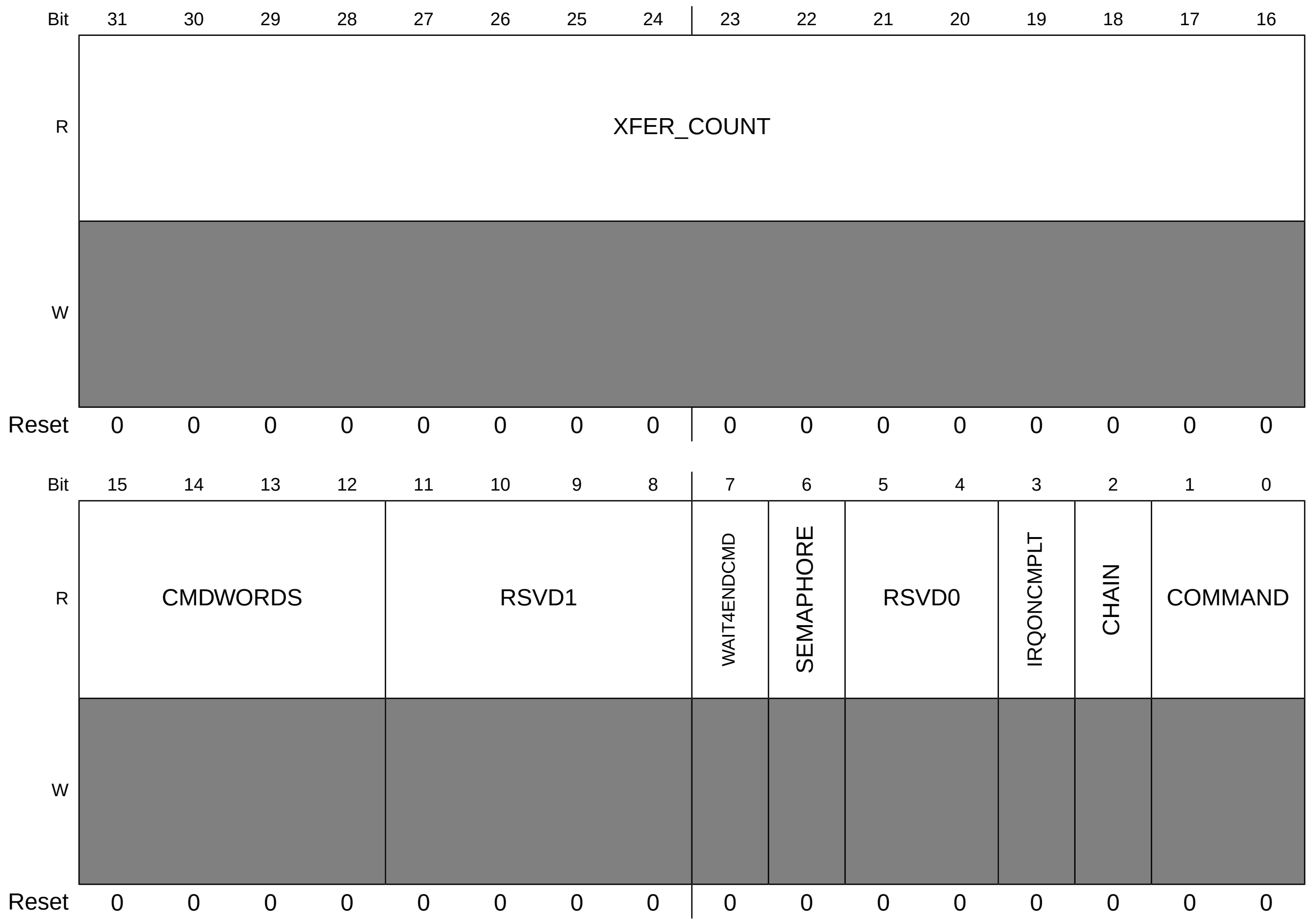
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 2.

7.5.22 APBX DMA Channel 2 Command Register (HW_APBX_CH2_CMD)

The APBX DMA Channel 2 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH2_CMD – 8002_4000h base + 200h offset = 8002_4200h



HW_APBX_CH2_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SPDIF device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the SPDIF, starting with the base PIO address of the SPDIF and incrementing from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.

HW_APBX_CH2_CMD field descriptions (continued)

Field	Description
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH2_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.23 APBX DMA Channel 2 Buffer Address Register (HW_APBX_CH2_BAR)

The APBX DMA Channel 2 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH2_BAR – 8002_4000h base + 210h offset = 8002_4210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.24 APBX DMA Channel 2 Semaphore Register (HW_APBX_CH2_SEMA)

The APBX DMA Channel 2 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH2_SEMA – 8002_4000h base + 220h offset = 8002_4220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.25 AHB to APBX DMA Channel 2 Debug Information (HW_APBX_CH2_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 2 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 2.

Address: HW_APBX_CH2_DEBUG1 – 8002_4000h base + 230h offset = 8002_4230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device

HW_APBX_CH2_DEBUG1 field descriptions (continued)

Field	Description
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 2 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.26 AHB to APBX DMA Channel 2 Debug Information (HW_APBX_CH2_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 2.

This register allows debug visibility of the APBX DMA Channel 2.

Address: HW_APBX_CH2_DEBUG2 – 8002_4000h base + 240h offset = 8002_4240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH2_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.27 APBX DMA Channel 3 Current Command Address Register (HW_APBX_CH3_CURCMDAR)

The APBX DMA Channel 3 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_CURCMDAR – 8002_4000h base + 250h offset = 8002_4250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 3.

7.5.28 APBX DMA Channel 3 Next Command Address Register (HW_APBX_CH3_NXTCMDAR)

The APBX DMA Channel 3 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_NXTCMDAR – 8002_4000h base + 260h offset = 8002_4260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 3.

7.5.29 APBX DMA Channel 3 Command Register (HW_APBX_CH3_CMD)

The APBX DMA Channel 3 command register specifies the cycle to perform for the current command chain item.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_CMD – 8002_4000h base + 270h offset = 8002_4270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in assigned peripheral. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to assigned peripheral, starting with the base PIO address of the assigned peripheral and increment from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

HW_APBX_CH3_CMD field descriptions (continued)

Field	Description
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH3_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.30 APBX DMA Channel 3 Buffer Address Register (HW_APBX_CH3_BAR)

The APBX DMA Channel 3 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_BAR – 8002_4000h base + 280h offset = 8002_4280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.31 APBX DMA Channel 3 Semaphore Register (HW_APBX_CH3_SEMA)

The APBX DMA Channel 3 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH3_SEMA – 8002_4000h base + 290h offset = 8002_4290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_SEMA field descriptions

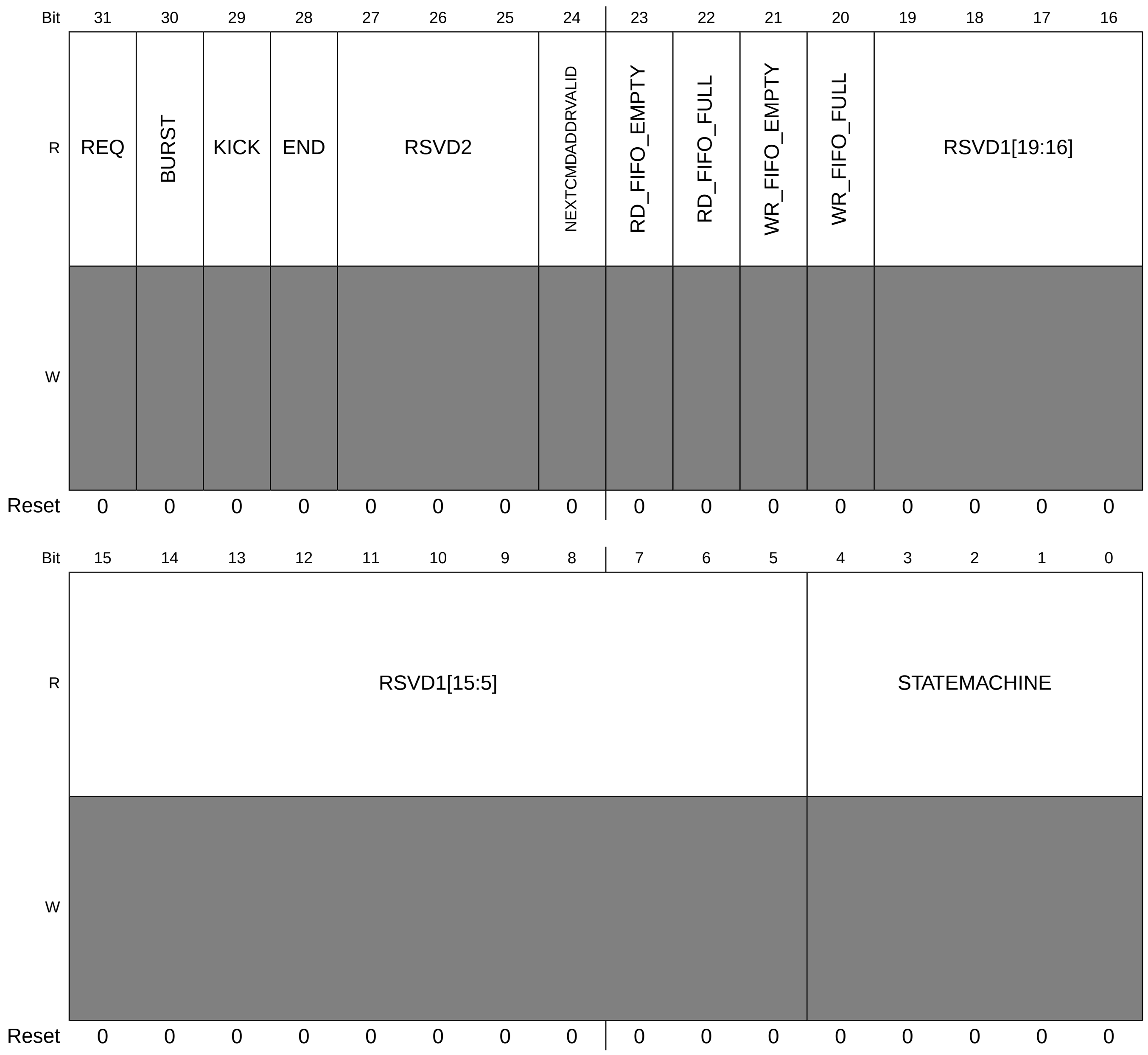
Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.32 AHB to APBX DMA Channel 3 Debug Information (HW_APBX_CH3_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 3 state machine and controls. Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_DEBUG1 – 8002_4000h base + 2A0h offset = 8002_42A0h



HW_APBX_CH3_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBX_CH3_DEBUG1 field descriptions (continued)

Field	Description
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 3 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.33 AHB to APBX DMA Channel 3 Debug Information (HW_APBX_CH3_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 3.

Because channel 3 is not assigned to any peripheral, this register is of no practical usage and just reserved for compatibility.

Address: HW_APBX_CH3_DEBUG2 – 8002_4000h base + 2B0h offset = 8002_42B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH3_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.34 APBX DMA Channel 4 Current Command Address Register (HW_APBX_CH4_CURCMDAR)

The APBX DMA Channel 4 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 4 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH4_CURCMDAR – 8002_4000h base + 2C0h offset = 8002_42C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 4.

7.5.35 APBX DMA Channel 4 Next Command Address Register (HW_APBX_CH4_NXTCMDAR)

The APBX DMA Channel 4 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 4 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 4 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH4_NXTCMDAR – 8002_4000h base + 2D0h offset = 8002_42D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_NXTCMDAR field descriptions

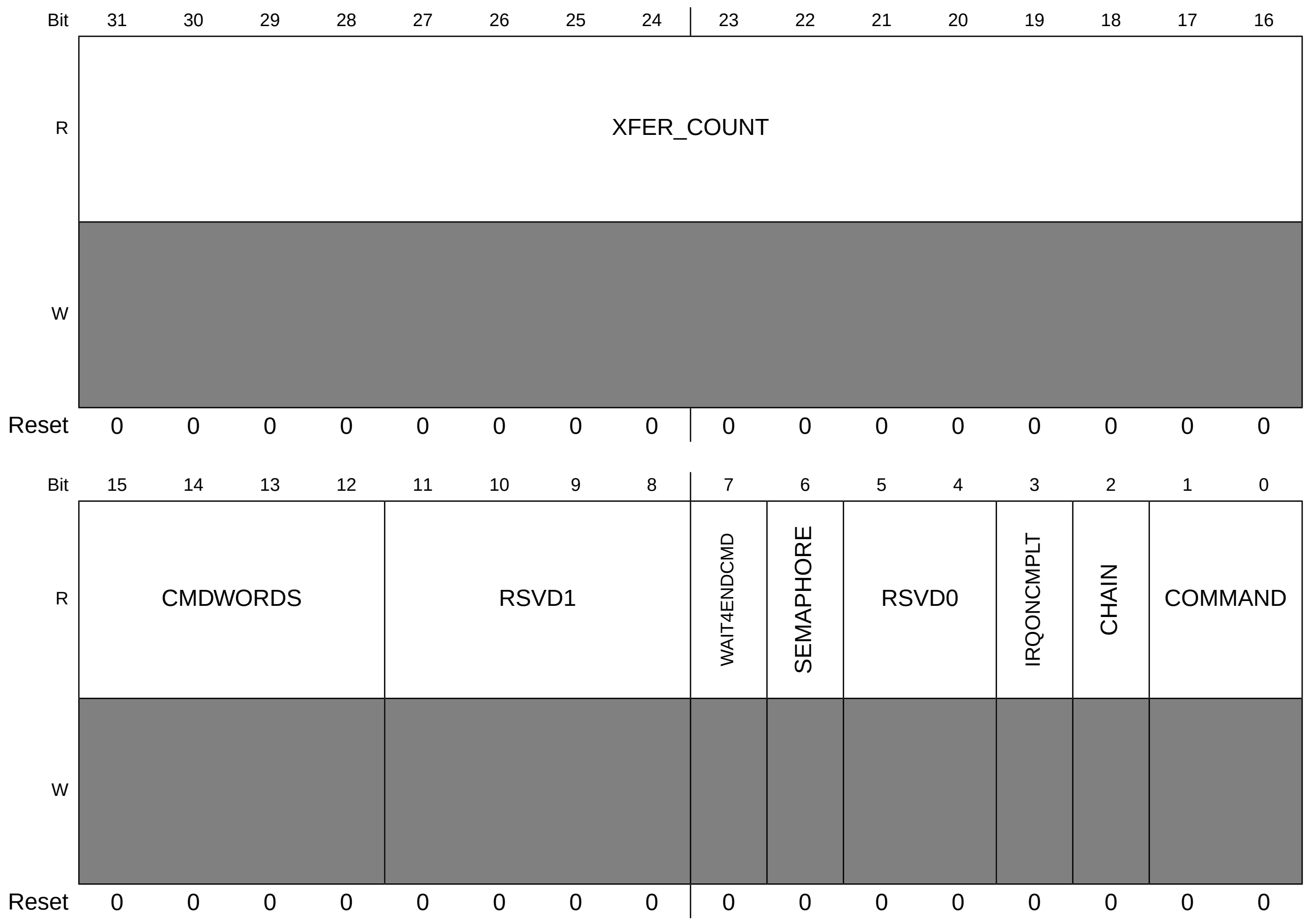
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 4.

7.5.36 APBX DMA Channel 4 Command Register (HW_APBX_CH4_CMD)

The APBX DMA Channel 4 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH4_CMD – 8002_4000h base + 2E0h offset = 8002_42E0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
XFER_COUNT
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
CMDWORDS
RSVD1
WAIT4ENDCMD
SEMAPHORE
RSVD0
IRQONCMPLT
CHAIN
COMMAND
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH4_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the SAIF0 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the SAIF0. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

HW_APBX_CH4_CMD field descriptions (continued)

Field	Description
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH4_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.37 APBX DMA Channel 4 Buffer Address Register (HW_APBX_CH4_BAR)

The APBX DMA Channel 4 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device associate with this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH4_BAR – 8002_4000h base + 2F0h offset = 8002_42F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.38 APBX DMA Channel 4 Semaphore Register (HW_APBX_CH4_SEMA)

The APBX DMA Channel 4 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH4_SEMA – 8002_4000h base + 300h offset = 8002_4300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.39 AHB to APBX DMA Channel 4 Debug Information (HW_APBX_CH4_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 4 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 4.

Address: HW_APBX_CH4_DEBUG1 – 8002_4000h base + 310h offset = 8002_4310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END		RSVD2		NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL		RSVD1[19:16]		
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R						RSVD1[15:5]								STATEMACHINE		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBX_CH4_DEBUG1 field descriptions (continued)

Field	Description
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 4 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.40 AHB to APBX DMA Channel 4 Debug Information (HW_APBX_CH4_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 4.

This register allows debug visibility of the APBX DMA Channel 4.

Address: HW_APBX_CH4_DEBUG2 – 8002_4000h base + 320h offset = 8002_4320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH4_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.41 APBX DMA Channel 5 Current Command Address Register (HW_APBX_CH5_CURCMDAR)

The APBX DMA Channel 5 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 5 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH5_CURCMDAR – 8002_4000h base + 330h offset = 8002_4330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH5_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 5.

7.5.42 APBX DMA Channel 5 Next Command Address Register (HW_APBX_CH5_NXTCMDAR)

The APBX DMA Channel 5 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 5 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 5 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH5_NXTCMDAR – 8002_4000h base + 340h offset = 8002_4340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH5_NXTCMDAR field descriptions

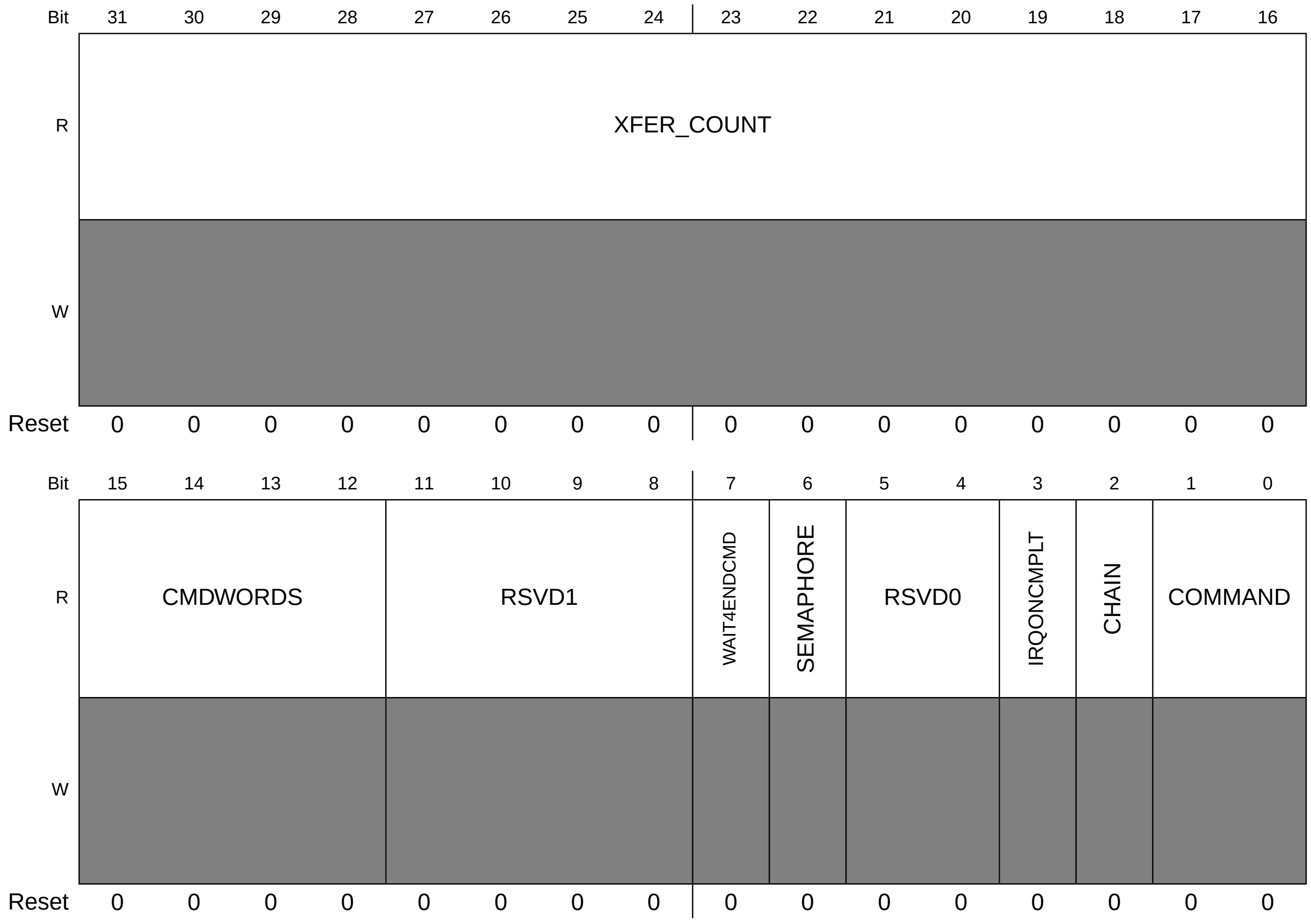
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 5.

7.5.43 APBX DMA Channel 5 Command Register (HW_APBX_CH5_CMD)

The APBX DMA Channel 5 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH5_CMD – 8002_4000h base + 350h offset = 8002_4350h



HW_APBX_CH5_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the SAIF1 register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the SAIF1. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.

HW_APBX_CH5_CMD field descriptions (continued)

Field	Description
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH5_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.44 APBX DMA Channel 5 Buffer Address Register (HW_APBX_CH5_BAR)

The APBX DMA Channel 5 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH5_BAR – 8002_4000h base + 360h offset = 8002_4360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH5_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.45 APBX DMA Channel 5 Semaphore Register (HW_APBX_CH5_SEMA)

The APBX DMA Channel 5 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH5_SEMA – 8002_4000h base + 370h offset = 8002_4370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH5_SEMA field descriptions

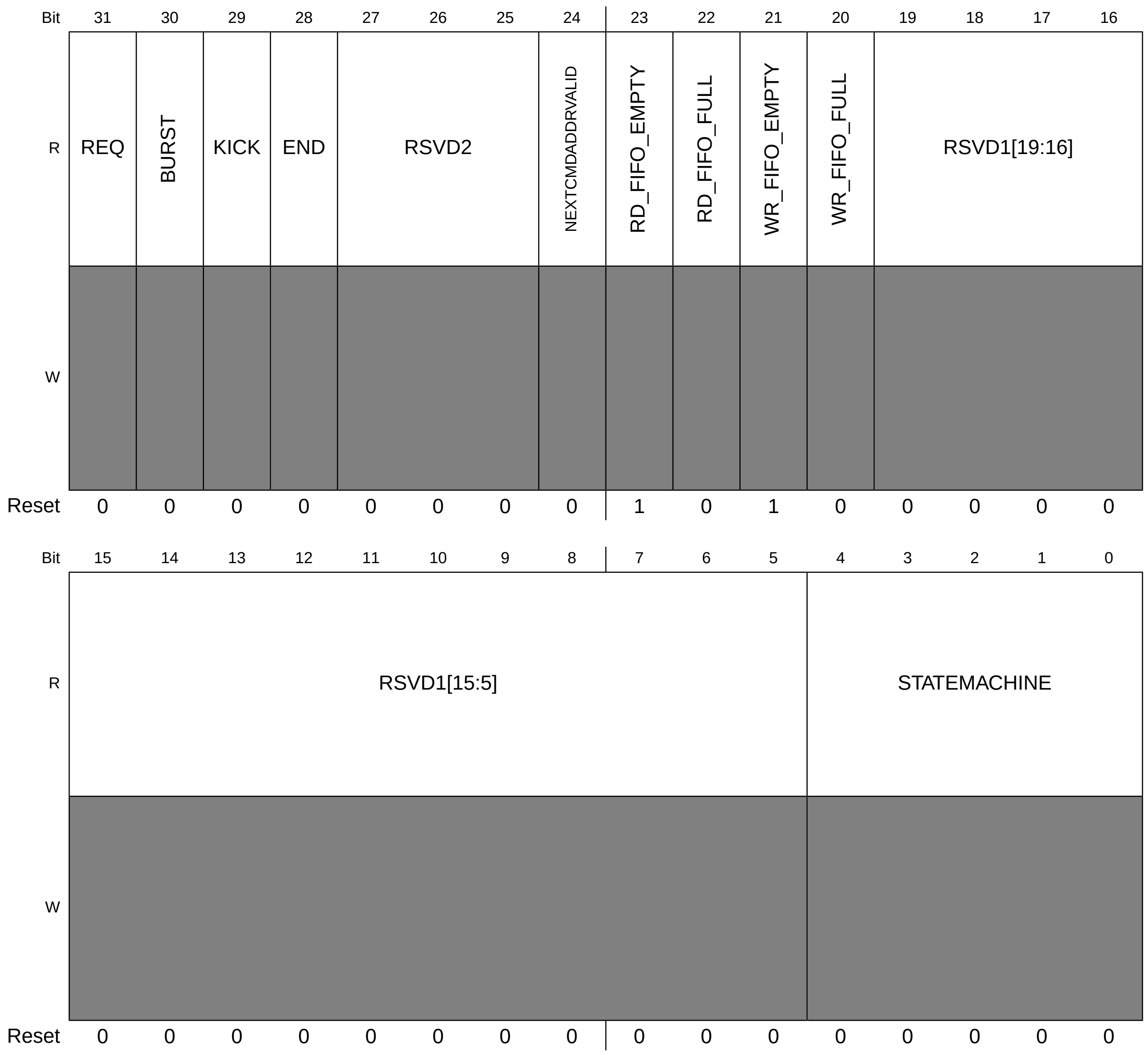
Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.46 AHB to APBX DMA Channel 5 Debug Information (HW_APBX_CH5_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 5 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 5.

Address: HW_APBX_CH5_DEBUG1 – 8002_4000h base + 380h offset = 8002_4380h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
REQ
BURST
KICK
END
RSVD2
NEXTCMDADDRVALID
RD_FIFO_EMPTY
RD_FIFO_FULL
WR_FIFO_EMPTY
WR_FIFO_FULL
RSVD1[19:16]
W
Reset 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSVD1[15:5]
STATEMACHINE
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH5_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBX_CH5_DEBUG1 field descriptions (continued)

Field	Description
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 5 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.47 AHB to APBX DMA Channel 5 Debug Information (HW_APBX_CH5_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 5.

This register allows debug visibility of the APBX DMA Channel 5.

Address: HW_APBX_CH5_DEBUG2 – 8002_4000h base + 390h offset = 8002_4390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH5_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.48 APBX DMA Channel 6 Current Command Address Register (HW_APBX_CH6_CURCMDAR)

The APBX DMA Channel 6 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 6 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH6_CURCMDAR – 8002_4000h base + 3A0h offset = 8002_43A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 6.

7.5.49 APBX DMA Channel 6 Next Command Address Register (HW_APBX_CH6_NXTCMDAR)

The APBX DMA Channel 6 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 6 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 6 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH6_NXTCMDAR – 8002_4000h base + 3B0h offset = 8002_43B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_NXTCMDAR field descriptions

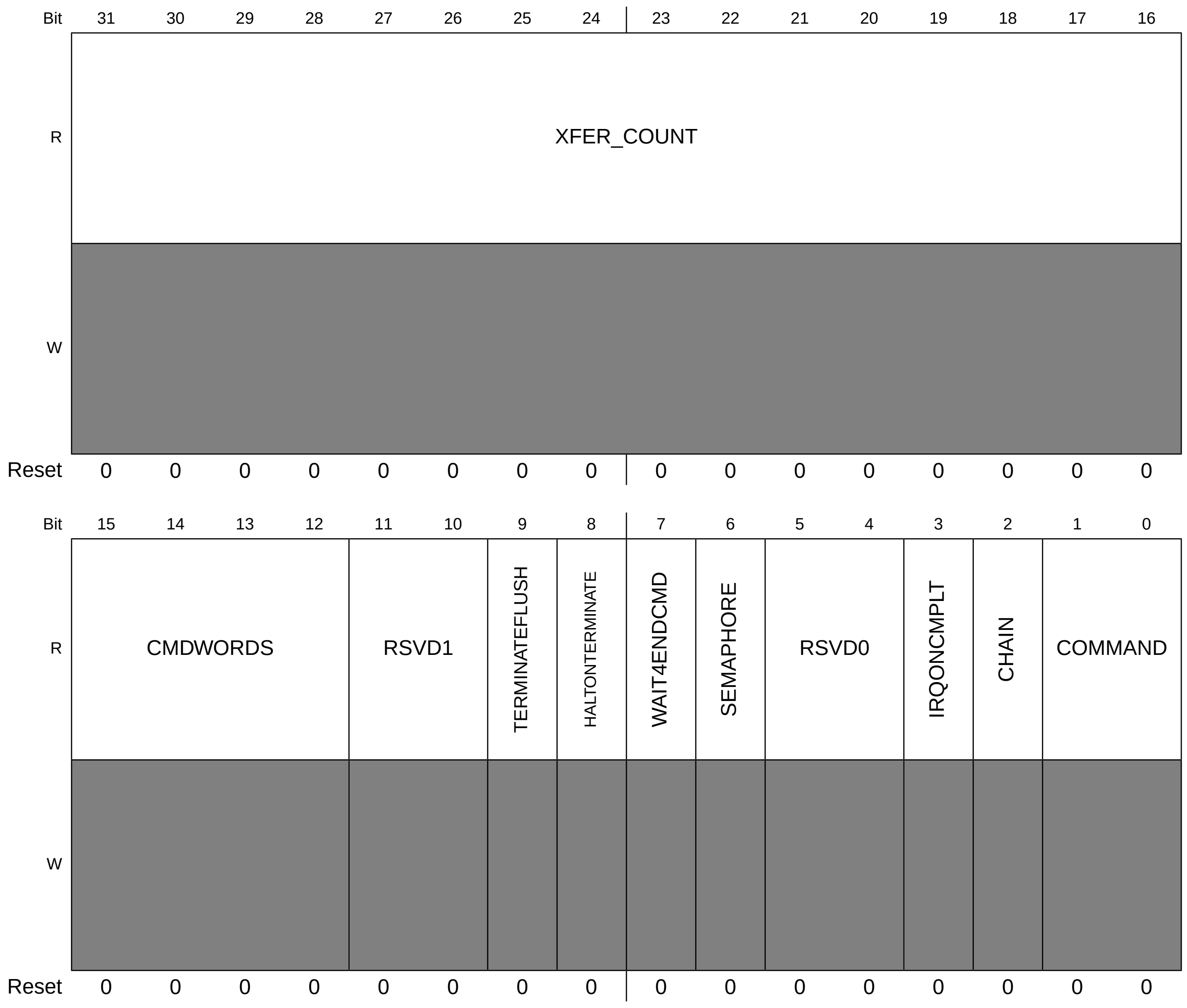
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 6.

7.5.50 APBX DMA Channel 6 Command Register (HW_APBX_CH6_CMD)

The APBX DMA Channel 6 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH6_CMD – 8002_4000h base + 3C0h offset = 8002_43C0h



HW_APBX_CH6_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the I2C0 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the I2C0 device. Zero means transfer NO command words
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.

HW_APBX_CH6_CMD field descriptions (continued)

Field	Description
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH6_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.51 APBX DMA Channel 6 Buffer Address Register (HW_APBX_CH6_BAR)

The APBX DMA Channel 6 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH6_BAR – 8002_4000h base + 3D0h offset = 8002_43D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.52 APBX DMA Channel 6 Semaphore Register (HW_APBX_CH6_SEMA)

The APBX DMA Channel 6 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH6_SEMA – 8002_4000h base + 3E0h offset = 8002_43E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.53 AHB to APBX DMA Channel 6 Debug Information (HW_APBX_CH6_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 6 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 6.

Address: HW_APBX_CH6_DEBUG1 – 8002_4000h base + 3F0h offset = 8002_43F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 6 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.

HW_APBX_CH6_DEBUG1 field descriptions (continued)

Field	Description
	0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.54 AHB to APBX DMA Channel 6 Debug Information (HW_APBX_CH6_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 6.

This register allows debug visibility of the APBX DMA Channel 6.

Address: HW_APBX_CH6_DEBUG2 – 8002_4000h base + 400h offset = 8002_4400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH6_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.55 APBX DMA Channel 7 Current Command Address Register (HW_APBX_CH7_CURCMDAR)

The APBX DMA Channel 7 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 7 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH7_CURCMDAR – 8002_4000h base + 410h offset = 8002_4410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for channel 7.

7.5.56 APBX DMA Channel 7 Next Command Address Register (HW_APBX_CH7_NXTCMDAR)

The APBX DMA Channel 7 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 7 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 7 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH7_NXTCMDAR – 8002_4000h base + 420h offset = 8002_4420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for channel 7.

7.5.57 APBX DMA Channel 7 Command Register (HW_APBX_CH7_CMD)

The APBX DMA Channel 7 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH7_CMD – 8002_4000h base + 430h offset = 8002_4430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1		TERMINATEFLUSH	HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in I2C1 device. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the I2C1 device. Zero means transfer NO command words

HW_APBX_CH7_CMD field descriptions (continued)

Field	Description
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH7_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.58 APBX DMA Channel 7 Buffer Address Register (HW_APBX_CH7_BAR)

The APBX DMA Channel 7 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH7_BAR – 8002_4000h base + 440h offset = 8002_4440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.59 APBX DMA Channel 7 Semaphore Register (HW_APBX_CH7_SEMA)

The APBX DMA Channel 7 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH7_SEMA – 8002_4000h base + 450h offset = 8002_4450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.

HW_APBX_CH7_SEMA field descriptions (continued)

Field	Description
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.60 AHB to APBX DMA Channel 7 Debug Information (HW_APBX_CH7_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 7 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 7.

Address: HW_APBX_CH7_DEBUG1 – 8002_4000h base + 460h offset = 8002_4460h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 7 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command.

HW_APBX_CH7_DEBUG1 field descriptions (continued)

Field	Description
	0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.61 AHB to APBX DMA Channel 7 Debug Information (HW_APBX_CH7_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 7.

This register allows debug visibility of the APBX DMA Channel 7.

Address: HW_APBX_CH7_DEBUG2 – 8002_4000h base + 470h offset = 8002_4470h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH7_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.62 APBX DMA Channel 8 Current Command Address Register (HW_APBX_CH8_CURCMDAR)

The APBX DMA Channel 8 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 8 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH8_CURCMDAR – 8002_4000h base + 480h offset = 8002_4480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 8.

7.5.63 APBX DMA Channel 8 Next Command Address Register (HW_APBX_CH8_NXTCMDAR)

The APBX DMA Channel 8 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 8 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 8 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH8_NXTCMDAR – 8002_4000h base + 490h offset = 8002_4490h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 8.

7.5.64 APBX DMA Channel 8 Command Register (HW_APBX_CH8_CMD)

The APBX DMA Channel 8 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH8_CMD – 8002_4000h base + 4A0h offset = 8002_44A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1		TERMINATEFLUSH	HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART4 device HW_UARTAPP_DATA. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART0, starting with the base PIO address of the UART0 (HW_UARTAPP_CTRL0). Zero means transfer NO command words
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH8_CMDAR to find the next command.

HW_APBX_CH8_CMD field descriptions (continued)

Field	Description
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.65 APBX DMA Channel 8 Buffer Address Register (HW_APBX_CH8_BAR)

The APBX DMA Channel 8 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH8_BAR – 8002_4000h base + 4B0h offset = 8002_44B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.66 APBX DMA Channel 8 Semaphore Register (HW_APBX_CH8_SEMA)

The APBX DMA Channel 8 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH8_SEMA – 8002_4000h base + 4C0h offset = 8002_44C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.67 AHB to APBX DMA Channel 8 Debug Information (HW_APBX_CH8_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 8 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 8.

Address: HW_APBX_CH8_DEBUG1 – 8002_4000h base + 4D0h offset = 8002_44D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBX_CH8_DEBUG1 field descriptions (continued)

Field	Description
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 8 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.68 AHB to APBX DMA Channel 8 Debug Information (HW_APBX_CH8_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 8.

This register allows debug visibility of the APBX DMA Channel 8.

Address: HW_APBX_CH8_DEBUG2 – 8002_4000h base + 4E0h offset = 8002_44E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH8_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.69 APBX DMA Channel 9 Current Command Address Register (HW_APBX_CH9_CURCMDAR)

The APBX DMA Channel 9 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 9 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH9_CURCMDAR – 8002_4000h base + 4F0h offset = 8002_44F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 9.

7.5.70 APBX DMA Channel 9 Next Command Address Register (HW_APBX_CH9_NXTCMDAR)

The APBX DMA Channel 9 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 9 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 9 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH9_NXTCMDAR – 8002_4000h base + 500h offset = 8002_4500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 9.

7.5.71 APBX DMA Channel 9 Command Register (HW_APBX_CH9_CMD)

The APBX DMA Channel 9 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH9_CMD – 8002_4000h base + 510h offset = 8002_4510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1		TERMINATEFLUSH	HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨		▨	▨	▨	▨	▨		▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART0 device HW_UARTAPP_DATA. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART0, starting with the base PIO address of the UART0 (HW_UARTAPP_CTRL1). Zero means transfer NO command words
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.

HW_APBX_CH9_CMD field descriptions (continued)

Field	Description
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH9_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.72 APBX DMA Channel 9 Buffer Address Register (HW_APBX_CH9_BAR)

The APBX DMA Channel 9 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH9_BAR – 8002_4000h base + 520h offset = 8002_4520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.73 APBX DMA Channel 9 Semaphore Register (HW_APBX_CH9_SEMA)

The APBX DMA Channel 9 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH9_SEMA – 8002_4000h base + 530h offset = 8002_4530h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.74 AHB to APBX DMA Channel 9 Debug Information (HW_APBX_CH9_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 9 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 9.

Address: HW_APBX_CH9_DEBUG1 – 8002_4000h base + 540h offset = 8002_4540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 9 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.

HW_APBX_CH9_DEBUG1 field descriptions (continued)

Field	Description
	0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.75 AHB to APBX DMA Channel 9 Debug Information (HW_APBX_CH9_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 9.

This register allows debug visibility of the APBX DMA Channel 9.

Address: HW_APBX_CH9_DEBUG2 – 8002_4000h base + 550h offset = 8002_4550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH9_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.76 APBX DMA Channel 10 Current Command Address Register (HW_APBX_CH10_CURCMDAR)

The APBX DMA Channel 10 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 10 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH10_CURCMDAR – 8002_4000h base + 560h offset = 8002_4560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 10.

7.5.77 APBX DMA Channel 10 Next Command Address Register (HW_APBX_CH10_NXTCMDAR)

The APBX DMA Channel 10 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 10 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 10 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH10_NXTCMDAR – 8002_4000h base + 570h offset = 8002_4570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 10.

7.5.78 APBX DMA Channel 10 Command Register (HW_APBX_CH10_CMD)

The APBX DMA Channel 10 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH10_CMD – 8002_4000h base + 580h offset = 8002_4580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1		TERMINATEFLUSH	HALTONTERMINATE	WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W	▨				▨		▨	▨	▨	▨	▨		▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART1 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART1, starting with the base PIO address of the UART1 (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words

HW_APBX_CH10_CMD field descriptions (continued)

Field	Description
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH10_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.79 APBX DMA Channel 10 Buffer Address Register (HW_APBX_CH10_BAR)

The APBX DMA Channel 10 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH10_BAR – 8002_4000h base + 590h offset = 8002_4590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.80 APBX DMA Channel 10 Semaphore Register (HW_APBX_CH10_SEMA)

The APBX DMA Channel 10 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH10_SEMA – 8002_4000h base + 5A0h offset = 8002_45A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.

HW_APBX_CH10_SEMA field descriptions (continued)

Field	Description
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.81 AHB to APBX DMA Channel 10 Debug Information (HW_APBX_CH10_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 10 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 10.

Address: HW_APBX_CH10_DEBUG1 – 8002_4000h base + 5B0h offset = 8002_45B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 10 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command.

HW_APBX_CH10_DEBUG1 field descriptions (continued)

Field	Description
	0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.82 AHB to APBX DMA Channel 10 Debug Information (HW_APBX_CH10_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 10.

This register allows debug visibility of the APBX DMA Channel 10.

Address: HW_APBX_CH10_DEBUG2 – 8002_4000h base + 5C0h offset = 8002_45C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH10_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.83 APBX DMA Channel 11 Current Command Address Register (HW_APBX_CH11_CURCMDAR)

The APBX DMA Channel 11 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 11 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH11_CURCMDAR – 8002_4000h base + 5D0h offset = 8002_45D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 11.

7.5.84 APBX DMA Channel 11 Next Command Address Register (HW_APBX_CH11_NXTCMDAR)

The APBX DMA Channel 11 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 11 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 11 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH11_NXTCMDAR – 8002_4000h base + 5E0h offset = 8002_45E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_NXTCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 11.

7.5.85 APBX DMA Channel 11 Command Register (HW_APBX_CH11_CMD)

The APBX DMA Channel 11 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH11_CMD – 8002_4000h base + 5F0h offset = 8002_45F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	XFER_COUNT															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDWORDS				RSVD1				WAIT4ENDCMD	SEMAPHORE	RSVD0		IRQONCMPLT	CHAIN	COMMAND	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART1 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART1, starting with the base PIO address of the UART1 (HW_UARTAPP_CTRL1) and increment from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH11_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer.

HW_APBX_CH11_CMD field descriptions (continued)

Field	Description
	0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.86 APBX DMA Channel 11 Buffer Address Register (HW_APBX_CH11_BAR)

The APBX DMA Channel 11 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH11_BAR – 8002_4000h base + 600h offset = 8002_4600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.87 APBX DMA Channel 11 Semaphore Register (HW_APBX_CH11_SEMA)

The APBX DMA Channel 11 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH11_SEMA – 8002_4000h base + 610h offset = 8002_4610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.88 AHB to APBX DMA Channel 11 Debug Information (HW_APBX_CH11_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 11 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 11.

Address: HW_APBX_CH11_DEBUG1 – 8002_4000h base + 620h offset = 8002_4620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device

HW_APBX_CH11_DEBUG1 field descriptions (continued)

Field	Description
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 11 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.89 AHB to APBX DMA Channel 11 Debug Information (HW_APBX_CH11_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 11.

This register allows debug visibility of the APBX DMA Channel 11.

Address: HW_APBX_CH11_DEBUG2 – 8002_4000h base + 630h offset = 8002_4630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH11_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.90 APBX DMA Channel 12 Current Command Address Register (HW_APBX_CH12_CURCMDAR)

The APBX DMA Channel 12 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 12 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH12_CURCMDAR – 8002_4000h base + 640h offset = 8002_4640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 12.

7.5.91 APBX DMA Channel 12 Next Command Address Register (HW_APBX_CH12_NXTCMDAR)

The APBX DMA Channel 12 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 12 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 12 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH12_NXTCMDAR – 8002_4000h base + 650h offset = 8002_4650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_NXTCMDAR field descriptions

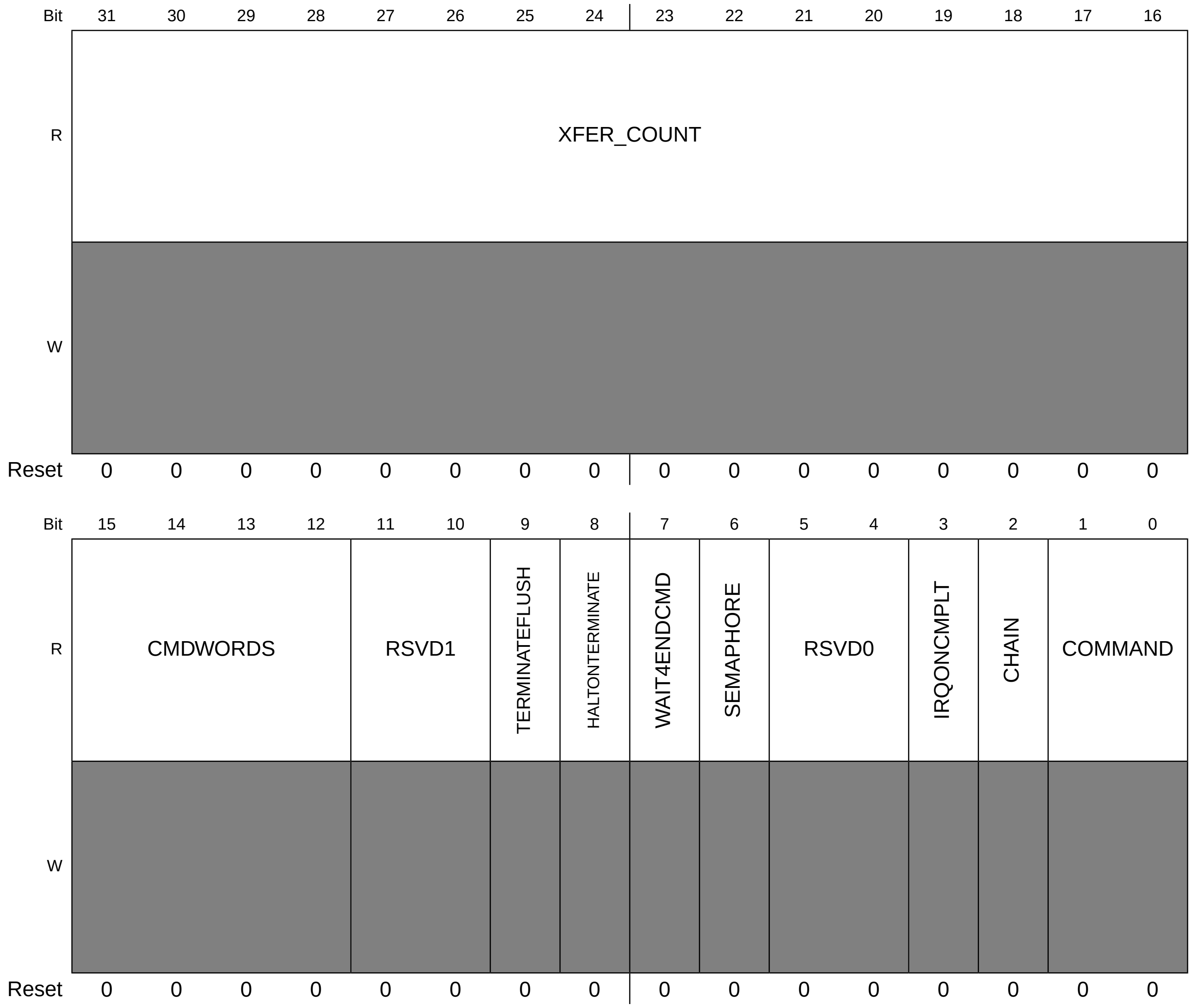
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 12.

7.5.92 APBX DMA Channel 12 Command Register (HW_APBX_CH12_CMD)

The APBX DMA Channel 12 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH12_CMD – 8002_4000h base + 660h offset = 8002_4660h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
XFER_COUNT
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
CMDWORDS
RSVD1
TERMINATEFLUSH
HALTONTERMINATE
WAIT4ENDCMD
SEMAPHORE
RSVD0
IRQONCMPLT
CHAIN
COMMAND
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH12_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART2 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART2, starting with the base PIO address of the UART2 (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.

HW_APBX_CH12_CMD field descriptions (continued)

Field	Description
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH12_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.93 APBX DMA Channel 12 Buffer Address Register (HW_APBX_CH12_BAR)

The APBX DMA Channel 12 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH12_BAR – 8002_4000h base + 670h offset = 8002_4670h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.94 APBX DMA Channel 12 Semaphore Register (HW_APBX_CH12_SEMA)

The APBX DMA Channel 12 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH12_SEMA – 8002_4000h base + 680h offset = 8002_4680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.95 AHB to APBX DMA Channel 12 Debug Information (HW_APBX_CH12_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 12 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 12.

Address: HW_APBX_CH12_DEBUG1 – 8002_4000h base + 690h offset = 8002_4690h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 12 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.

HW_APBX_CH12_DEBUG1 field descriptions (continued)

Field	Description
	0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.96 AHB to APBX DMA Channel 12 Debug Information (HW_APBX_CH12_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 12.

This register allows debug visibility of the APBX DMA Channel 12.

Address: HW_APBX_CH12_DEBUG2 – 8002_4000h base + 6A0h offset = 8002_46A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH12_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.97 APBX DMA Channel 13 Current Command Address Register (HW_APBX_CH13_CURCMDAR)

The APBX DMA Channel 13 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 13 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH13_CURCMDAR – 8002_4000h base + 6B0h offset = 8002_46B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 13.

7.5.98 APBX DMA Channel 13 Next Command Address Register (HW_APBX_CH13_NXTCMDAR)

The APBX DMA Channel 13 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 13 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 13 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH13_NXTCMDAR – 8002_4000h base + 6C0h offset = 8002_46C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_NXTCMDAR field descriptions

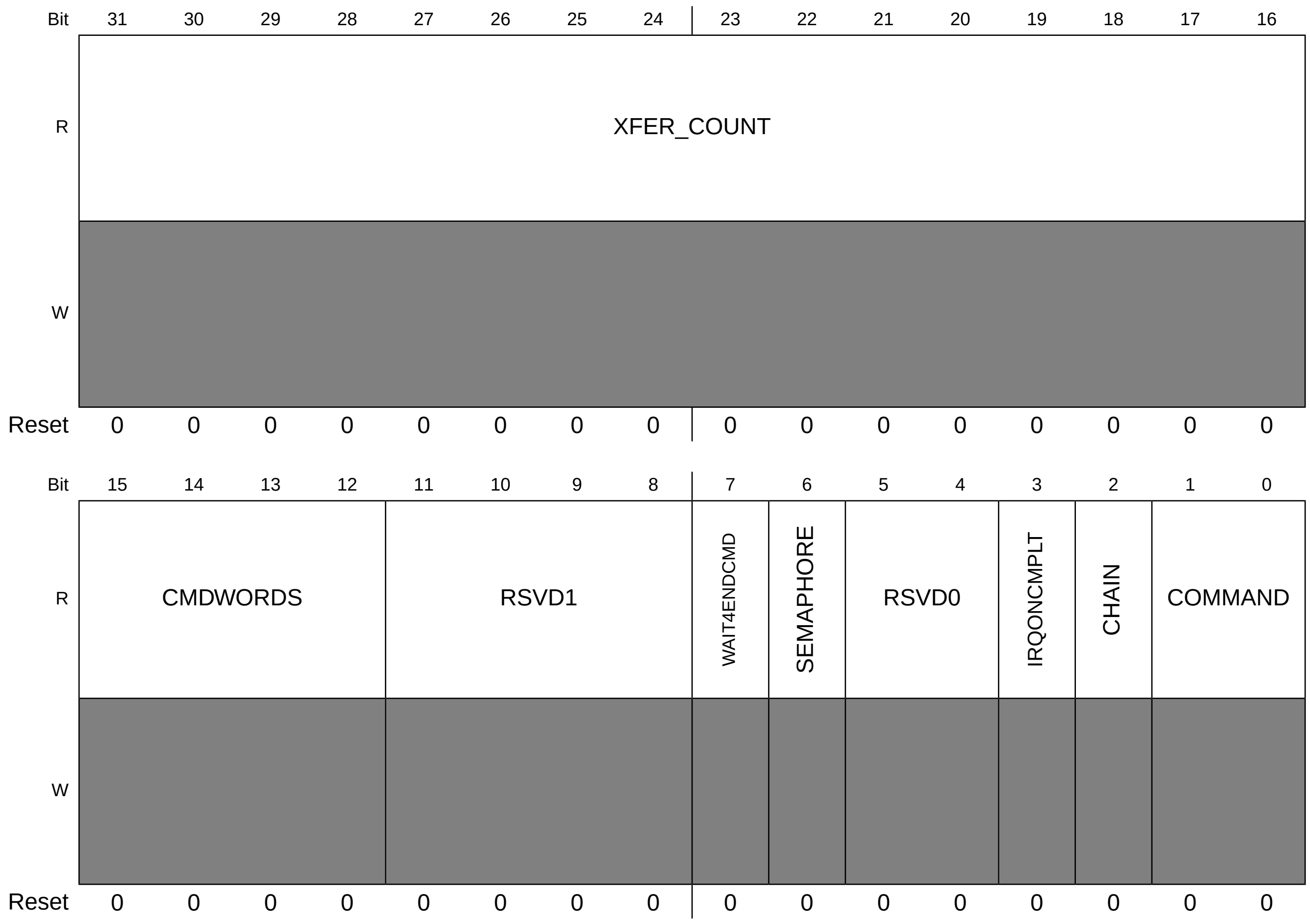
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 13.

7.5.99 APBX DMA Channel 13 Command Register (HW_APBX_CH13_CMD)

The APBX DMA Channel 13 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH13_CMD – 8002_4000h base + 6D0h offset = 8002_46D0h



HW_APBX_CH13_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART1 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART2, starting with the base PIO address of the UART2 (HW_UARTAPP_CTRL1) and increment from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.

HW_APBX_CH13_CMD field descriptions (continued)

Field	Description
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH13_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.100 APBX DMA Channel 13 Buffer Address Register (HW_APBX_CH13_BAR)

The APBX DMA Channel 13 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH13_BAR – 8002_4000h base + 6E0h offset = 8002_46E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.101 APBX DMA Channel 13 Semaphore Register (HW_APBX_CH13_SEMA)

The APBX DMA Channel 13 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH13_SEMA – 8002_4000h base + 6F0h offset = 8002_46F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.102 AHB to APBX DMA Channel 13 Debug Information (HW_APBX_CH13_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 13 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 13.

Address: HW_APBX_CH13_DEBUG1 – 8002_4000h base + 700h offset = 8002_4700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W	▨	▨	▨	▨	▨			▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device

HW_APBX_CH13_DEBUG1 field descriptions (continued)

Field	Description
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 13 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.103 AHB to APBX DMA Channel 13 Debug Information (HW_APBX_CH13_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 13.

This register allows debug visibility of the APBX DMA Channel 13.

Address: HW_APBX_CH13_DEBUG2 – 8002_4000h base + 710h offset = 8002_4710h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH13_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.104 APBX DMA Channel 14 Current Command Address Register (HW_APBX_CH14_CURCMDAR)

The APBX DMA Channel 14 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 14 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH14_CURCMDAR – 8002_4000h base + 720h offset = 8002_4720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 14.

7.5.105 APBX DMA Channel 14 Next Command Address Register (HW_APBX_CH14_NXTCMDAR)

The APBX DMA Channel 14 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 14 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 14 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH14_NXTCMDAR – 8002_4000h base + 730h offset = 8002_4730h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_NXTCMDAR field descriptions

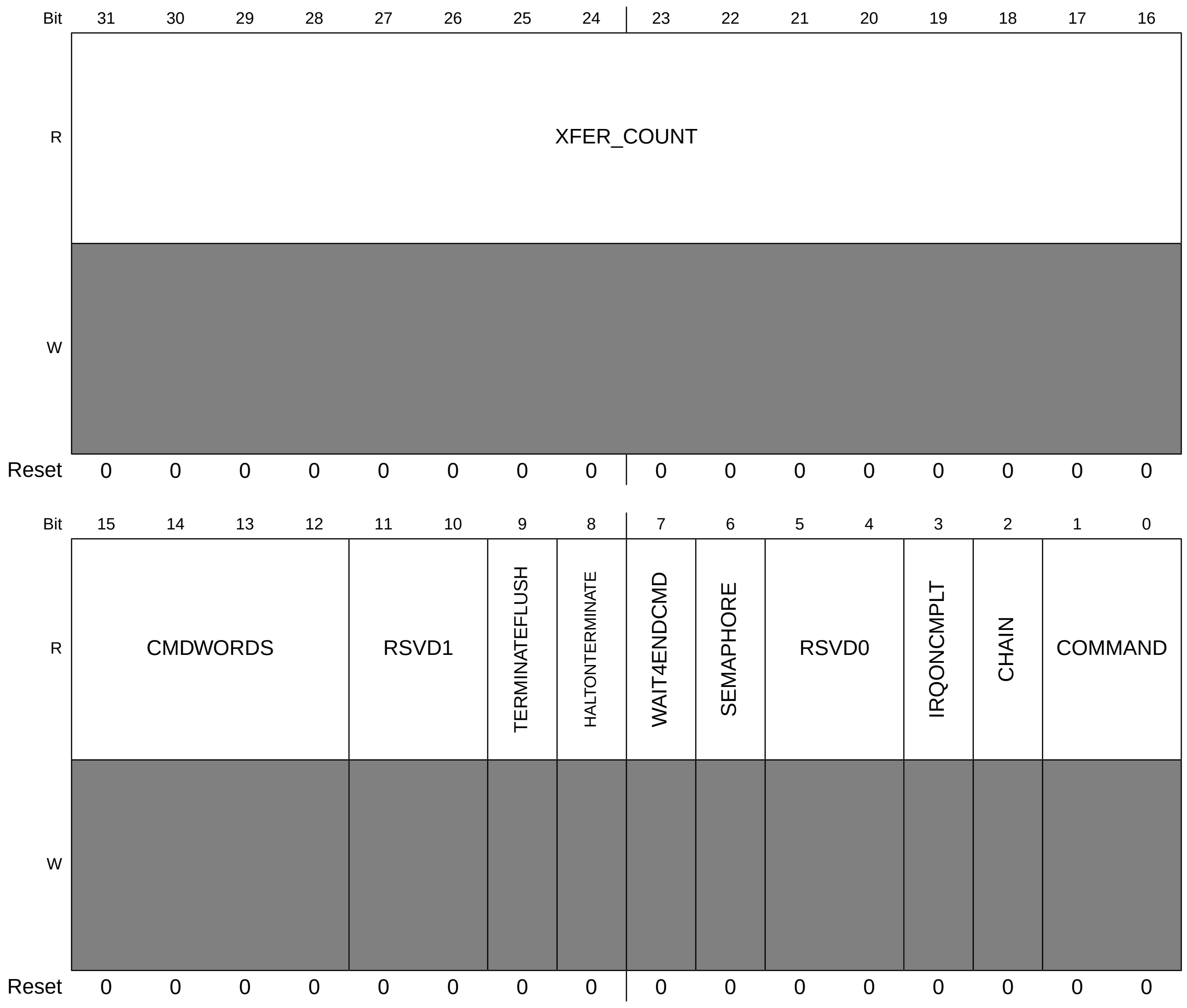
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 14.

7.5.106 APBX DMA Channel 14 Command Register (HW_APBX_CH14_CMD)

The APBX DMA Channel 14 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH14_CMD – 8002_4000h base + 740h offset = 8002_4740h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
XFER_COUNT
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
CMDWORDS
RSVD1
TERMINATEFLUSH
HALTONTERMINATE
WAIT4ENDCMD
SEMAPHORE
RSVD0
IRQONCMPLT
CHAIN
COMMAND
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH14_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART3 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART3, starting with the base PIO address of the UART3 (HW_UARTAPP_CTRL0) and increment from there. Zero means transfer NO command words
11–10 RSVD1	Reserved, always set to zero.
9 TERMINATEFLUSH	A value of one indicates that the channel will flush out any remainder data in DMA FIFO when termination occurs. Only appliese for write DMA operation.

HW_APBX_CH14_CMD field descriptions (continued)

Field	Description
8 HALTONTERMINATE	A value of one indicates that the channel will immeditately terminate the current descriptor and halt the DMA channel if a terminate signal is set. A value of 0 will still cause an immediate terminate of the channel if the terminate signal is set, but the channel will continue as if the count had been exhausted, meaning it will honor IRQONCMPLT, CHAIN, SEMAPHORE, and WAIT4ENDCMD.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH14_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.107 APBX DMA Channel 14 Buffer Address Register (HW_APBX_CH14_BAR)

The APBX DMA Channel 14 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH14_BAR – 8002_4000h base + 750h offset = 8002_4750h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.108 APBX DMA Channel 14 Semaphore Register (HW_APBX_CH14_SEMA)

The APBX DMA Channel 14 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH14_SEMA – 8002_4000h base + 760h offset = 8002_4760h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.109 AHB to APBX DMA Channel 14 Debug Information (HW_APBX_CH14_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 14 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 14.

Address: HW_APBX_CH14_DEBUG1 – 8002_4000h base + 770h offset = 8002_4770h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 14 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly.

HW_APBX_CH14_DEBUG1 field descriptions (continued)

Field	Description
	0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.110 AHB to APBX DMA Channel 14 Debug Information (HW_APBX_CH14_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 14.

This register allows debug visibility of the APBX DMA Channel 14.

Address: HW_APBX_CH14_DEBUG2 – 8002_4000h base + 780h offset = 8002_4780h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH14_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.111 APBX DMA Channel 15 Current Command Address Register (HW_APBX_CH15_CURCMDAR)

The APBX DMA Channel 15 current command address register points to the multiword command that is currently being executed. Commands are threaded on the command address.

APBX DMA Channel 15 is controlled by a variable sized command structure. This register points to the command structure currently being executed.

Address: HW_APBX_CH15_CURCMDAR – 8002_4000h base + 790h offset = 8002_4790h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMD_ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_CURCMDAR field descriptions

Field	Description
31–0 CMD_ADDR	Pointer to command structure currently being processed for Channel 15.

7.5.112 APBX DMA Channel 15 Next Command Address Register (HW_APBX_CH15_NXTCMDAR)

The APBX DMA Channel 15 next command address register points to the next multiword command to be executed. Commands are threaded on the command address. Set CHAIN to one to process command lists.

APBX DMA Channel 15 is controlled by a variable sized command structure. Software loads this register with the address of the first command structure to process and increments the Channel 15 semaphore to start processing. This register points to the next command structure to be executed when the current command is completed.

Address: HW_APBX_CH15_NXTCMDAR – 8002_4000h base + 7A0h offset = 8002_47A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	CMD_ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_NXTCMDAR field descriptions

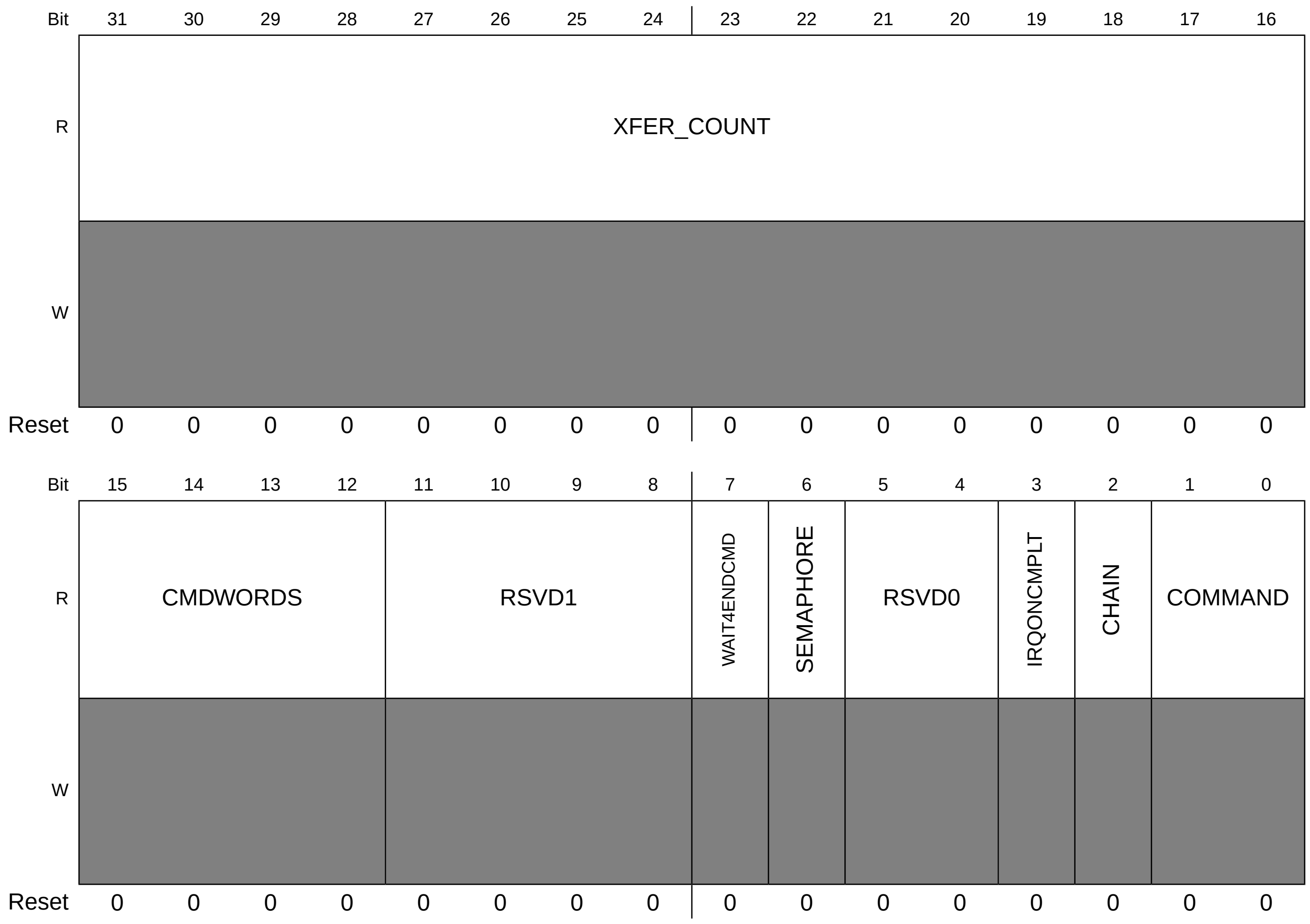
Field	Description
31–0 CMD_ADDR	Pointer to next command structure for Channel 15.

7.5.113 APBX DMA Channel 15 Command Register (HW_APBX_CH15_CMD)

The APBX DMA Channel 15 command register specifies the cycle to perform for the current command chain item.

The command register controls the overall operation of each DMA command for this channel. It includes the number of bytes to transfer to or from the device, the number of APB PIO command words included with this command structure, whether to interrupt at command completion, whether to chain an additional command to the end of this one and whether this transfer is a read or write DMA transfer.

Address: HW_APBX_CH15_CMD – 8002_4000h base + 7B0h offset = 8002_47B0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
XFER_COUNT
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
CMDWORDS
RSVD1
WAIT4ENDCMD
SEMAPHORE
RSVD0
IRQONCMPLT
CHAIN
COMMAND
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_APBX_CH15_CMD field descriptions

Field	Description
31–16 XFER_COUNT	This field indicates the number of bytes to transfer to or from the appropriate PIO register in the UART3 device HW_UARTAPP_DATA register. A value of 0 indicates a 64 KBytes transfer.
15–12 CMDWORDS	This field indicates the number of command words to send to the UART3, starting with the base PIO address of the UART3 (HW_UARTAPP_CTRL1) and increment from there. Zero means transfer NO command words
11–8 RSVD1	Reserved, always set to zero.
7 WAIT4ENDCMD	A value of one indicates that the channel will wait for the end of command signal to be sent from the APBX device to the DMA before starting the next DMA command.

HW_APBX_CH15_CMD field descriptions (continued)

Field	Description
6 SEMAPHORE	A value of one indicates that the channel will decrement its semaphore at the completion of the current command structure. If the semaphore decrements to zero, then this channel stalls until software increments it again.
5–4 RSVD0	Reserved, always set to zero.
3 IRQONCMPLT	A value of one indicates that the channel will cause its interrupt status bit to be set upon completion of the current command, i.e. after the DMA transfer is complete.
2 CHAIN	A value of one indicates that another command is chained onto the end of the current command structure. At the completion of the current command, this channel will follow the pointer in HW_APBX_CH15_CMDAR to find the next command.
1–0 COMMAND	This bitfield indicates the type of current command: 00- NO DMA TRANSFER 01- write transfers, i.e. data sent from the APBX device (APB PIO Read) to the system memory (AHB master write). 10- read transfer 11- reserved 0x0 **NO_DMA_XFER** — Perform any requested PIO word transfers but terminate command before any DMA transfer. 0x1 **DMA_WRITE** — Perform any requested PIO word transfers and then perform a DMA transfer from the peripheral for the specified number of bytes. 0x2 **DMA_READ** — Perform any requested PIO word transfers and then perform a DMA transfer to the peripheral for the specified number of bytes.

7.5.114 APBX DMA Channel 15 Buffer Address Register (HW_APBX_CH15_BAR)

The APBX DMA Channel 15 buffer address register contains a pointer to the data buffer for the transfer. For immediate forms, the data is taken from this register. This is a byte address which means transfers can start on any byte boundary.

This register holds a pointer to the data buffer in system memory. After the command values have been read into the DMA controller and the device controlled by this channel, then the DMA transfer will begin, to or from the buffer pointed to by this register.

Address: HW_APBX_CH15_BAR – 8002_4000h base + 7C0h offset = 8002_47C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_BAR field descriptions

Field	Description
31–0 ADDRESS	Address of system memory buffer to be read or written over the AHB bus.

7.5.115 APBX DMA Channel 15 Semaphore Register (HW_APBX_CH15_SEMA)

The APBX DMA Channel 15 semaphore register is used to synchronize between the CPU instruction stream and the DMA chain processing state.

Each DMA channel has an 8 bit counting semaphore used to synchronize between the program stream and the DMA chain processing. DMA processing continues until the DMA attempts to decrement a semaphore which has already reached a value of zero. When the attempt is made, the DMA channel is stalled until software increments the semaphore count.

Address: HW_APBX_CH15_SEMA – 8002_4000h base + 7D0h offset = 8002_47D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								PHORE								RSVD1								INCREMENT_SEMA							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 PHORE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT_ SEMA	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DMA hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DMA channel decrements the count on the same clock, then the count is incremented by a net one.

7.5.116 AHB to APBX DMA Channel 15 Debug Information (HW_APBX_CH15_DEBUG1)

This register gives debug visibility into the APBX DMA Channel 15 state machine and controls.

This register allows debug visibility of the APBX DMA Channel 15.

Address: HW_APBX_CH15_DEBUG1 – 8002_4000h base + 7E0h offset = 8002_47E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REQ	BURST	KICK	END	RSVD2			NEXTCMDADDRVALID	RD_FIFO_EMPTY	RD_FIFO_FULL	WR_FIFO_EMPTY	WR_FIFO_FULL	RSVD1[19:16]			
W																
Reset	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:5]											STATEMACHINE				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_DEBUG1 field descriptions

Field	Description
31 REQ	This bit reflects the current state of the DMA Request Signal from the APB device
30 BURST	This bit reflects the current state of the DMA Burst Signal from the APB device
29 KICK	This bit reflects the current state of the DMA Kick Signal sent to the APB Device

HW_APBX_CH15_DEBUG1 field descriptions (continued)

Field	Description
28 END	This bit reflects the current state of the DMA End Command Signal sent from the APB Device
27–25 RSVD2	Reserved
24 NEXTCMDADDRVALID	This bit reflects the internal bit which indicates whether the channel's next command address is valid.
23 RD_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Read FIFO Empty signal.
22 RD_FIFO_FULL	This bit reflects the current state of the DMA Channel's Read FIFO Full signal.
21 WR_FIFO_EMPTY	This bit reflects the current state of the DMA Channel's Write FIFO Empty signal.
20 WR_FIFO_FULL	This bit reflects the current state of the DMA Channel's Write FIFO Full signal.
19–5 RSVD1	Reserved
4–0 STATEMACHINE	PIO Display of the DMA Channel 15 state machine state. 0x00 **IDLE** — This is the idle state of the DMA state machine. 0x01 **REQ_CMD1** — State in which the DMA is waiting to receive the first word of a command. 0x02 **REQ_CMD3** — State in which the DMA is waiting to receive the third word of a command. 0x03 **REQ_CMD2** — State in which the DMA is waiting to receive the second word of a command. 0x04 **XFER_DECODE** — The state machine processes the descriptor command field in this state and branches accordingly. 0x05 **REQ_WAIT** — The state machine waits in this state for the PIO APB cycles to complete. 0x06 **REQ_CMD4** — State in which the DMA is waiting to receive the fourth word of a command, or waiting to receive the PIO words when PIO count is greater than 1. 0x07 **PIO_REQ** — This state determines whether another PIO cycle needs to occur before starting DMA transfers. 0x08 **READ_FLUSH** — During a read transfers, the state machine enters this state waiting for the last bytes to be pushed out on the APB. 0x09 **READ_WAIT** — When an AHB read request occurs, the state machine waits in this state for the AHB transfer to complete. 0x0C **WRITE** — During DMA Write transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0D **READ_REQ** — During DMA Read transfers, the state machine waits in this state until the AHB master arbiter accepts the request from this channel. 0x0E **CHECK_CHAIN** — Upon completion of the DMA transfers, this state checks the value of the Chain bit and branches accordingly. 0x0F **XFER_COMPLETE** — The state machine goes to this state after the DMA transfers are complete, and determines what step to take next. 0x15 **WAIT_END** — When the Wait for Command End bit is set, the state machine enters this state until the DMA device indicates that the command is complete. 0x1C **WRITE_WAIT** — During DMA Write transfers, the state machine waits in this state until the AHB master completes the write to the AHB memory space. 0x1E **CHECK_WAIT** — If the Chain bit is a 0, the state machine enters this state and effectively halts.

7.5.117 AHB to APBX DMA Channel 15 Debug Information (HW_APBX_CH15_DEBUG2)

This register gives debug visibility for the APB and AHB byte counts for DMA Channel 15.

This register allows debug visibility of the APBX DMA Channel 15.

Address: HW_APBX_CH15_DEBUG2 – 8002_4000h base + 7F0h offset = 8002_47F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	APB_BYTES																AHB_BYTES															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_CH15_DEBUG2 field descriptions

Field	Description
31–16 APB_BYTES	This value reflects the current number of APB bytes remaining to be transfered in the current transfer.
15–0 AHB_BYTES	This value reflects the current number of AHB bytes remaining to be transfered in the current transfer.

7.5.118 APBX Bridge Version Register (HW_APBX_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (HW_APBX_VERSION.B.MAJOR != 1)
   Error();
```

Address: HW_APBX_VERSION – 8002_4000h base + 800h offset = 8002_4800h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_APBX_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 8
Pin Descriptions

8.1 Pin Descriptions Overview

This chapter provides various views of the pinout for the i.MX28.

- Pin definitions for the 289-pin BGA are in Pin Definitions for 289-pin BGA
- Pin sorted by functional groups are in Functional Pin Groups

The pin tables in this chapter include columns with the headings MUX0, MUX1, and MUX2. These columns refer to the different functions that can be enabled for each individual pin by programming the pin control registers (HW_PINCTRL_MUXSEL*x*). For example:

- Enable the function listed in the "MUX0" column by programming the BANK*x*_PIN*x* bit field to 00.
- Enable the function listed in the "MUX1" column by programming the BANK*x*_PIN*x* bit field to 01.
- Enable the function listed in the "MUX2" column by programming the BANK*x*_PIN*x* bit field to 10.

See Pin Control and GPIO Overview for more information.

Table 8-1 lists the abbreviations used in the pin tables in this chapter.

Table 8-1. Nomenclature for Pin Tables

TYPE	DESCRIPTION		MODULE	DESCRIPTION
A	Analog pin		ADC	ADC analog pins
D	Digital pin		CLOCK	Clock pins
E	EMI pin		DCDC	DC-DC Converter pins
P	Power pin		EMI	External Memory Interface pins
I	Input pin		ETM	Embedded Trace Macrocell

TYPE	DESCRIPTION		MODULE	DESCRIPTION
O	Output pin		GPIO	General-Purpose Input/Output pins
I/O	Input/output pin		GPMI	General-Purpose Media Interface (NAND/ATA/CMOS) pins
			HP	Headphone pins
			I^2C	I^2C pins
			LCDIF	LCD Interface pins
			LRADC	Low-Resolution ADC/Touch-Screen pins
			POWER	Power pins
			PWM	Pulse Width Modulator pins
			RTC	Real-Time Clock pins
			SSP	Synchronous Serial Port pins
			SYSTEM	System pins
			TIMER	Timer/Rotary Encoder pins
			UART	Either debug or application UART pins
			USB	USB pins

Note

Almost all digital pins are powered down (that is, high-impedance) or configured as GPIO input at reset, until reprogrammed. The only exceptions are TEST PINS(TESTMODE, DEBUG, JTAG pins). These pins are always active.

8.2 Pin Definitions for 289-pin BGA

This section includes the following pin information for the 289-pin BGA package:

- Table 8-2
- Figure 8-1

Table 8-2. Pin Definitions

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
AUART0_CTS	J6	AUART	D	AUART0_CTS	AUART4_RX	DUART_RX
AUART0_RTS	J7	AUART	D	AUART0_RTS	AUART4_TX	DUART_TX

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
AUART0_RX	G5	AUART	D	AUART0_RX	I2C0_SCL	DUART_CTS
AUART0_TX	H5	AUART	D	AUART0_TX	I2C0_SDA	DUART_RTS
AUART1_CTS	K5	AUART	D	AUART1_CTS	USB0_OVERCURRENT	TIMROT_ROTARYA
AUART1_RTS	J5	AUART	D	AUART1_RTS	USB0_ID	TIMROT_ROTARYB
AUART1_RX	L4	AUART	D	AUART1_RX	SSP2_CARD_DETECT	PWM_0
AUART1_TX	K4	AUART	D	AUART1_TX	SSP3_CARD_DETECT	PWM_1
AUART2_CTS	H6	AUART	D	AUART2_CTS	I2C1_SCL	SAIF1_BITCLK
AUART2_RTS	H7	AUART	D	AUART2_RTS	I2C1_SDA	SAIF1_LRCLK
AUART2_RX	F6	AUART	D	AUART2_RX	SSP3_D1	SSP3_D4
AUART2_TX	F5	AUART	D	AUART2_TX	SSP3_D2	SSP3_D5
AUART3_CTS	L6	AUART	D	AUART3_CTS	CAN1_TX	EN-ET0_1588_EVENT1_OUT
AUART3_RTS	K6	AUART	D	AUART3_RTS	CAN1_RX	EN-ET0_1588_EVENT1_IN
AUART3_RX	M5	AUART	D	AUART3_RX	CAN0_TX	EN-ET0_1588_EVENT0_OUT
AUART3_TX	L5	AUART	D	AUART3_TX	CAN0_RX	EN-ET0_1588_EVENT0_IN
BATTERY	A15	DCDC	A			
DCDC_BATT	B15	DCDC	A			
DCDC_GND	A17	DCDC	A			
DCDC_LN1	B17	DCDC	A			
DCDC_LP	A16	DCDC	A			
DCDC_VDDA	B16	DCDC	A			
DCDC_VDDD	D17	DCDC	A			
DCDC_VDDIO	C17	DCDC	A			
DEBUG	B9	SYS-TEM	D			
EMI_A00	U15	EMI	E	EMI_ADDR0		
EMI_A01	U12	EMI	E	EMI_ADDR1		
EMI_A02	U14	EMI	E	EMI_ADDR2		
EMI_A03	T11	EMI	E	EMI_ADDR3		
EMI_A04	U10	EMI	E	EMI_ADDR4		
EMI_A05	R11	EMI	E	EMI_ADDR5		
EMI_A06	R9	EMI	E	EMI_ADDR6		
EMI_A07	N11	EMI	E	EMI_ADDR7		
EMI_A08	U9	EMI	E	EMI_ADDR8		
EMI_A09	P10	EMI	E	EMI_ADDR9		

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
EMI_A10	U13	EMI	E	EMI_ADDR10		
EMI_A11	T10	EMI	E	EMI_ADDR11		
EMI_A12	U11	EMI	E	EMI_ADDR12		
EMI_A13	T9	EMI	E	EMI_ADDR13		
EMI_A14	N10	EMI	E	EMI_ADDR14		
EMI_BA0	T16	EMI	E	EMI_BA0		
EMI_BA1	T12	EMI	E	EMI_BA1		
EMI_BA2	N12	EMI	E	EMI_BA2		
EMI_CASN	U16	EMI	E	EMI_CASN		
EMI_CE0N	P12	EMI	E	EMI_CE0N		
EMI_CE1N	P9	EMI	E	EMI_CE1N		
EMI_CKE	T13	EMI	E	EMI_CKE		
EMI_CLK	L17	EMI	E	EMI_CLK		
EMI_CLKN	L16	EMI	E			
EMI_D00	N16	EMI	E	EMI_DATA0		
EMI_D01	M13	EMI	E	EMI_DATA1		
EMI_D02	P15	EMI	E	EMI_DATA2		
EMI_D03	N14	EMI	E	EMI_DATA3		
EMI_D04	P13	EMI	E	EMI_DATA4		
EMI_D05	P17	EMI	E	EMI_DATA5		
EMI_D06	L14	EMI	E	EMI_DATA6		
EMI_D07	M17	EMI	E	EMI_DATA7		
EMI_D08	G16	EMI	E	EMI_DATA8		
EMI_D09	H15	EMI	E	EMI_DATA9		
EMI_D10	G14	EMI	E	EMI_DATA10		
EMI_D11	J14	EMI	E	EMI_DATA11		
EMI_D12	H13	EMI	E	EMI_DATA12		
EMI_D13	H17	EMI	E	EMI_DATA13		
EMI_D14	F13	EMI	E	EMI_DATA14		
EMI_D15	F17	EMI	E	EMI_DATA15		
EMI_DDR_OPEN	K14	EMI	E	EMI_DDR_OPEN		
EMI_DDR_OPEN_FB	L15	EMI	E	EMI_DDR_OPEN_FEED-BACK		
EMI_DQM0	M15	EMI	E	EMI_DQM0		
EMI_DQM1	F15	EMI	E	EMI_DQM1		
EMI_DQS0	K17	EMI	E	EMI_DQS0		

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
EMI_DQS0N	K16	EMI	E			
EMI_DQS1	J17	EMI	E	EMI_DQS1		
EMI_DQS1N	J16	EMI	E			
EMI_ODT0	R17	EMI	E	EMI_ODT0		
EMI_ODT1	T17	EMI	E	EMI_ODT1		
EMI_RASN	R16	EMI	E	EMI_RASN		
EMI_VREF0	R14	EMI	E			
EMI_VREF1	K13	EMI	E			
EMI_WEN	T15	EMI	E	EMI_WEN		
ENET0_COL	J4	ENET	D	ENET0_COL	ENET1_TX_EN	EN-ET0_1588_EVENT3_OUT
ENET0_CRS	J3	ENET	D	ENET0_CRS	ENET1_RX_EN	EN-ET0_1588_EVENT3_IN
ENET0_MDC	G4	ENET	D	ENET0_MDC	GPMI_CE4N	SAIF0_SDATA1
ENET0_MDIO	H4	ENET	D	ENET0_MDIO	GPMI_CE5N	SAIF0_SDATA2
ENET0_RXD0	H1	ENET	D	ENET0_RXD0	GPMI_CE7N	SAIF1_SDATA2
ENET0_RXD1	H2	ENET	D	ENET0_RXD1	GPMI_READY4	
ENET0_RXD2	J1	ENET	D	ENET0_RXD2	ENET1_RXD0	EN-ET0_1588_EVENT0_OUT
ENET0_RXD3	J2	ENET	D	ENET0_RXD3	ENET1_RXD1	EN-ET0_1588_EVENT0_IN
ENET0_RX_CLK	F3	ENET	D	ENET0_RX_CLK	ENET0_RX_ER	EN-ET0_1588_EVENT2_IN
ENET0_RX_EN	E4	ENET	D	ENET0_RX_EN	GPMI_CE6N	SAIF1_SDATA1
ENET0_TXD0	F1	ENET	D	ENET0_TXD0	GPMI_READY6	
ENET0_TXD1	F2	ENET	D	ENET0_TXD1	GPMI_READY7	
ENET0_TXD2	G1	ENET	D	ENET0_TXD2	ENET1_TXD0	EN-ET0_1588_EVENT1_OUT
ENET0_TXD3	G2	ENET	D	ENET0_TXD3	ENET1_TXD1	EN-ET0_1588_EVENT1_IN
ENET0_TX_CLK	E3	ENET	D	ENET0_TX_CLK	HSADC_TRIGGER	EN-ET0_1588_EVENT2_OUT
ENET0_TX_EN	F4	ENET	D	ENET0_TX_EN	GPMI_READY5	
ENET_CLK	E2	ENET	D	CLKCTRL_ENET		
GPMI_ALE	P6	GPMI	D	GPMI_ALE	SSP3_D1	SSP3_D4
GPMI_CE0N	N7	GPMI	D	GPMI_CE0N	SSP3_D0	
GPMI_CE1N	N9	GPMI	D	GPMI_CE1N	SSP3_D3	
GPMI_CE2N	M7	GPMI	D	GPMI_CE2N	CAN1_TX	ENET0_RX_ER
GPMI_CE3N	M9	GPMI	D	GPMI_CE3N	CAN1_RX	SAIF1_MCLK

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
GPMI_CLE	P7	GPMI	D	GPMI_CLE	SSP3_D2	SSP3_D5
GPMI_D00	U8	GPMI	D	GPMI_D0	SSP1_D0	
GPMI_D01	T8	GPMI	D	GPMI_D1	SSP1_D1	
GPMI_D02	R8	GPMI	D	GPMI_D2	SSP1_D2	
GPMI_D03	U7	GPMI	D	GPMI_D3	SSP1_D3	
GPMI_D04	T7	GPMI	D	GPMI_D4	SSP1_D4	
GPMI_D05	R7	GPMI	D	GPMI_D5	SSP1_D5	
GPMI_D06	U6	GPMI	D	GPMI_D6	SSP1_D6	
GPMI_D07	T6	GPMI	D	GPMI_D7	SSP1_D7	
GPMI_RDN	R6	GPMI	D	GPMI_RDN	SSP3_SCK	
GPMI_RDY0	N6	GPMI	D	GPMI_READY0	SSP1_CARD_DETECT	USB0_ID
GPMI_RDY1	N8	GPMI	D	GPMI_READY1	SSP1_CMD	
GPMI_RDY2	M8	GPMI	D	GPMI_READY2	CAN0_TX	ENET0_TX_ER
GPMI_RDY3	L8	GPMI	D	GPMI_READY3	CAN0_RX	HSADC_TRIGGER
GPMI_RESETN	L9	GPMI	D	GPMI_RESETN	SSP3_CMD	
GPMI_WRN	P8	GPMI	D	GPMI_WRN	SSP1_SCK	
HSADC0	B14	HSADC	A			
I2C0_SCL	C7	I2C	D	I2C0_SCL	TIMROT_ROTARYA	DUART_RX
I2C0_SDA	D8	I2C	D	I2C0_SDA	TIMROT_ROTARYB	DUART_TX
JTAG_RTCK	E14	SYS-TEM	D	JTAG_RTCK		
JTAG_TCK	E11	SYS-TEM	D			
JTAG_TDI	E12	SYS-TEM	D			
JTAG_TDO	E13	SYS-TEM	D			
JTAG_TMS	D12	SYS-TEM	D			
JTAG_TRST	D14	SYS-TEM	D			
LCD_CS	P5	LCD	D	LCD_CS	LCD_ENABLE	
LCD_D00	K2	LCD	D	LCD_D0		ETM_DA0
LCD_D01	K3	LCD	D	LCD_D1		ETM_DA1
LCD_D02	L2	LCD	D	LCD_D2		ETM_DA2
LCD_D03	L3	LCD	D	LCD_D3	ETM_DA8	ETM_DA3
LCD_D04	M2	LCD	D	LCD_D4	ETM_DA9	ETM_DA4
LCD_D05	M3	LCD	D	LCD_D5		ETM_DA5

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
LCD_D06	N2	LCD	D	LCD_D6		ETM_DA6
LCD_D07	P1	LCD	D	LCD_D7		ETM_DA7
LCD_D08	P2	LCD	D	LCD_D8	ETM_DA3	ETM_DA8
LCD_D09	P3	LCD	D	LCD_D9	ETM_DA4	ETM_DA9
LCD_D10	R1	LCD	D	LCD_D10		ETM_DA10
LCD_D11	R2	LCD	D	LCD_D11		ETM_DA11
LCD_D12	T1	LCD	D	LCD_D12		ETM_DA12
LCD_D13	T2	LCD	D	LCD_D13		ETM_DA13
LCD_D14	U2	LCD	D	LCD_D14		ETM_DA14
LCD_D15	U3	LCD	D	LCD_D15		ETM_DA15
LCD_D16	T3	LCD	D	LCD_D16		ETM_DA7
LCD_D17	R3	LCD	D	LCD_D17		ETM_DA6
LCD_D18	U4	LCD	D	LCD_D18		ETM_DA5
LCD_D19	T4	LCD	D	LCD_D19		ETM_DA4
LCD_D20	R4	LCD	D	LCD_D20	EN-ET1_1588_EVENT2_OUT	ETM_DA3
LCD_D21	U5	LCD	D	LCD_D21	EN-ET1_1588_EVENT2_IN	ETM_DA2
LCD_D22	T5	LCD	D	LCD_D22	EN-ET1_1588_EVENT3_OUT	ETM_DA1
LCD_D23	R5	LCD	D	LCD_D23	EN-ET1_1588_EVENT3_IN	ETM_DA0
LCD_DOTCLK	N1	LCD	D	LCD_DOTCLK	SAIF1_MCLK	ETM_TCLK
LCD_ENABLE	N5	LCD	D	LCD_ENABLE		
LCD_HSYNC	M1	LCD	D	LCD_HSYNC	SAIF1_SDATA1	ETM_TCTL
LCD_RD_E	P4	LCD	D	LCD_RD_E	LCD_VSYNC	ETM_TCTL
LCD_RESET	M6	LCD	D	LCD_RESET	LCD_VSYNC	
LCD_RS	M4	LCD	D	LCD_RS	LCD_DOTCLK	
LCD_VSYNC	L1	LCD	D	LCD_VSYNC	SAIF1_SDATA0	
LCD_WR_RWN	K1	LCD	D	LCD_WR_RWN	LCD_HSYNC	ETM_TCLK
LRADC0	C15	LRADC	A			
LRADC1	C9	LRADC	A			
LRADC2	C8	LRADC	A			
LRADC3	D9	LRADC	A			
LRADC4	D13	LRADC	A			
LRADC5	D15	LRADC	A			
LRADC6	C14	LRADC	A			
PSWITCH	A11	DCDC	A			

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
PWM0	K7	PWM	D	PWM_0	I2C1_SCL	DUART_RX
PWM1	L7	PWM	D	PWM_1	I2C1_SDA	DUART_TX
PWM2	K8	PWM	D	PWM_2	USB0_ID	USB1_OVERCURRENT
PWM3	E9	PWM	D	PWM_3		
PWM4	E10	PWM	D	PWM_4		
RESETN	A14	DCDC	A			
RTC_XTALI	D11	XTAL	A			
RTC_XTALO	C11	XTAL	A			
SAIF0_BITCLK	F7	SAIF	D	SAIF0_BITCLK	PWM_5	AUART4_RX
SAIF0_LRCLK	G6	SAIF	D	SAIF0_LRCLK	PWM_4	AUART4_RTS
SAIF0_MCLK	G7	SAIF	D	SAIF0_MCLK	PWM_3	AUART4_CTS
SAIF0_SDATA0	E7	SAIF	D	SAIF0_SDATA0	PWM_6	AUART4_TX
SAIF1_SDATA0	E8	SAIF	D	SAIF1_SDATA0	PWM_7	SAIF0_SDATA1
SPDIF	D7	SPDIF	D	SPDIF_TX		ENET1_RX_ER
SSP0_CMD	A4	SSP	D	SSP0_CMD		
SSP0_DATA0	B6	SSP	D	SSP0_D0		
SSP0_DATA1	C6	SSP	D	SSP0_D1		
SSP0_DATA2	D6	SSP	D	SSP0_D2		
SSP0_DATA3	A5	SSP	D	SSP0_D3		
SSP0_DATA4	B5	SSP	D	SSP0_D4	SSP2_D0	
SSP0_DATA5	C5	SSP	D	SSP0_D5	SSP2_D3	
SSP0_DATA6	D5	SSP	D	SSP0_D6	SSP2_CMD	
SSP0_DATA7	B4	SSP	D	SSP0_D7	SSP2_SCK	
SSP0_DETECT	D10	SSP	D	SSP0_CARD_DE-TECT		
SSP0_SCK	A6	SSP	D	SSP0_SCK		
SSP1_CMD	C1	SSP	D	SSP1_CMD	SSP2_D2	EN-ET0_1588_EVENT2_IN
SSP1_DATA0	D1	SSP	D	SSP1_D0	SSP2_D6	EN-ET0_1588_EVENT3_OUT
SSP1_DATA3	E1	SSP	D	SSP1_D3	SSP2_D7	EN-ET0_1588_EVENT3_IN
SSP1_SCK	B1	SSP	D	SSP1_SCK	SSP2_D1	EN-ET0_1588_EVENT2_OUT
SSP2_MISO	B3	SSP	D	SSP2_D0	AUART3_RX	SAIF1_SDATA1
SSP2_MOSI	C3	SSP	D	SSP2_CMD	AUART2_TX	SAIF0_SDATA2
SSP2_SCK	A3	SSP	D	SSP2_SCK	AUART2_RX	SAIF0_SDATA1
SSP2_SS0	C4	SSP	D	SSP2_D3	AUART3_TX	SAIF1_SDATA2

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
SSP2_SS1	D3	SSP	D	SSP2_D4	SSP2_D1	USB1_OVERCURRENT
SSP2_SS2	D4	SSP	D	SSP2_D5	SSP2_D2	USB0_OVERCURRENT
SSP3_MISO	B2	SSP	D	SSP3_D0	AUART4_RTS	EN-ET1_1588_EVENT1_OUT
SSP3_MOSI	C2	SSP	D	SSP3_CMD	AUART4_RX	EN-ET1_1588_EVENT0_IN
SSP3_SCK	A2	SSP	D	SSP3_SCK	AUART4_TX	EN-ET1_1588_EVENT0_OUT
SSP3_SS0	D2	SSP	D	SSP3_D3	AUART4_CTS	EN-ET1_1588_EVENT1_IN
TESTMODE	C10	SYS-TEM	D			
USB0DM	A10	USB	A			
USB0DP	B10	USB	A			
USB1DM	B8	USB	A			
USB1DP	A8	USB	A			
VDD1P5	D16	DCDC	A			
VDD4P2	A13	DCDC	A			
VDD5V	E17	DCDC	A			
VDDA1	C13	POWER	A			
VDDD	G12, G11, F10, F11, K12, F12, G10	POWER	P			
VDDIO18	G8, F9, F8, G9	POWER	P			
VDDIO33	H8, J8, N3, G3, E6, J9, J10, A7, E16	POWER	P			
VDDIO33_EMI	N17	POWER	P			
VDDIO_EMI	P11, R13, N13, N15, G17, M12, M10, G13, M11, L13, G15	POWER	P			
VDDIO_EMIQ	K15, J13, R15	POWER	P			
VDDXTAL	C12	POWER	A			

PIN NAME	PIN	GROUP	TYPE	MUX0	MUX1	MUX2
VSS	E15, L11, A1, K10, K11, J11, M14, H11, U1, H9, H12, H3, K9, C16, L10, H16, J12, H10, B7, E5, J15, A9, N4	POWER	P			
VSSA1	B13	POWER	A			
VSSA2	B11	POWER	A			
VSSIO_EMI	F16, R10, H14, M16, F14, L12, P16, U17, T14, P14, R12	POWER	P			
XTALI	A12	XTAL	A			
XTALO	B12	XTAL	A			

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
A	VSS	SSP3_SCK	SSP2_SCK	SSP0_CMD	SSP0_DATA3	SSP0_SCK	VDDIO33	USB1DP	VSS	USB0DM	PSWITCH	XTALI	VDD4P2	RESETN	BATTERY	DCDC_LP	DCDC_GND	A
B	SSP1_SCK	SSP3_MISO	SSP2_MISO	SSP0_DATA7	SSP0_DATA4	SSP0_DATA0	VSS	USB1DM	DEBUG	USB0DP	VSSA2	XTALO	VSSA1	HSADC0	DCDC_BATT	DCDC_VDDA	DCDC_LN1	B
C	SSP1_CMD	SSP3_MOSI	SSP2_MOSI	SSP2_SS0	SSP0_DATA5	SSP0_DATA1	I2C0_SCL	LRADC2	LRADC1	TESTMODE	RTC_XTALO	VDDXTAL	VDDA1	LRADC6	LRADC0	VSS	DCDC_VDDIO	C
D	SSP1_DATA0	SSP3_SS0	SSP2_SS1	SSP2_SS2	SSP0_DATA6	SSP0_DATA2	SPDIF	I2C0_SDA	LRADC3	SSP0_DETECT	RTC_XTALI	JTAG_TMS	LRADC4	JTAG_TRST	LRADC5	VDD1P5	DCDC_VDDD	D
E	SSP1_DATA3	ENET_CLK	ENET0_TX_CLK	ENET0_RX_EN	VSS	VDDIO33	SAIF0_SDATA0	SAIF1_SDATA0	PWM3	PWM4	JTAG_TCK	JTAG_TDI	JTAG_TDO	JTAG_RTCK	VSS	VDDIO33	VDD5V	E
F	ENET0_TXD0	ENET0_TXD1	ENET0_RX_CLK	ENET0_TX_EN	AUART2_TX	AUART2_RX	SAIF0_BITCLK	VDDIO18	VDDIO18	VDDD	VDDD	VDDD	EMI_D14	VSSIO_EMI	EMI_DQM1	VSSIO_EMI	EMI_D15	F
G	ENET0_TXD2	ENET0_TXD3	VDDIO33	ENET0_MDC	AUART0_RX	SAIF0_LRCLK	SAIF0_MCLK	VDDIO18	VDDIO18	VDDD	VDDD	VDDD	VDDIO_EMI	EMI_D10	VDDIO_EMI	EMI_D08	VDDIO_EMI	G
H	ENET0_RXD0	ENET0_RXD1	VSS	ENET0_MDIO	AUART0_TX	AUART2_CTS	AUART2_RTS	VDDIO33	VSS	VSS	VSS	VSS	EMI_D12	VSSIO_EMI	EMI_D09	VSS	EMI_D13	H
J	ENET0_RXD2	ENET0_RXD3	ENET0_CRS	ENET0_COL	AUART1_RTS	AUART0_CTS	AUART0_RTS	VDDIO33	VDDIO33	VDDIO33	VSS	VSS	VDDIO_EMIQ	EMI_D11	VSS	EMI_DQS1N	EMI_DQS1	J
K	LCD_WR_RWN	LCD_D00	LCD_D01	AUART1_TX	AUART1_CTS	AUART3_RTS	PWM0	PWM2	VSS	VSS	VSS	VDDD	EMI_VREF1	EMI_DDR_OPEN	VDDIO_EMIQ	EMI_DQS0N	EMI_DQS0	K
L	LCD_VSYNC	LCD_D02	LCD_D03	AUART1_RX	AUART3_TX	AUART3_CTS	PWM1	GPMI_RDY3	GPMI_RESETN	VSS	VSS	VSSIO_EMI	VDDIO_EMI	EMI_D06	EMI_DDR_OPEN_FB	EMI_CLKN	EMI_CLK	L
M	LCD_HSYNC	LCD_D04	LCD_D05	LCD_RS	AUART3_RX	LCD_RESET	GPMI_CE2N	GPMI_RDY2	GPMI_CE3N	VDDIO_EMI	VDDIO_EMI	VDDIO_EMI	EMI_D01	VSS	EMI_DQM0	VSSIO_EMI	EMI_D07	M
N	LCD_DOTCLK	LCD_D06	VDDIO33	VSS	LCD_ENABLE	GPMI_RDY0	GPMI_CE0N	GPMI_RDY1	GPMI_CE1N	EMI_A14	EMI_A07	EMI_BA2	VDDIO_EMI	EMI_D03	VDDIO_EMI	EMI_D00	VDDIO33_EMI	N
P	LCD_D07	LCD_D08	LCD_D09	LCD_RD_E	LCD_CS	GPMI_ALE	GPMI_CLE	GPMI_WRN	EMI_CE1N	EMI_A09	VDDIO_EMI	EMI_CE0N	EMI_D04	VSSIO_EMI	EMI_D02	VSSIO_EMI	EMI_D05	P
R	LCD_D10	LCD_D11	LCD_D17	LCD_D20	LCD_D23	GPMI_RDN	GPMI_D05	GPMI_D02	EMI_A06	VSSIO_EMI	EMI_A05	VSSIO_EMI	VDDIO_EMI	EMI_VREF0	VDDIO_EMIQ	EMI_RASN	EMI_ODT0	R
T	LCD_D12	LCD_D13	LCD_D16	LCD_D19	LCD_D22	GPMI_D07	GPMI_D04	GPMI_D01	EMI_A13	EMI_A11	EMI_A03	EMI_BA1	EMI_CKE	VSSIO_EMI	EMI_WEN	EMI_BA0	EMI_ODT1	T
U	VSS	LCD_D14	LCD_D15	LCD_D18	LCD_D21	GPMI_D06	GPMI_D03	GPMI_D00	EMI_A08	EMI_A04	EMI_A12	EMI_A01	EMI_A10	EMI_A02	EMI_A00	EMI_CASN	VSSIO_EMI	U
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	

Figure 8-1. Ball Map

8.3 Functional Pin Groups

This section includes the i.MX28 pins listed in tables by function.

Refer to the pin name tables in the previous sections for appropriate package pin numbers.

1. Table 8-3
2. Table 8-4

3. Table 8-5
4. Table 8-6
5. Table 8-7
6. Table 8-8
7. Table 8-9
8. Table 8-10
9. Table 8-11
10. Table 8-12
11. Table 8-13
12. Table 8-14
13. Table 8-15
14. Table 8-16
15. Table 8-17
16. Table 8-18
17. Table 8-19
18. Table 8-20
19. Table 8-21

Table 8-3. DCDC

PIN NAME	GROUP	TYPE	DESCRIPTION
VDD1P5	DCDC	A	1.5V regulator output for LVDDR2 power supply
DCDC_VDDD	DCDC	A	DCDC output for Digital Core Power w/ typical value of 1.2V
DCDC_LN2	DCDC	A	DCDC Inductor N 2
DCDC_VDDIO	DCDC	A	DCDC output for I/O Power w/ typical value of 3.3V
DCDC_VDDA	DCDC	A	DCDC output for analog/DDR2/1.8V IO power w/ typical value of 1.8V
DCDC_LN1	DCDC	A	DCDC Inductor N 1
DCDC_GND	DCDC	A	DCDC Ground
DCDC_LP	DCDC	A	DCDC Inductor P

PIN NAME	GROUP	TYPE	DESCRIPTION
DCDC_BATT	DCDC	A	DCDC Battery
VDD4P2_DCDC	DCDC	A	DCDC 4.2V Input
VDD4P2	DCDC	A	DCDC 4.2V Regulated Output
VDD5V	DCDC	A	5V Power Input
BATTERY	DCDC	A	Battery Input
RESETN	DCDC	A	Chip-wide Reset In
PSWITCH	DCDC	A	Power On / Recovery / Software Visible

Table 8-4. POWER

PIN NAME	GROUP	TYPE	DESCRIPTION
VDDA1	POWER	A	Analog Power 1
VDDD	POWER	P	
VDDIO18	POWER	P	
VDDIO33	POWER	P	
VDDIO33_EMI	POWER	P	Digital I/O Quiet Power 0 / EMI
VDDIO_EMI	POWER	P	
VDDIO_EMIQ	POWER	P	Digital I/O Quiet Power 0 / EMI
VDDXTAL	POWER	A	Crystal Power Filter Capacitor
VSSA1	POWER	A	Analog Ground 1
VSSA2	POWER	A	Analog Ground 2
VSSA3	POWER	A	Analog Ground 3
VSSD	POWER	P	
VSSIO18	POWER	P	
VSSIO33	POWER	P	
VSSIO_EMI	POWER	P	
VSSIO_EMIQ	POWER	P	

Table 8-5. Analog application pins

PIN NAME	GROUP	TYPE	DESCRIPTION
HSADC0	HSADC	A	HSADC
LRADC6	LRADC	A	LRADC6
LRADC5	LRADC	A	LRADC5
LRADC4	LRADC	A	LRADC4
LRADC3	LRADC	A	LRADC3
LRADC2	LRADC	A	LRADC2
LRADC1	LRADC	A	LRADC1
LRADC0	LRADC	A	LRADC0
USB0DP	USB	A	USB0 Positive Data Line
USB0DM	USB	A	USB0 Negative Data Line
USB1DP	USB	A	USB1 Positive Data Line
USB1DM	USB	A	USB1 Negative Data Line
RTC_XTALI	XTAL	A	32.768 or 32.0 KHz Xtal In
RTC_XTALO	XTAL	A	32.768 or 32.0 KHz Xtal Out
XTALO	XTAL	A	24MHz Crystal Out
XTALI	XTAL	A	24MHz Crystal In

Table 8-6. AUART

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SAIF0_SDATA0	SAIF	D	SAIF0 Serial Data0 (stereo)	SAIF0_SDATA0	PWM_6	AUART4_TX
SSP2_SCK	SSP	D	SSP Serial Clock	SSP2_SCK	AUART2_RX	SAIF0_SDATA1
SSP3_SCK	SSP	D	SSP Serial Clock	SSP3_SCK	AUART4_TX	ENET1_1588_EVENT0_OUT
SSP2_MISO	SSP	D	SD/MMC/Data0 / SPI MISO	SSP2_D0	AUART3_RX	SAIF1_SDATA1
SSP2_SS0	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP2_D3	AUART3_TX	SAIF1_SDATA2
SAIF0_BITCLK	SAIF	D		SAIF0_BITCLK	PWM_5	AUART4_RX

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SSP3_MISO	SSP	D	SD/MMC/Data0 / SPI MISO	SSP3_D0	AUART4_RTS	EN-ET1_1588_EVENT1_OUT
SSP2_MOSI	SSP	D	SD/MMC CMD / SPI MOSI	SSP2_CMD	AUART2_TX	SAIF0_SDATA2
AUART2_RX	AUART	D	Application UART2 RX	AUART2_RX	SSP3_D1	SSP3_D4
SSP3_MOSI	SSP	D	SD/MMC CMD / SPI MOSI	SSP3_CMD	AUART4_RX	EN-ET1_1588_EVENT0_IN
SSP3_SS0	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP3_D3	AUART4_CTS	EN-ET1_1588_EVENT1_IN
SAIF0_MCLK	SAIF	D		SAIF0_MCLK	PWM_3	AUART4_CTS
SAIF0_LRCLK	SAIF	D	SAIF0 Left/Right Clock	SAIF0_LRCLK	PWM_4	AUART4_RTS
AUART2_TX	AUART	D	Application UART2 TX	AUART2_TX	SSP3_D2	SSP3_D5
AUART2_RTS	AUART	D	Application UART2 RTS Flow Control	AUART2_RTS	I2C1_SDA	SAIF1_LRCLK
AUART2_CTS	AUART	D	Application UART2 CTS Flow Control	AUART2_CTS	I2C1_SCL	SAIF1_BITCLK
AUART0_RX	AUART	D	Application UART0 RX	AUART0_RX	I2C0_SCL	DUART_CTS
AUART0_RTS	AUART	D	Application UART0 RTS Flow Control	AUART0_RTS	AUART4_TX	DUART_TX
AUART0_TX	AUART	D	Application UART0 TX	AUART0_TX	I2C0_SDA	DUART_RTS
AUART0_CTS	AUART	D	Application UART0 CTS Flow Control	AUART0_CTS	AUART4_RX	DUART_RX
AUART1_RTS	AUART	D	Application UART1 RTS Flow Control	AUART1_RTS	USB0_ID	TIMROT_ROTARYB
AUART3_RTS	AUART	D	Application UART2 RTS Flow Control	AUART3_RTS	CAN1_RX	EN-ET0_1588_EVENT1_IN
AUART1_CTS	AUART	D	Application UART1 CTS Flow Control	AUART1_CTS	USB0_OVERCUR-RENT	TIMROT_ROTARYA
AUART1_TX	AUART	D	Application UART1 TX	AUART1_TX	SSP3_CARD_DE-TECT	PWM_1
AUART3_TX	AUART	D	Application UART2 TX	AUART3_TX	CAN0_RX	EN-ET0_1588_EVENT0_IN
AUART1_RX	AUART	D	Application UART1 RX	AUART1_RX	SSP2_CARD_DE-TECT	PWM_0
AUART3_CTS	AUART	D	Application UART2 CTS Flow Control	AUART3_CTS	CAN1_TX	EN-ET0_1588_EVENT1_OUT

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
AUART3_RX	AUART	D	Application UART2 RX	AUART3_RX	CAN0_TX	EN-ET0_1588_EVENT0_OUT

Table 8-7. CAN

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
AUART3_RTS	AUART	D	Application UART2 RTS Flow Control	AUART3_RTS	CAN1_RX	EN-ET0_1588_EVENT1_IN
AUART3_TX	AUART	D	Application UART2 TX	AUART3_TX	CAN0_RX	EN-ET0_1588_EVENT0_IN
AUART3_CTS	AUART	D	Application UART2 CTS Flow Control	AUART3_CTS	CAN1_TX	EN-ET0_1588_EVENT1_OUT
AUART3_RX	AUART	D	Application UART2 RX	AUART3_RX	CAN0_TX	EN-ET0_1588_EVENT0_OUT
GPMI_CE2N	GPMI	D	NAND Chip Enable 2	GPMI_CE2N	CAN1_TX	ENET0_RX_ER
GPMI_RDY2	GPMI	D	NAND2 Ready/Busy#	GPMI_READY2	CAN0_TX	ENET0_TX_ER
GPMI_CE3N	GPMI	D	NAND Chip Enable 3	GPMI_CE3N	CAN1_RX	SAIF1_MCLK
GPMI_RDY3	GPMI	D	NAND3 Ready/Busy#	GPMI_READY3	CAN0_RX	HSADC_TRIGGER

Table 8-8. DUART

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
I2C0_SDA	I2C	D	I2C0 Serial Data	I2C0_SDA	TIMROT_ROTARYB	DUART_TX
I2C0_SCL	I2C	D	I2C0 Serial Clock	I2C0_SCL	TIMROT_ROTARYA	DUART_RX
AUART0_RX	AUART	D	Application UART0 RX	AUART0_RX	I2C0_SCL	DUART_CTS
AUART0_RTS	AUART	D	Application UART0 RTS Flow Control	AUART0_RTS	AUART4_TX	DUART_TX
AUART0_TX	AUART	D	Application UART0 TX	AUART0_TX	I2C0_SDA	DUART_RTS
AUART0_CTS	AUART	D	Application UART0 CTS Flow Control	AUART0_CTS	AUART4_RX	DUART_RX
PWM0	PWM	D		PWM_0	I2C1_SCL	DUART_RX
PWM1	PWM	D		PWM_1	I2C1_SDA	DUART_TX

Table 8-9. EMI

PIN NAME	GROUP	TYPE	DESCRIPTION
EMI_A13	EMI	E	EMI Address 13
EMI_A12	EMI	E	EMI Address 12
EMI_A08	EMI	E	EMI Address 8
EMI_A11	EMI	E	EMI Address 11
EMI_A09	EMI	E	EMI Address 9
EMI_A07	EMI	E	EMI Address 7
EMI_A04	EMI	E	EMI Address 4
EMI_A14	EMI	E	EMI Address 14
EMI_A06	EMI	E	EMI Address 6
EMI_A05	EMI	E	EMI Address 5
EMI_A03	EMI	E	EMI Address 3
EMI_CE0N	EMI	E	EMI CE0n
EMI_A00	EMI	E	EMI Address 0
EMI_A02	EMI	E	EMI Address 2
EMI_A01	EMI	E	EMI Address 1
EMI_CE1N	EMI	E	EMI CE1n
EMI_A10	EMI	E	EMI Address 10
EMI_BA1	EMI	E	EMI Bank Address 1
EMI_BA0	EMI	E	EMI Bank Address 0
EMI_ODT1	EMI	E	EMI DDR Byte1 ODT Enable
EMI_ODT0	EMI	E	EMI DDR Byte0 ODT Enable
EMI_RASN	EMI	E	EMI RASn
EMI_BA2	EMI	E	EMI Bank Address 2
EMI_CASN	EMI	E	EMI CASn
EMI_WEN	EMI	E	EMI WEn
EMI_CKE	EMI	E	EMI Clock Enable

PIN NAME	GROUP	TYPE	DESCRIPTION
EMI_D05	EMI	E	EMI Data 5
EMI_D02	EMI	E	EMI Data 2
EMI_D03	EMI	E	EMI Data 3
EMI_D04	EMI	E	EMI Data 4
EMI_D00	EMI	E	EMI Data 0
EMI_D01	EMI	E	EMI Data 1
EMI_DQM0	EMI	E	EMI DDR Data Mask 0 (Low Byte)
EMI_D06	EMI	E	EMI Data 6
EMI_D07	EMI	E	EMI Data 7
EMI_CLK	EMI	E	EMI Clock
EMI_CLKN	EMI	E	EMI Clock Invert
EMI_DQS0	EMI	E	EMI DDR Data Strobe 0 (Low Byte)
EMI_DQS0N	EMI	E	EMI DDR Data Strobe 0 Invert (Low Byte)
EMI_DQS1	EMI	E	EMI DDR Data Strobe 1
EMI_DQS1N	EMI	E	EMI DDR Data Strobe 1 Invert
EMI_DDR_OPEN_FB	EMI	E	EMI DDR Echo Gating feedback
EMI_DDR_OPEN	EMI	E	EMI DDR DDR Echo Gating
EMI_D13	EMI	E	EMI Data 13
EMI_D10	EMI	E	EMI Data 10
EMI_D11	EMI	E	EMI Data 11
EMI_D12	EMI	E	EMI Data 12
EMI_D08	EMI	E	EMI Data 8
EMI_D09	EMI	E	EMI Data 9
EMI_D15	EMI	E	EMI Data 15
EMI_DQM1	EMI	E	EMI DDR Data Mask 1
EMI_D14	EMI	E	EMI Data 14

Table 8-10. ENET

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
EN-ET_CLK	ENET	D	reference clock	CLK_ENET		ENET1_RX_ER
SPDIF	SPDIF	D		SPDIF_TX		ENET1_RX_ER
SSP3_SCK	SSP	D	SSP Serial Clock	SSP3_SCK	AUART4_TX	EN-ET1_1588_EVENT0_OUT
SSP3_MISO	SSP	D	SD/MMC/Data0 / SPI MISO	SSP3_D0	AUART4_RTS	EN-ET1_1588_EVENT1_OUT
SSP1_SCK	SSP	D	SSP Serial Clock	SSP1_SCK	SSP2_D1	EN-ET0_1588_EVENT2_OUT
SSP3_MOSI	SSP	D	SD/MMC CMD / SPI MOSI	SSP3_CMD	AUART4_RX	EN-ET1_1588_EVENT0_IN
SSP1_CMD	SSP	D	SD/MMC CMD / SPI MOSI	SSP1_CMD	SSP2_D2	EN-ET0_1588_EVENT2_IN
SSP3_SS0	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP3_D3	AUART4_CTS	EN-ET1_1588_EVENT1_IN
SSP1_DATA0	SSP	D	SD/MMC/ Data0 / SPI MISO	SSP1_D0	SSP2_D6	EN-ET0_1588_EVENT3_OUT
SSP1_DATA3	SSP	D	SD/MMC/ Data3 / SPI Slave Select 0	SSP1_D3	SSP2_D7	EN-ET0_1588_EVENT3_IN
EN-ET0_RX_EN	ENET	D	DataValid/CRS_DV carrier sense	ENET0_RX_EN	GPMI_CE6N	SAIF1_SDATA1
EN-ET0_TX_CLK	ENET	D	transmit clock	ENET0_TX_CLK	HSADC_TRIGGER	EN-ET0_1588_EVENT2_OUT
EN-ET0_TX_EN	ENET	D	Transmit data valid	ENET0_TX_EN	GPMI_READY5	
EN-ET0_RX_CLK	ENET	D	receive clock	ENET0_RX_CLK	ENET0_RX_ER	EN-ET0_1588_EVENT2_IN
EN-ET0_TXD1	ENET	D	Transmit DATA1	ENET0_TXD1	GPMI_READY7	
EN-ET0_TXD0	ENET	D	Transmit DATA0	ENET0_TXD0	GPMI_READY6	
EN-ET0_MDC	ENET	D		ENET0_MDC	GPMI_CE4N	SAIF0_SDATA1
EN-ET0_TXD3	ENET	D	Transmit DATA3	ENET0_TXD3	ENET1_TXD1	EN-ET0_1588_EVENT1_IN

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
EN-ET0_TXD2	ENET	D	Transmit DATA2	ENET0_TXD2	ENET1_TXD0	EN-ET0_1588_EVENT1_OUT
EN-ET0_MDIO	ENET	D	Management data	ENET0_MDIO	GPMI_CE5N	SAIF0_SDATA2
EN-ET0_RXD1	ENET	D		ENET0_RXD1	GPMI_READY4	
EN-ET0_RXD0	ENET	D	Receive DATA0	ENET0_RXD0	GPMI_CE7N	SAIF1_SDATA2
EN-ET0_COL	ENET	D	collision detect	ENET0_COL	ENET1_TX_EN	EN-ET0_1588_EVENT3_OUT
EN-ET0_CRS	ENET	D	carrier sense	ENET0_CRS	ENET1_RX_EN	EN-ET0_1588_EVENT3_IN
EN-ET0_RXD3	ENET	D	Receive DATA3	ENET0_RXD3	ENET1_RXD1	EN-ET0_1588_EVENT0_IN
EN-ET0_RXD2	ENET	D	ENET0_RXD2	ENET0_RXD2	ENET1_RXD0	EN-ET0_1588_EVENT0_OUT
AUART3_RTS	AUART	D	Application UART2 RTS Flow Control	AUART3_RTS	CAN1_RX	EN-ET0_1588_EVENT1_IN
AUART3_TX	AUART	D	Application UART2 TX	AUART3_TX	CAN0_RX	EN-ET0_1588_EVENT0_IN
AUART3_CTS	AUART	D	Application UART2 CTS Flow Control	AUART3_CTS	CAN1_TX	EN-ET0_1588_EVENT1_OUT
AUART3_RX	AUART	D	Application UART2 RX	AUART3_RX	CAN0_TX	EN-ET0_1588_EVENT0_OUT
LCD_D20	LCD	D	LCD Interface Data 20	LCD_D20	EN-ET1_1588_EVENT2_OUT	ETM_DA3
LCD_D23	LCD	D	LCD Interface Data 23	LCD_D23	EN-ET1_1588_EVENT3_IN	ETM_DA0
LCD_D22	LCD	D	LCD Interface Data 22	LCD_D22	EN-ET1_1588_EVENT3_OUT	ETM_DA1
LCD_D21	LCD	D	LCD Interface Data 21	LCD_D21	EN-ET1_1588_EVENT2_IN	ETM_DA2
GPMI_CE2N	GPMI	D	NAND Chip Enable 2	GPMI_CE2N	CAN1_TX	ENET0_RX_ER
GPMI_RDY2	GPMI	D	NAND2 Ready/Busy#	GPMI_READY2	CAN0_TX	ENET0_TX_ER

Table 8-11. ETM

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
LCD_D01	LCD	D	LCD Interface Data 01	LCD_D1		ETM_DA1
LCD_D00	LCD	D	LCD Interface Data 00	LCD_D0		ETM_DA0
LCD_WR_RWN	LCD	D	LCD Interface 6800 R/W_ / 8080 W	LCD_WR_RWN	LCD_HSYNC	ETM_TCLK
LCD_D03	LCD	D	LCD Interface Data 03	LCD_D3	ETM_DA8	ETM_DA3
LCD_D02	LCD	D	LCD Interface Data 02	LCD_D2		ETM_DA2
LCD_D05	LCD	D	LCD Interface Data 05	LCD_D5		ETM_DA5
LCD_D04	LCD	D	LCD Interface Data 04	LCD_D4	ETM_DA9	ETM_DA4
LCD_HSYNC	LCD	D	LCD0 Horizontal Sync	LCD_HSYNC	SAIF1_SDATA1	ETM_TCTL
LCD_D06	LCD	D	LCD Interface Data 06	LCD_D6		ETM_DA6
LCD_DOTCLK	LCD	D	LCD Interface DOT clock	LCD_DOTCLK	SAIF1_MCLK	ETM_TCLK
LCD_RD_E	LCD	D	LCD Interface 6800 Enable / 8080 RD	LCD_RD_E	LCD_VSYNC	ETM_TCTL
LCD_D09	LCD	D	LCD Interface Data 09	LCD_D9	ETM_DA4	ETM_DA9
LCD_D08	LCD	D	LCD Interface Data 08	LCD_D8	ETM_DA3	ETM_DA8
LCD_D07	LCD	D	LCD Interface Data 07	LCD_D7		ETM_DA7
LCD_D17	LCD	D	LCD Interface Data 17	LCD_D17		ETM_DA6
LCD_D11	LCD	D	LCD Interface Data 11	LCD_D11		ETM_DA11
LCD_D10	LCD	D	LCD Interface Data 10	LCD_D10		ETM_DA10
LCD_D16	LCD	D	LCD Interface Data 16	LCD_D16		ETM_DA7
LCD_D13	LCD	D	LCD Interface Data 13	LCD_D13		ETM_DA13
LCD_D12	LCD	D	LCD Interface Data 12	LCD_D12		ETM_DA12
LCD_D14	LCD	D	LCD Interface Data 14	LCD_D14		ETM_DA14
LCD_D15	LCD	D	LCD Interface Data 15	LCD_D15		ETM_DA15
LCD_D20	LCD	D	LCD Interface Data 20	LCD_D20	EN-ET1_1588_EVENT2_OUT	ETM_DA3
LCD_D19	LCD	D	LCD Interface Data 19	LCD_D19		ETM_DA4
LCD_D18	LCD	D	LCD Interface Data 18	LCD_D18		ETM_DA5

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
LCD_D23	LCD	D	LCD Interface Data 23	LCD_D23	EN-ET1_1588_EVENT3_IN	ETM_DA0
LCD_D22	LCD	D	LCD Interface Data 22	LCD_D22	EN-ET1_1588_EVENT3_OUT	ETM_DA1
LCD_D21	LCD	D	LCD Interface Data 21	LCD_D21	EN-ET1_1588_EVENT2_IN	ETM_DA2

Table 8-12. GPMI

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
ENET0_RX_EN	ENET	D		ENET0_RX_EN	GPMI_CE6N	SAIF1_SDATA1
ENET0_TX_EN	ENET	D	RMII	ENET0_TX_EN	GPMI_READY5	
ENET0_TXD1	ENET	D		ENET0_TXD1	GPMI_READY7	
ENET0_TXD0	ENET	D		ENET0_TXD0	GPMI_READY6	
ENET0_MDC	ENET	D	ENET0_MDC	ENET0_MDC	GPMI_CE4N	SAIF0_SDATA1
ENET0_MDIO	ENET	D		ENET0_MDIO	GPMI_CE5N	SAIF0_SDATA2
ENET0_RXD1	ENET	D	Receive DATA1	ENET0_RXD1	GPMI_READY4	
ENET0_RXD0	ENET	D		ENET0_RXD0	GPMI_CE7N	SAIF1_SDATA2
GPMI_RDY0	GPMI	D	NAND0 Ready/Busy#	GPMI_READY0	SSP1_CARD_DE-TECT	USB0_ID
GPMI_ALE	GPMI	D	NAND ALE	GPMI_ALE	SSP3_D1	SSP3_D4
GPMI_RDN	GPMI	D	NAND Read Strobe	GPMI_RDN	SSP3_SCK	
GPMI_D07	GPMI	D	NAND Data 7	GPMI_D7	SSP1_D7	
GPMI_D06	GPMI	D	NAND Data 6	GPMI_D6	SSP1_D6	
GPMI_CE2N	GPMI	D	NAND Chip Enable 2	GPMI_CE2N	CAN1_TX	ENET0_RX_ER
GPMI_CE0N	GPMI	D	NAND Chip Enable 0	GPMI_CE0N	SSP3_D0	
GPMI_CLE	GPMI	D	NAND CLE	GPMI_CLE	SSP3_D2	SSP3_D5
GPMI_D05	GPMI	D	NAND Data 5	GPMI_D5	SSP1_D5	
GPMI_D04	GPMI	D	NAND Data 4	GPMI_D4	SSP1_D4	
GPMI_D03	GPMI	D	NAND Data 3	GPMI_D3	SSP1_D3	
GPMI_RDY2	GPMI	D	NAND2 Ready/Busy#	GPMI_READY2	CAN0_TX	ENET0_TX_ER

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
GPMI_RDY1	GPMI	D	NAND1 Ready/Busy#	GPMI_READY1	SSP1_CMD	
GPMI_WRN	GPMI	D	NAND Write Strobe	GPMI_WRN	SSP1_SCK	
GPMI_D02	GPMI	D	NAND Data 2	GPMI_D2	SSP1_D2	
GPMI_D01	GPMI	D	NAND Data 1	GPMI_D1	SSP1_D1	
GPMI_D00	GPMI	D	NAND Data 0	GPMI_D0	SSP1_D0	
GPMI_RESETN	GPMI	D	NAND Write Protect	GPMI_RESETN	SSP3_CMD	
GPMI_CE3N	GPMI	D	NAND Chip Enable 3	GPMI_CE3N	CAN1_RX	SAIF1_MCLK
GPMI_CE1N	GPMI	D	NAND Chip Enable 1	GPMI_CE1N	SSP3_D3	
GPMI_RDY3	GPMI	D	NAND3 Ready/Busy#	GPMI_READY3	CAN0_RX	HSADC_TRIG-GER

Table 8-13. HSADC

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
ENET0_TX_CLK	ENET	D		ENET0_TX_CLK	HSADC_TRIG-GER	EN-ET0_1588_EVENT2_OUT
GPMI_RDY3	GPMI	D	NAND3 Ready/Busy#	GPMI_READY3	CAN0_RX	HSADC_TRIGGER

Table 8-14. I2C

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
I2C0_SDA	I2C	D	I2C0 Serial Data	I2C0_SDA	TIM-ROT_ROTARYB	DUART_TX
I2C0_SCL	I2C	D	I2C0 Serial Clock	I2C0_SCL	TIM-ROT_ROTARYA	DUART_RX
AUART2_RTS	AUART	D	Application UART2 RTS Flow Control	AUART2_RTS	I2C1_SDA	SAIF1_LRCLK
AUART2_CTS	AUART	D	Application UART2 CTS Flow Control	AUART2_CTS	I2C1_SCL	SAIF1_BITCLK
AUART0_RX	AUART	D	Application UART0 RX	AUART0_RX	I2C0_SCL	DUART_CTS
AUART0_TX	AUART	D	Application UART0 TX	AUART0_TX	I2C0_SDA	DUART_RTS
PWM0	PWM	D		PWM_0	I2C1_SCL	DUART_RX
PWM1	PWM	D		PWM_1	I2C1_SDA	DUART_TX

Table 8-15. JTAG

PIN NAME	GROUP	TYPE	DESCRIPTION
DEBUG	SYSTEM	D	ARM jtag chain/Boundary scan chain selection
TESTMODE	SYSTEM	D	Test Mode Pin
JTAG_RTCK	SYSTEM	D	JTAG feedback clock (only for ARM ICE)
JTAG_TRST	SYSTEM	D	JTAG Reset
JTAG_TDO	SYSTEM	D	JTAG Serial Data Out
JTAG_TMS	SYSTEM	D	JTAG Test Mode Select
JTAG_TDI	SYSTEM	D	JTAG Serial Data In
JTAG_TCK	SYSTEM	D	JTAG Clock

Table 8-16. LCD

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
LCD_D01	LCD	D	LCD Interface Data 01	LCD_D1		ETM_DA1
LCD_D00	LCD	D	LCD Interface Data 00	LCD_D0		ETM_DA0
LCD_WR_RWN	LCD	D	LCD Interface 6800 R/W_ / 8080 W	LCD_WR_RWN	LCD_HSYNC	ETM_TCLK
LCD_D03	LCD	D	LCD Interface Data 03	LCD_D3	ETM_DA8	ETM_DA3
LCD_D02	LCD	D	LCD Interface Data 02	LCD_D2		ETM_DA2
LCD_VSYNC	LCD	D	LCD Interface Vertical Sync	LCD_VSYNC	SAIF1_SDATA0	
LCD_RS	LCD	D	LCD Interface Re-gister Select	LCD_RS	LCD_DOTCLK	
LCD_D05	LCD	D	LCD Interface Data 05	LCD_D5		ETM_DA5
LCD_D04	LCD	D	LCD Interface Data 04	LCD_D4	ETM_DA9	ETM_DA4
LCD_HSYNC	LCD	D	LCD0 Horizontal Sync	LCD_HSYNC	SAIF1_SDATA1	ETM_TCTL
LCD_D06	LCD	D	LCD Interface Data 06	LCD_D6		ETM_DA6

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
LCD_DOTCLK	LCD	D	LCD Interface DOT clock	LCD_DOTCLK	SAIF1_MCLK	ETM_TCLK
LCD_ENABLE	LCD	D	LCD Interface Enable	LCD_ENABLE		
LCD_RD_E	LCD	D	LCD Interface 6800 Enable / 8080 RD	LCD_RD_E	LCD_VSYNC	ETM_TCTL
LCD_D09	LCD	D	LCD Interface Data 09	LCD_D9	ETM_DA4	ETM_DA9
LCD_D08	LCD	D	LCD Interface Data 08	LCD_D8	ETM_DA3	ETM_DA8
LCD_D07	LCD	D	LCD Interface Data 07	LCD_D7		ETM_DA7
LCD_D17	LCD	D	LCD Interface Data 17	LCD_D17		ETM_DA6
LCD_D11	LCD	D	LCD Interface Data 11	LCD_D11		ETM_DA11
LCD_D10	LCD	D	LCD Interface Data 10	LCD_D10		ETM_DA10
LCD_D16	LCD	D	LCD Interface Data 16	LCD_D16		ETM_DA7
LCD_D13	LCD	D	LCD Interface Data 13	LCD_D13		ETM_DA13
LCD_D12	LCD	D	LCD Interface Data 12	LCD_D12		ETM_DA12
LCD_D14	LCD	D	LCD Interface Data 14	LCD_D14		ETM_DA14
LCD_D15	LCD	D	LCD Interface Data 15	LCD_D15		ETM_DA15
LCD_D20	LCD	D	LCD Interface Data 20	LCD_D20	EN-ET1_1588_EVENT2_OUT	ETM_DA3
LCD_D19	LCD	D	LCD Interface Data 19	LCD_D19		ETM_DA4
LCD_D18	LCD	D	LCD Interface Data 18	LCD_D18		ETM_DA5
LCD_CS	LCD	D	LCD Interface Chip Select	LCD_CS	LCD_ENABLE	
LCD_D23	LCD	D	LCD Interface Data 23	LCD_D23	EN-ET1_1588_EVENT3_IN	ETM_DA0

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
LCD_D22	LCD	D	LCD Interface Data 22	LCD_D22	EN-ET1_1588_EVENT3_OUT	ETM_DA1
LCD_D21	LCD	D	LCD Interface Data 21	LCD_D21	EN-ET1_1588_EVENT2_IN	ETM_DA2
LCD_RESET	LCD	D	LCD Interface Reset Out	LCD_RESET	LCD_VSYNC	

Table 8-17. PWM

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SAIF1_SDATA0	SAIF	D	SAIF1 Serial Data0 (stereo)	SAIF1_SDATA0	PWM_7	SAIF0_SDATA1
PWM3	PWM	D		PWM_3		
PWM4	PWM	D		PWM_4		
SAIF0_SDATA0	SAIF	D	SAIF0 Serial Data0 (stereo)	SAIF0_SDATA0	PWM_6	AUART4_TX
SAIF0_BITCLK	SAIF	D	SAIF0 BIT clock	SAIF0_BITCLK	PWM_5	AUART4_RX
SAIF0_MCLK	SAIF	D	SAIF0 Master Clock	SAIF0_MCLK	PWM_3	AUART4_CTS
SAIF0_LRCLK	SAIF	D	SAIF0 Left/Right Clock	SAIF0_LRCLK	PWM_4	AUART4_RTS
PWM0	PWM	D		PWM_0	I2C1_SCL	DUART_RX
AUART1_TX	AUART	D	Application UART1 TX	AUART1_TX	SSP3_CARD_DE-TECT	PWM_1
AUART1_RX	AUART	D	Application UART1 RX	AUART1_RX	SSP2_CARD_DE-TECT	PWM_0
PWM1	PWM	D		PWM_1	I2C1_SDA	DUART_TX
PWM2	PWM	D		PWM_2	USB0_ID	USB1_OVER-CURRENT

Table 8-18. AUDIO(SAIF/SPDIF)

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SAIF1_SDATA0	SAIF	D	SAIF1 Serial Data0 (stereo)	SAIF1_SDATA0	PWM_7	SAIF0_SDATA1
SAIF0_SDATA0	SAIF	D	SAIF0 Serial Data0 (stereo)	SAIF0_SDATA0	PWM_6	AUART4_TX
SSP2_SCK	SSP	D	SSP Serial Clock	SSP2_SCK	AUART2_RX	SAIF0_SDATA1
SSP2_MISO	SSP	D	SD/MMC/Data0 / SPI MISO	SSP2_D0	AUART3_RX	SAIF1_SDATA1

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SSP2_SS0	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP2_D3	AUART3_TX	SAIF1_SDATA2
SAIF0_BITCLK	SAIF	D		SAIF0_BITCLK	PWM_5	AUART4_RX
SSP2_MOSI	SSP	D	SD/MMC CMD / BS / SPI MOSI	SSP2_CMD	AUART2_TX	SAIF0_SDATA2
SAIF0_MCLK	SAIF	D		SAIF0_MCLK	PWM_3	AUART4_CTS
SAIF0_LRCLK	SAIF	D	SAIF0 Left/Right Clock	SAIF0_LRCLK	PWM_4	AUART4_RTS
ENET0_RX_EN	ENET	D		ENET0_RX_EN	GPMI_CE6N	SAIF1_SDATA1
AUART2_RTS	AUART	D	Application UART2 RTS Flow Control	AUART2_RTS	I2C1_SDA	SAIF1_LRCLK
AUART2_CTS	AUART	D	Application UART2 CTS Flow Control	AUART2_CTS	I2C1_SCL	SAIF1_BITCLK
ENET0_MDC	ENET	D		ENET0_MDC	GPMI_CE4N	SAIF0_SDATA1
ENET0_MDIO	ENET	D		ENET0_MDIO	GPMI_CE5N	SAIF0_SDATA2
ENET0_RXD0	ENET	D		ENET0_RXD0	GPMI_CE7N	SAIF1_SDATA2
LCD_VSYNC	LCD	D	LCD Interface Vertical Sync	LCD_VSYNC	SAIF1_SDATA0	
LCD_HSYNC	LCD	D	LCD0 Horizontal Sync	LCD_HSYNC	SAIF1_SDATA1	ETM_TCTL
LCD_DOTCLK	LCD	D	LCD Interface DOT clock	LCD_DOTCLK	SAIF1_MCLK	ETM_TCLK
GPMI_CE3N	GPMI	D	NAND Chip Enable 3	GPMI_CE3N	CAN1_RX	SAIF1_MCLK
SPDIF	SPDIF	D		SPDIF_TX		ENET1_RX_ER

Table 8-19. SSP

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SSP0_DE-TECT	SSP	D	Removable Card Detect	SSP0_CARD_DE-TECT		
SSP0_SCK	SSP	D	SSP Serial Clock	SSP0_SCK		
SSP0_DATA0	SSP	D	SD/MMC/Data0 / SPI MISO	SSP0_D0		
SSP0_DATA3	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP0_D3		
SSP0_DATA4	SSP	D	SD/MMC/Data4 / SPI Slave Select 1	SSP0_D4	SSP2_D0	
SSP0_DATA1	SSP	D	SD/MMC/ Data1 / Winbond Write Protect_	SSP0_D1		

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
SSP0_CMD	SSP	D	SD/MMC CMD / SPI MOSI	SSP0_CMD		
SSP2_SCK	SSP	D	SSP Serial Clock	SSP2_SCK	AUART2_RX	SAIF0_SDATA1
SSP0_DATA7	SSP	D	SD/MMC/Data7	SSP0_D7	SSP2_SCK	
SSP0_DATA5	SSP	D	SD/MMC/Data5 / SPI Slave Select 2	SSP0_D5	SSP2_D3	
SSP0_DATA2	SSP	D	SD/MMC/Data2 / Winbond Hold_	SSP0_D2		
SSP3_SCK	SSP	D	SSP Serial Clock	SSP3_SCK	AUART4_TX	EN-ET1_1588_EVENT0_OUT
SSP2_MISO	SSP	D	SD/MMC/Data0 / SPI MISO	SSP2_D0	AUART3_RX	SAIF1_SDATA1
SSP2_SS0	SSP	D	SD/MMC/Memstick Data3 / SPI Slave Select 0	SSP2_D3	AUART3_TX	SAIF1_SDATA2
SSP0_DATA6	SSP	D	SD/MMC/Memstick Data6	SSP0_D6	SSP2_CMD	
SSP3_MISO	SSP	D	SD/MMC/Memstick Data0 / SPI MISO	SSP3_D0	AUART4_RTS	EN-ET1_1588_EVENT1_OUT
SSP2_MOSI	SSP	D	SD/MMC CMD / Memstick BS / SPI MOSI	SSP2_CMD	AUART2_TX	SAIF0_SDATA2
SSP2_SS2	SSP	D	SD/MMC/Memstick Data0 / SPI MISO	SSP2_D5	SSP2_D2	USB0_OVERCUR-RENT
AUART2_RX	AUART	D		AUART2_RX	SSP3_D1	SSP3_D4
SSP1_SCK	SSP	D	SSP Serial Clock	SSP1_SCK	SSP2_D1	EN-ET0_1588_EVENT2_OUT
SSP3_MOSI	SSP	D	SD/MMC CMD / SPI MOSI	SSP3_CMD	AUART4_RX	EN-ET1_1588_EVENT0_IN
SSP2_SS1	SSP	D	SD/MMC/Data4 / SPI Slave Select 1	SSP2_D4	SSP2_D1	USB1_OVERCUR-RENT
SSP1_CMD	SSP	D	SD/MMC CMD / SPI MOSI	SSP1_CMD	SSP2_D2	EN-ET0_1588_EVENT2_IN
SSP3_SS0	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP3_D3	AUART4_CTS	EN-ET1_1588_EVENT1_IN
SSP1_DATA0	SSP	D	SD/MMC/Data0 / SPI MISO	SSP1_D0	SSP2_D6	EN-ET0_1588_EVENT3_OUT
SSP1_DATA3	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP1_D3	SSP2_D7	EN-ET0_1588_EVENT3_IN
AUART2_TX	AUART	D	Application UART2 TX	AUART2_TX	SSP3_D2	SSP3_D5

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
AUART1_TX	AUART	D	Application UART1 TX	AUART1_TX	SSP3_CARD_DE-TECT	PWM_1
AUART1_RX	AUART	D	Application UART1 RX	AUART1_RX	SSP2_CARD_DE-TECT	PWM_0
GPMI_RDY0	GPMI	D	NAND0 Ready/Busy#	GPMI_READY0	SSP1_CARD_DE-TECT	USB0_ID
GPMI_ALE	GPMI	D	NAND ALE	GPMI_ALE	SSP3_D1	SSP3_D4
GPMI_RDN	GPMI	D	NAND Read Strobe	GPMI_RDN	SSP3_SCK	
GPMI_D07	GPMI	D	NAND Data 7	GPMI_D7	SSP1_D7	
GPMI_D06	GPMI	D	NAND Data 6	GPMI_D6	SSP1_D6	
GPMI_CE0N	GPMI	D	NAND Chip Enable 0	GPMI_CE0N	SSP3_D0	
GPMI_CLE	GPMI	D	NAND CLE	GPMI_CLE	SSP3_D2	SSP3_D5
GPMI_D05	GPMI	D	NAND Data 5	GPMI_D5	SSP1_D5	
GPMI_D04	GPMI	D	NAND Data 4	GPMI_D4	SSP1_D4	
GPMI_D03	GPMI	D	NAND Data 3	GPMI_D3	SSP1_D3	
GPMI_RDY1	GPMI	D	NAND1 Ready/Busy#	GPMI_READY1	SSP1_CMD	
GPMI_WRN	GPMI	D	NAND Write Strobe	GPMI_WRN	SSP1_SCK	
GPMI_D02	GPMI	D	NAND Data 2	GPMI_D2	SSP1_D2	
GPMI_D01	GPMI	D	NAND Data 1	GPMI_D1	SSP1_D1	
GPMI_D00	GPMI	D	NAND Data 0	GPMI_D0	SSP1_D0	
GPMI_RE-SETN	GPMI	D	NAND Write Protect	GPMI_RESETN	SSP3_CMD	
GPMI_CE1N	GPMI	D	NAND Chip Enable 1	GPMI_CE1N	SSP3_D3	

Table 8-20. TIMROT

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
I2C0_SDA	I2C	D	I2C0 Serial Data	I2C0_SDA	TIM-ROT_ROTARYB	DUART_TX
I2C0_SCL	I2C	D	I2C0 Serial Clock	I2C0_SCL	TIM-ROT_ROTARYA	DUART_RX
AUART1_RTS	AUART	D	Application UART1 RTS Flow Control	AUART1_RTS	USB0_ID	TIM-ROT_ROTARYB

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
AUART1_CTS	AUART	D	Application UART1 CTS Flow Control	AUART1_CTS	USB0_OVERCURRENT	TIMROT_ROTARYA

Table 8-21. USB

PIN NAME	GROUP	TYPE	DESCRIPTION	MUX0	MUX1	MUX2
USB0DP	USB	A	USB0 Positive Data Line			
USB0DM	USB	A	USB0 Negative Data Line			
USB1DP	USB	A	USB1 Positive Data Line			
USB1DM	USB	A	USB1 Negative Data Line			
SSP2_SS2	SSP	D	SD/MMC/Data3 / SPI Slave Select 0	SSP2_D5	SSP2_D2	USB0_OVERCURRENT
SSP2_SS1	SSP	D	SD/MMC/Data0 / SPI MISO	SSP2_D4	SSP2_D1	USB1_OVERCURRENT
AUART1_RTS	AUART	D	Application UART1 RTS Flow Control	AUART1_RTS	USB0_ID	TIMROT_ROTARYB
AUART1_CTS	AUART	D	Application UART1 CTS Flow Control	AUART1_CTS	USB0_OVERCURRENT	TIMROT_ROTARYA
PWM2	PWM	D		PWM_2	USB0_ID	USB1_OVERCURRENT
GPMI_RDY0	GPMI	D	NAND0 Ready/Busy#	GPMI_READY0	SSP1_CARD_DETECT	USB0_ID

Chapter 9
Pin Control and GPIO (PinCtrl)

9.1 Pin Control and GPIO Overview

The i.MX28 digital interface pins have the following features: (In the context of this chapter, "digital pin" means the standard digital interface pins. This does not include test pins used for boundary scan control.)

- The device has seven banks of pins, Banks **0~4** serve as GPIOs. Banks **5** and **6** contain the EMI sixteen data pins and EMI control/address signals, they are not multiplexed with other functions since all the use cases require at least sixteen bit external memory.
- Each GPIO pin has separate control on **voltage**(1.8 V/3.3 V), Interrupt (trigger type/polarity).
- All non-EMI digital pins have selectable output **drive strengths** as described in Pin Drive Strength Selection.
- Each group of EMI data[7:0], data[15:8], control pins and address pins, dual pads (clk/clkn/dqs#/dqs#n), have selectable output **drive strengths**.
- All digital pins have weak internal **keepers** to minimize power loss due to undriven pins.
- All EMI pins' internal **keepers** can be disabled to allow them to change to a high-impedance state (as required by some DRAM manufacturers).
- The following pin interfaces have selectable **pull up** resistors:
    - SSP data - 47 kΩ
    - SSP command/detect - 10 kΩ
    - GPMI chip enable - 47 kΩ
    - GPMI ready/busy - 10 kΩ
- The following pin interfaces are slow transitioning pins with internal Schmidt Triggers for noise immunity:

- I2C SCL
- I2C SDA

9.2 Operation

Each individual digital pin supporting the GPIO operation may be dynamically programmed at any time to be in one of the following states:

- High-impedance (for input, three-state, or open-drain applications)
- Low
- High
- Controlled by one to three selectable on-chip peripheral module interfaces, on a pin by pin basis, as described in GPIO Interface.

All non-EMI digital pins can be programmed for 1.8 or 3.3 V operation. All EMI pads run at 1.8 V/1.5 V.

Selected pins have pullups that can be configured using register settings. When pullups are enabled, the pin's weak keeper devices are disabled.

Additionally, the state of each pin may be read at any time (no matter how it is configured), and its drive strength may be configured as described in Pin Drive Strength Selection.

Each GPIO pin may also be used as an interrupt input and the interrupt trigger type may be configured to be low level-sensitive, high level-sensitive, rising edge-sensitive, or falling edge-sensitive.

The following sections show how to use all the features of each pin.

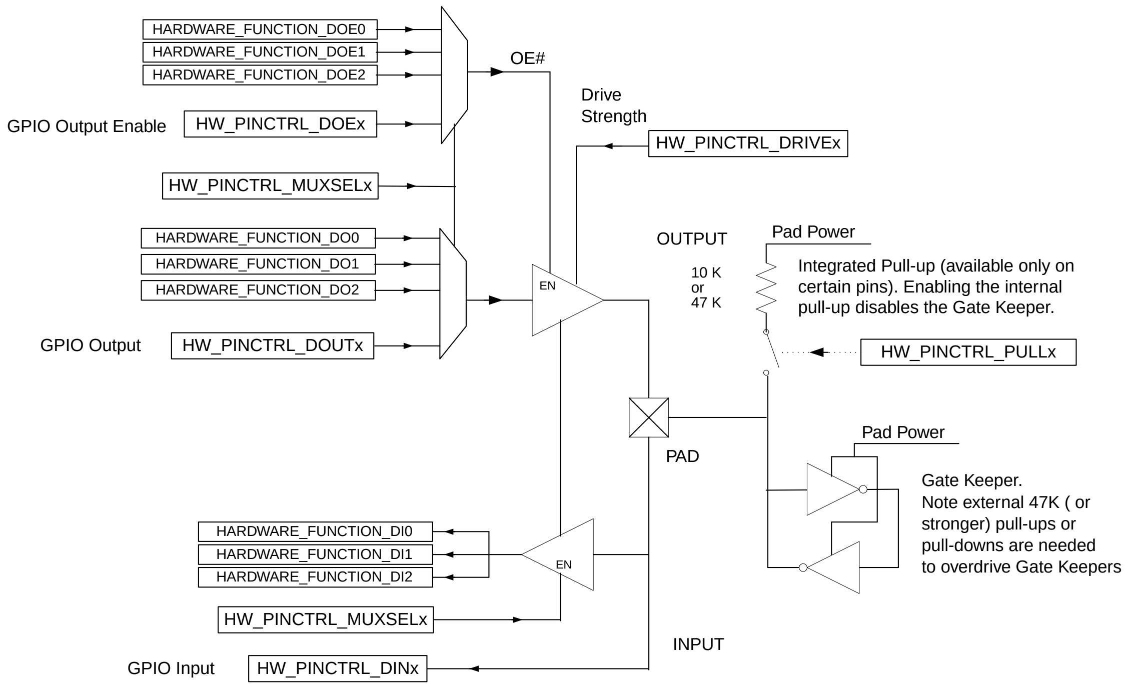


Figure 9-1. Pad Diagram

9.2.1 Reset Configuration

Out of reset (hardware/software reset), all the pins (except JTAG related) are configured as GPIO inputs with gate keepers enabled.

Here are some exceptions on general reset behavior:

- Ethernet pins works differently, they will only reset when power is on. Software reset will not affect them. This is to make sure that the enet switch can still work during software reset.
- EMI pins do not mux with GPIO function. So after reset, configure as GPIO means it is disabled.

.

9.2.2 Pin Interface Multiplexing

The device is designed for cost sensitive applications. It contains a rich set of specialized hardware interfaces (mDDR, NAND Flash, LCD panels, many types of insertable media, and so on), but does not have enough pins to allow use of all signals of all the interfaces simultaneously. Consequently, a pin multiplexing scheme is used to allow customers to choose which specialized interfaces to enable for their application. In addition to these specialized hardware interfaces, the device allows many digital pins to be used as GPIOs.

This capability supports custom interfacing requirements, such as the ability to communicate with LEDs, digital buttons, and other devices that are not directly supported by any of the i.MX28 specialized hardware interfaces.

Each pin is connected to one, two, or three specialized hardware interfaces, in addition to the GPIO function available on banks 0 through 4. The description of each pin in and contains full details of which specialized hardware interfaces are attached to that pin. For example, the package pin named PWM0 is shared between the PWM and debug UART hardware interfaces.

Users define which of the available hardware interfaces controls each pin by writing a two-bit field for that pin into one of the HW_PINCTRL_MUXSELx registers.

Table 9-1– Table 9-2 illustrate the pin multiplexing on the device.

Both 289 and 204 packages have same mux options, but some pins on 289 package are not available on 204 package and caused those related mux unavailable too. These unavailable pins for 204-pins package are marked gray on muxreg row in .

- Table 9-1 shows the color mapping used in the tables.
- Table 9-2 shows the multiplexing used in both the package.

Table 9-1. Color Mapping for Pin Control Bank Tables

EMI	GPIO	GPMI	LCD	SSP	ENET	USB	
ETM	DUART	AUART	I2C	PWM	SAIF	SPDIF	JTAG

Table 9-2. Pin Multiplexing for the 289-pin Package

Bank 0	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Mux Reg 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
se-lect = 00																	gpmi_d7		gpmi_d6_gpmi_d7		gpmi_d5		gpmi_d4		gpmi_d3		gpmi_d2		gpmi_d1		gpmi_d0	
se-lect = 01																	ssp1_d7		ssp1_d6		ssp1_d5		ssp1_d4		ssp1_d3		ssp1_d2		ssp1_d1		ssp1_d0	

select = 10																
select = 11									GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

Bank 0	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 1	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
select = 00				gpmi_resetn	gpmi_cle	gpmi_ale	gpmi_wrn	gpmi_rdn	gpmi_ready3	gpmi_ready2	gpmi_ready1	gpmi_ready0	gpmi_ce3n	gpmi_ce2n	gpmi_ce1n	gpmi_ce0n
select = 01				ssp3_cmd	ssp3_d2	ssp3_d1	ssp1_sck	ssp3_sck	can0_rx	can0_tx	ssp1_cmd	ssp1_card_detect	can1_rx	can1_tx	ssp3_d3	ssp3_d0
select = 10					ssp3_d5	ssp3_d4			hsadc_trigger	enet0_tx_er		usb0_id	saif1_mclk	enet0_rx_er		
select = 11				GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

Bank 1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 2	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
select = 00	lcd_d15	lcd_d14	lcd_d13	lcd_d12	lcd_d11	lcd_d10	lcd_d9	lcd_d8	lcd_d7	lcd_d6	lcd_d5	lcd_d4	lcd_d3	lcd_d2	lcd_d1	lcd_d0

se-lect = 01																
se-lect = 10	etm_da15	etm_da14	etm_da13	etm_da12	etm_da11	etm_da10	etm_da9	etm_da8	etm_da7	etm_da5	etm_da6	etm_da4	etm_da3	etm_da2	etm_da1	etm_da0
se-lect = 11	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 1	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 3	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
se-lect = 00	lcd_enable	lcd_dotclk	lcd_hsync	lcd_vsync	lcd_cs	lcd_rs	lcd_wr_rwn	lcd_rd_e	lcd_d23	lcd_d22	lcd_d21	lcd_d20	lcd_d19	lcd_d18	lcd_d17	lcd_d16
se-lect = 01					lcd_enable	lcd_dotclk	lcd_hsync	lcd_vsync	enet1_1588_event3_out	enet1_1588_event3_in	enet1_1588_event2_out	enet1_1588_event2_in				
se-lect = 10		etm_tclk	etm_tclk				etm_tclk	etm_tctl	etm_da0	etm_da1	etm_da2	etm_da3	etm_da4	etm_da5	etm_da6	etm_da7
se-lect = 11	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 2	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 4	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0

se- lect = 00	ssp1_d3	ssp1_d0	ssp1_cmd	ssp1_sck		ssp0_sck	ssp0_card_detect	ssp0_cmd	ssp0_d7	ssp0_d6	ssp0_d5	ssp0_d4	ssp0_d3	ssp0_d2	ssp0_d1	ssp0_d0
se- lect = 01	ssp2_d7	ssp2_d6	ssp2_d2	ssp2_d1					ssp2_sck	ssp2_cmd	ssp2_d3	ssp2_d0				
se- lect = 10	enet0_1588_event3_in	enet0_1588_event3_out	enet0_1588_event2_in	enet0_1588_event2_out												
se- lect = 11	GPIO	GPIO	GPIO	GPIO		GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO
Bank 2	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 5	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
se- lect = 00					ssp3_d3	ssp3_d0	ssp3_cmd	ssp3_sck			ssp2_d5	ssp2_d4	ssp2_d3	ssp2_d0	ssp2_cmd	ssp2_sck
se- lect = 01					auart4_cts	auart4_rts	auart4_rx	auart4_tx			ssp2_d2	ssp2_d1	auart3_tx	auart3_rx	auart2_tx	auart2_rx
se- lect = 10					enet1_1588_event1_in	enet1_1588_event1_out	enet1_1588_event0_in	enet1_1588_event0_out			usb0_overcurrent	usb1_overcurrent	saif1_sdata2	saif1_sdata1	saif0_sdata2	saif0_sdata1

se-lect = 11					GPIO	GPIO	GPIO	GPIO			GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

Bank 3	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 6	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
se-lect = 00	auart3_rts	auart3_cts	auart3_tx	auart3_rx	auart2_rts	auart2_cts	auart2_tx	auart2_rx	auart1_rts	auart1_cts	auart1_tx	auart1_rx	auart0_rts	auart0_cts	auart0_tx	auart0_rx
se-lect = 01	can1_rx	can1_tx	can0_rx	can0_tx	i2c1_sda	i2c1_scl	ssp3_d2	ssp3_d1	usb0_id	usb0_overcurrent	ssp3_card_detect	ssp2_card_detect	auart4_tx	auart4_rx	i2c0_sda	i2c0_scl
se-lect = 10	enet0_1588_event1_in	enet0_1588_event1_out	enet0_1588_event0_in	enet0_1588_event0_out	saif1_lrclk	saif1_bitclk	ssp3_d5	ssp3_d4	timrot_rotaryb	timrot_rotarya	pwm_1	pwm_0	duart_tx	duart_rx	duart_rts	duart_cts
se-lect = 11	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

Bank 3	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Mux Reg 7	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
se-lect = 00		lcd_reset			spdif_tx	saif1_sdata0	i2c0_sda	i2c0_scl	saif0_sdata0	saif0_bitclk	saif0_lrclk	saif0_mclk		pwm_2	pwm_1	pwm_0

se-lect = 01		lcd_vsync				pwm_7	timrot_rotaryb	timrot_rotarya	pwm_6	pwm_5	pwm_4	pwm_3		usb0_id	i2c1_sda	i2c1_scl
se-lect = 10					enet1_rx_er	saif0_sdata1	duart_tx	duart_rx	auart4_tx	auart4_rx	auart4_rts	auart4_cts		usb1_overcurrent	duart_tx	duart_rx
se-lect = 11		GPIO			GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO		GPIO	GPIO	GPIO

Bank 4	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mux Reg 8	31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
se-lect = 00	enet0_crs	enet0_col	enet0_rx_clk	enet0_txd3	enet0_txd2	enet0_rxd3	enet0_rxd2	enet0_txd1	enet0_txd0	enet0_tx_en	enet0_tx_clk	enet0_rxd1	enet0_rxd0	enet0_rx_en	enet0_mdio	enet0_mdc
se-lect = 01	enet1_rx_en	enet1_tx_en	enet0_rx_er	enet1_txd1	enet1_txd0	enet1_rxd1	enet1_rxd0	gpmi_ready7	gpmi_ready6	gpmi_ready5	hsadc_trigger	gpmi_ready4	gpmi_ce7n	gpmi_ce6n	gpmi_ce5n	gpmi_ce4n
se-lect = 10	enet0_1588_event3_in	enet0_1588_event3_out	enet0_1588_event2_in	enet0_1588_event1_in	enet0_1588_event1_out	enet0_1588_event0_in	enet0_1588_event0_out				enet0_1588_event2_out		saif1_sdata2	saif1_sdata1	saif0_sdata2	saif0_sdata1
se-lect = 11	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO	GPIO

Bank 4	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

Mux Reg 9	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00																							jtag_rtck								clkctrl_enet	
select = 01																																
select = 10																																
select = 11	GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO		GPIO	
Bank 5	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	

Mux Reg 10	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00	emi_data15		emi_data14		emi_data13		emi_data12		emi_data11		emi_data10		emi_data9		emi_data8		emi_data7		emi_data6		emi_data5		emi_data4		emi_data3		emi_data2		emi_data1		emi_data0	
select = 01																																
select = 10																																
select = 11	disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled	
Bank 5	31		30		29		28		27		26		25		24		23		22		21		20		19		18		17		16	

Mux Reg 11	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
se- lect = 00											emi_ddr_open						emi_dqs1		emi_dqs0		emi_clk		emi_ddr_open_feedback		emi_dqm1		emi_odt1		emi_dqm0		emi_odt0	
se- lect = 01																																
se- lect = 10																																
se- lect = 11											dis- abled						dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled	

Bank 6	15		14		13		12		11		10		9		8		7		6		5		4		3		2		1		0	
Mux Reg 12	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
se- lect = 00			emi_addr14		emi_addr13		emi_addr12		emi_addr11		emi_addr10		emi_addr9		emi_addr8		emi_addr7		emi_addr6		emi_addr5		emi_addr4		emi_addr3		emi_addr2		emi_addr1		emi_addr0	
se- lect = 01																																
se- lect = 10																																
se- lect = 11			dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled		dis- abled	

Bank 6	31		30		29		28				26		25		24		23		22		21		20		19		18		17		16	
Mux Reg 13	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
select = 00															emi_cke		emi_ce1n		emi_ce0n		emi_wen		emi_rasn		emi_casn		emi_ba2		emi_ba1		emi_ba0	
select = 01																																
select = 10																																
select = 11															disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled		disabled	

Readback registers are never affected by the operation of the HW_PINCTRL_MUXSELx registers and always sense the actual value on the data pin.

For example, if a pin is programmed to be a GPIO output and then driven high, then any specialized hardware interfaces that are actively monitoring that pin will read the high logic value. Conversely, if the pin mux is programmed to give a specialized hardware interface such as the GPMI block control of a particular pin, the current state of that pin can be read through its GPIO read register at any time, even while active GPMI cycles are in progress. This is not true for the EMI pads that are GPIO capable due to the pad design.

Because the pin mux configuration is independent for each individual pin, many pins which are not required for a given active interface can be reused as GPIO pins. For example, the LCD_RESET pin can be configured and controlled as a GPIO pin, while the other LCD interface pins are still controlled by the LCDIF.

Banks 5 and 6 are for the EMI only and do not have GPIO capability.

9.2.2.1 Pin Drive Strength Selection

The drive strength for each digital pin can be programmed by setting the bit corresponding to that pin in one of the HW_PINCTRL_DRIVEx registers. All digital pins have selectable output drive strengths of 4, 8, and 12 mA, with the following exceptions:

- All clock pads have 8 and 16mA drive strength.
- All EMI pads have 5, 10 and 20 mA drive strengths.

Note

The HW_PINCTRL_DRIVEx registers must be configured prior to the operation of the pins and cannot be changed mid-course during active operation. Drive-strength options are provided to optimize simultaneous switching output (SSO) noise. The majority of GPIO pins must be programmed in 4-mA mode. For EMI pins, the weakest mode should be used as long as the timing is met.

Note

It is recommended that the drive strength of GPMI_RDn and GPMI_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of GPMI_RDn and GPMI_WRn.

9.2.2.2 Pin Voltage Selection

Each GPIO (non-EMI) pin can be programmed to operate at either 1.8 V or 3.3 V by setting the bit corresponding to that pin in one of the HW_PINCTRL_DRIVEx registers. The EMI pins are capable of running at 1.5/1.8 V depending on the EMI_VDDIO voltage.

Note

The GPIO *pad driver has two PMOS pullup drivers directly connected to the 1.8 or 3.3-V power supply*.

9.2.2.3 Pullup Selection

Several digital pins can be programmed to enable pullups by setting the appropriate bit in one of the HW_PINCTRL_PULLx registers. Note that enabling the pullup will also disable the internal gate keeper on that pin. The EMI pads do not have pullup selection, but do have keeper disable capability.

The pullups are tied to the physical pin pad and not to the function. So, for example, if the AUART1_TX pullup is enabled, that pullup will be present on the AUART1_TX pin regardless of what function (AUART1_TX, IR_TX, or SSP1_DATA7) is pin multiplexed out of that pin.

The table below lists which function (not which pin name), an internal pullup has been implemented, to assist in the hardware interfaces' operation.

Table 9-3. i.MX28 Functions with Pullup Resistors

FUNCTION	Type	Value	Present on 289BGA	Present on 204-pins
SSP0_DATA0	Pullup	47K	Y	Y
SSP0_DATA1	Pullup	47K	Y	Y
SSP0_DATA2	Pullup	47K	Y	Y
SSP0_DATA3	Pullup	47K	Y	Y
SSP0_DATA4	Pullup	47K	Y	Y
SSP0_DATA5	Pullup	47K	Y	Y
SSP0_DATA6	Pullup	47K	Y	Y
SSP0_DATA7	Pullup	47K	Y	Y
SSP0_CMD	Pullup	10K	Y	Y
SSP0_DETECT	Pullup	10K	Y	Y
SSP1_DATA0	Pullup	47K	Y	Y
SSP1_DATA1	Pullup	47K	Y	Y
SSP1_DATA2	Pullup	47K	Y	Y
SSP1_DATA3	Pullup	47K	Y	Y
SSP1_DATA4	Pullup	47K	Y	Y
SSP1_DATA5	Pullup	47K	Y	Y
SSP1_DATA6	Pullup	47K	Y	Y
SSP1_DATA7	Pullup	47K	Y	Y
SSP1_CMD	Pullup	10K	Y	Y
SSP1_DETECT	Pullup	10K	Y	Y
SSP2_DATA0	Pullup	47K	Y	Y

FUNCTION	Type	Value	Present on 289BGA	Present on 204-pins
SSP2_DATA1	Pullup	47K	Y	Y
SSP2_DATA2	Pullup	47K	Y	Y
SSP2_DATA3	Pullup	47K	Y	Y
SSP2_DATA4	Pullup	47K	Y	Y
SSP2_DATA5	Pullup	47K	Y	Y
SSP2_DATA6	Pullup	47K	Y	N
SSP2_DATA7	Pullup	47K	Y	N
SSP2_CMD	Pullup	10K	Y	Y
SSP2_DETECT	Pullup	10K	Y	Y
SSP3_DATA0	Pullup	47K	Y	Y
SSP3_DATA1	Pullup	47K	Y	Y
SSP3_DATA2	Pullup	47K	Y	Y
SSP3_DATA3	Pullup	47K	Y	Y
SSP3_DATA4	Pullup	47K	Y	Y
SSP3_DATA5	Pullup	47K	Y	Y
SSP3_DATA6	Pullup	47K	Y	N
SSP3_DATA7	Pullup	47K	Y	N
SSP3_CMD	Pullup	10K	Y	Y
SSP3_DETECT	Pullup	10K	Y	Y
GPMI_CE0N	Pullup	47K	Y	Y
GPMI_CE1N	Pullup	47K	Y	Y
GPMI_CE2N	Pullup	47K	Y	N
GPMI_CE3N	Pullup	47K	Y	N
GPMI_RDY0	Pullup	10K	Y	Y
GPMI_RDY1	Pullup	10K	Y	Y
GPMI_RDY2	Pullup	10K	Y	N
GPMI_RDY3	Pullup	10K	Y	N

9.2.3 GPIO Interface

The registers discussed in the following sections exist within each of the three GPIO banks to configure the chip's digital pins. Some pins exist in the 289-pin package only. The registers that control those pins exist but do not perform any useful function when in a 204-pin package.

9.2.3.1 Output Operation

Programming and controlling a digital pin as a GPIO output is accomplished by programming the appropriate bits in four registers, as shown in the figure below.

- After setting the field in the HW_PINCTRL_MUXSELx to program for GPIO control, the HW_PINCTRL_DRIVEx register bit is set for the desired drive strength and pin voltage. Set bits in HW_PINCTRL_PULLx as required to enable pullups.
- The HW_PINCTRL_DOUTx register bit is then loaded with the level that will initially be driven on the pin.
- Finally, the HW_PINCTRL_DOEx register bit is set.
- Once set, the logic value the HW_PINCTRL_DOUTx bit will be driven on the pin and the value can be toggled with repeated writes.

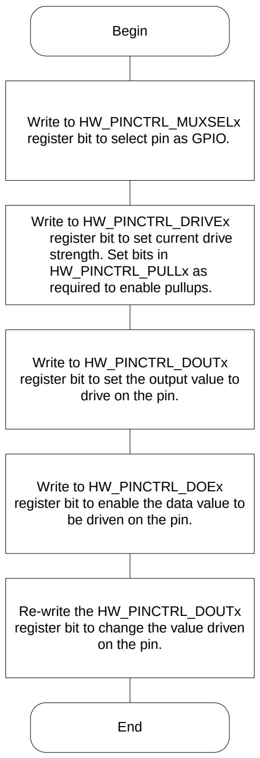

Begin
Write to HW_PINCTRL_MUXSELx register bit to select pin as GPIO.
Write to HW_PINCTRL_DRIVEx register bit to set current drive strength. Set bits in HW_PINCTRL_PULLx as required to enable pullups.
Write to HW_PINCTRL_DOUTx register bit to set the output value to drive on the pin.
Write to HW_PINCTRL_DOEx register bit to enable the data value to be driven on the pin.
Re-write the HW_PINCTRL_DOUTx register bit to change the value driven on the pin.
End

Figure 9-2. GPIO Output Setup Flowchart

9.2.3.2 Input Operation

Digital pins in Banks **0~4** may be used as a GPIO input by programming its HW_PINCTRL_MUXSELx field to 3 to enable GPIO mode, programming its HW_PINCTRL_DOEx field to 0 to disable output, and then reading from the HW_PINCTRL_DINx register, as shown in the figure below. Note that because of clock synchronization issues, the logic levels read from the HW_PINCTRL_DINx registers are delayed from the pins by several APBX clock cycles.

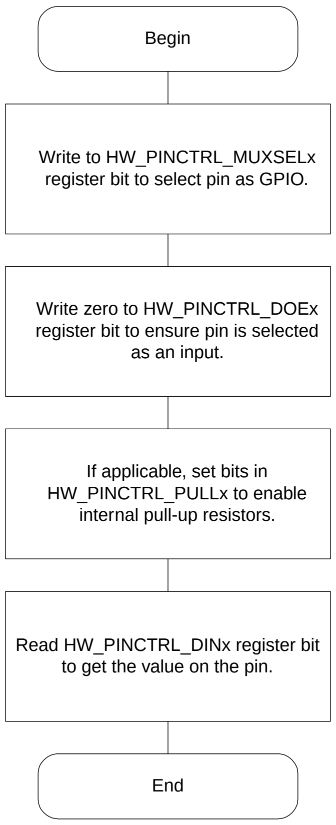

Begin
Write to HW_PINCTRL_MUXSELx register bit to select pin as GPIO.
Write zero to HW_PINCTRL_DOEx register bit to ensure pin is selected as an input.
If applicable, set bits in HW_PINCTRL_PULLx to enable internal pull-up resistors.
Read HW_PINCTRL_DINx register bit to get the value on the pin.
End

Figure 9-3. GPIO Input Setup Flowchart

9.2.3.3 Input Interrupt Operation

Programming and controlling a digital pin as a GPIO interrupt input is accomplished by programming the appropriate bits in six registers, as shown in the figure below.

- After setting the HW_PINCTRL_MUXSELx register for GPIO, the HW_PINCTRL_IRQLEVELx and HW_PINCTRL_IRQPOLx registers set the interrupt trigger mode. A GPIO interrupt pin can be programmed in one of four trigger detect modes: positive edge, negative edge, positive level, and negative level triggered.
- The HW_PINCTRL_IRQSTATx register bit should then be cleared to ensure that there are no interrupts pending when enabled.
- Setting the HW_PINCTRL_PIN2IRQx register bit will then set up the pin to be an interrupt pin.
- At this point, if an interrupt event occurs on the pin, it will be sensed and recorded in the appropriate HW_PINCTRL_IRQSTATx bit.
- However, the interrupt will not be communicated back to the interrupt collector until the HW_PINCTRL_IRQENx register bit is enabled.

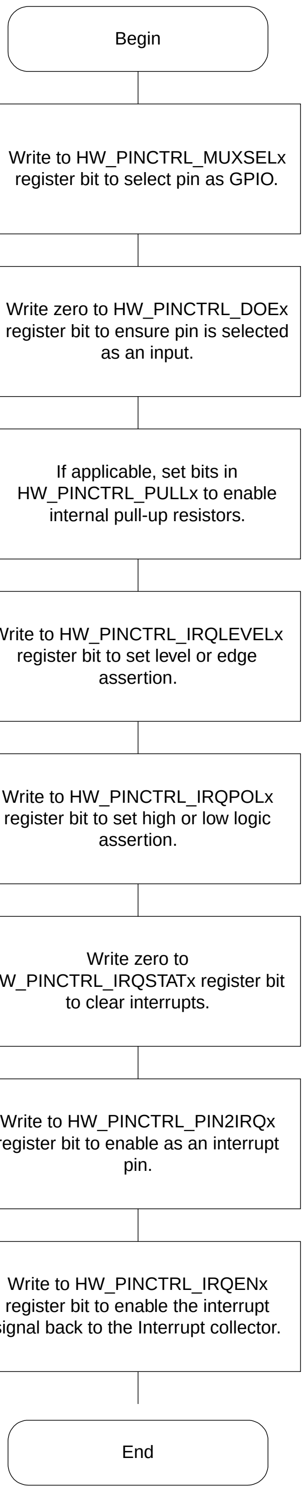

Begin
Write to HW_PINCTRL_MUXSELx register bit to select pin as GPIO.
Write zero to HW_PINCTRL_DOEx register bit to ensure pin is selected as an input.
If applicable, set bits in HW_PINCTRL_PULLx to enable internal pull-up resistors.
Write to HW_PINCTRL_IRQLEVELx register bit to set level or edge assertion.
Write to HW_PINCTRL_IRQPOLx register bit to set high or low logic assertion.
Write zero to HW_PINCTRL_IRQSTATx register bit to clear interrupts.
Write to HW_PINCTRL_PIN2IRQx register bit to enable as an interrupt pin.
Write to HW_PINCTRL_IRQENx register bit to enable the interrupt signal back to the Interrupt collector.
End

Figure 9-4. GPIO Interrupt Flowchart

The figure below shows the logic diagram for the interrupt-generation circuit.

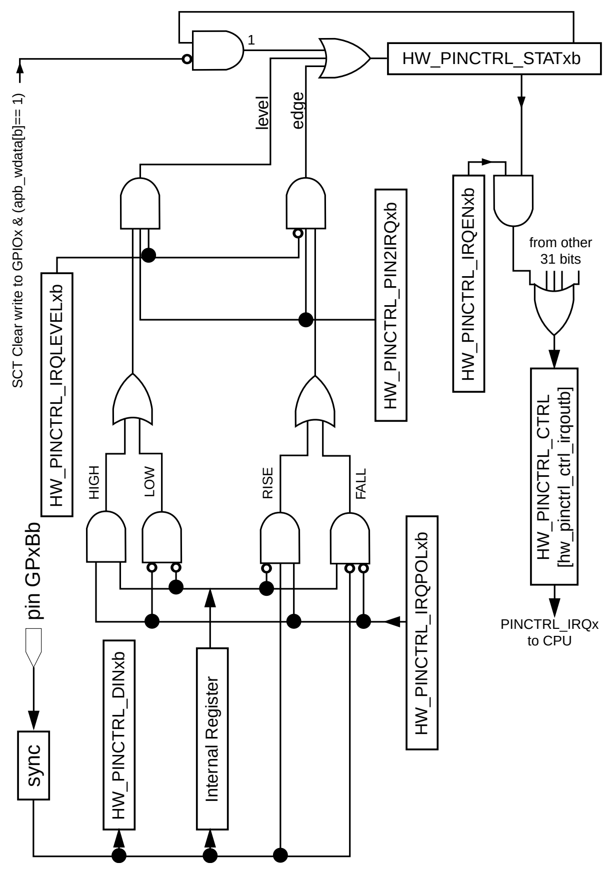


Figure 9-5. GPIO Interrupt Generation

9.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

ENET pins will not be affected by SFTRST, it only resets when power is on.

9.4 Programmable Registers

PINCTRL Hardware Register Format Summary

HW_PINCTRL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_8000	PINCTRL Block Control Register (HW_PINCTRL_CTRL)	32	R/W	C1F0_0000h	9.4.1/692
8001_8100	PINCTRL Pin Mux Select Register 0 (HW_PINCTRL_MUXSEL0)	32	R/W	0000_FFFFh	9.4.2/694

HW_PINCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_8110	PINCTRL Pin Mux Select Register 1 (HW_PINCTRL_MUXSEL1)	32	R/W	03FF_FFFFh	9.4.3/696
8001_8120	PINCTRL Pin Mux Select Register 2 (HW_PINCTRL_MUXSEL2)	32	R/W	FFFF_FFFFh	9.4.4/699
8001_8130	PINCTRL Pin Mux Select Register 3 (HW_PINCTRL_MUXSEL3)	32	R/W	FFFF_FFFFh	9.4.5/702
8001_8140	PINCTRL Pin Mux Select Register 4 (HW_PINCTRL_MUXSEL4)	32	R/W	FF3F_FFFFh	9.4.6/705
8001_8150	PINCTRL Pin Mux Select Register 5 (HW_PINCTRL_MUXSEL5)	32	R/W	00FF_0FFFh	9.4.7/708
8001_8160	PINCTRL Pin Mux Select Register 6 (HW_PINCTRL_MUXSEL6)	32	R/W	FFFF_FFFFh	9.4.8/710
8001_8170	PINCTRL Pin Mux Select Register 7 (HW_PINCTRL_MUXSEL7)	32	R/W	3FFF_FF3Fh	9.4.9/713
8001_8180	PINCTRL Pin Mux Select Register 8 (HW_PINCTRL_MUXSEL8)	32	R/W	FFFF_FFFFh	9.4.10/716
8001_8190	PINCTRL Pin Mux Select Register 9 (HW_PINCTRL_MUXSEL9)	32	R/W	0000_0003h	9.4.11/719
8001_81A0	PINCTRL Pin Mux Select Register 10 (HW_PINCTRL_MUXSEL10)	32	R/W	FFFF_FFFFh	9.4.12/720
8001_81B0	PINCTRL Pin Mux Select Register 11 (HW_PINCTRL_MUXSEL11)	32	R/W	0030_FFFFh	9.4.13/723
8001_81C0	PINCTRL Pin Mux Select Register 12 (HW_PINCTRL_MUXSEL12)	32	R/W	3FFF_FFFFh	9.4.14/725
8001_81D0	PINCTRL Pin Mux Select Register 13 (HW_PINCTRL_MUXSEL13)	32	R/W	0003_FFFFh	9.4.15/728
8001_8300	PINCTRL Drive Strength and Voltage Register 0 (HW_PINCTRL_DRIVE0)	32	R/W	4444_4444h	9.4.16/730
8001_8310	PINCTRL Drive Strength and Voltage Register 1 (HW_PINCTRL_DRIVE1)	32	R/W	0000_0000h	9.4.17/734
8001_8320	PINCTRL Drive Strength and Voltage Register 2 (HW_PINCTRL_DRIVE2)	32	R/W	4444_4444h	9.4.18/734
8001_8330	PINCTRL Drive Strength and Voltage Register 3 (HW_PINCTRL_DRIVE3)	32	R/W	0004_4444h	9.4.19/737
8001_8340	PINCTRL Drive Strength and Voltage Register 4 (HW_PINCTRL_DRIVE4)	32	R/W	4444_4444h	9.4.20/740
8001_8350	PINCTRL Drive Strength and Voltage Register 5 (HW_PINCTRL_DRIVE5)	32	R/W	4444_4444h	9.4.21/743
8001_8360	PINCTRL Drive Strength and Voltage Register 6 (HW_PINCTRL_DRIVE6)	32	R/W	4444_4444h	9.4.22/746
8001_8370	PINCTRL Drive Strength and Voltage Register 7 (HW_PINCTRL_DRIVE7)	32	R/W	4444_4444h	9.4.23/749

HW_PINCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_8380	PINCTRL Drive Strength and Voltage Register 8 (HW_PINCTRL_DRIVE8)	32	R/W	4444_4444h	9.4.24/753
8001_8390	PINCTRL Drive Strength and Voltage Register 9 (HW_PINCTRL_DRIVE9)	32	R/W	4444_0444h	9.4.25/756
8001_83A0	PINCTRL Drive Strength and Voltage Register 10 (HW_PINCTRL_DRIVE10)	32	R/W	0044_4444h	9.4.26/759
8001_83B0	PINCTRL Drive Strength and Voltage Register 11 (HW_PINCTRL_DRIVE11)	32	R/W	0000_4444h	9.4.27/762
8001_83C0	PINCTRL Drive Strength and Voltage Register 12 (HW_PINCTRL_DRIVE12)	32	R/W	4444_4444h	9.4.28/764
8001_83D0	PINCTRL Drive Strength and Voltage Register 13 (HW_PINCTRL_DRIVE13)	32	R/W	4444_4444h	9.4.29/767
8001_83E0	PINCTRL Drive Strength and Voltage Register 14 (HW_PINCTRL_DRIVE14)	32	R/W	4444_0444h	9.4.30/770
8001_83F0	PINCTRL Drive Strength and Voltage Register 15 (HW_PINCTRL_DRIVE15)	32	R/W	0444_4444h	9.4.31/773
8001_8400	PINCTRL Drive Strength and Voltage Register 16 (HW_PINCTRL_DRIVE16)	32	R/W	4444_4444h	9.4.32/776
8001_8410	PINCTRL Drive Strength and Voltage Register 17 (HW_PINCTRL_DRIVE17)	32	R/W	4444_4444h	9.4.33/779
8001_8420	PINCTRL Drive Strength and Voltage Register 18 (HW_PINCTRL_DRIVE18)	32	R/W	0004_0004h	9.4.34/783
8001_8430	PINCTRL Drive Strength and Voltage Register 19 (HW_PINCTRL_DRIVE19)	32	R	0000_0000h	9.4.35/784
8001_8600	PINCTRL Bank 0 Pull Up Resistor Enable Register (HW_PINCTRL_PULL0)	32	R/W	0000_0000h	9.4.36/785
8001_8610	PINCTRL Bank 1 Pull Up Resistor Enable Register (HW_PINCTRL_PULL1)	32	R/W	0000_0000h	9.4.37/787
8001_8620	PINCTRL Bank 2 Pull Up Resistor Enable Register (HW_PINCTRL_PULL2)	32	R/W	0000_0000h	9.4.38/789
8001_8630	PINCTRL Bank 3 Pull Up Resistor Enable Register (HW_PINCTRL_PULL3)	32	R/W	0000_0000h	9.4.39/791
8001_8640	PINCTRL Bank 4 Pull Up Resistor Enable Register (HW_PINCTRL_PULL4)	32	R/W	0000_0000h	9.4.40/794
8001_8650	PINCTRL Bank 5 Pad Keeper Disable Register (HW_PINCTRL_PULL5)	32	R/W	0000_0000h	9.4.41/796
8001_8660	PINCTRL Bank 6 Pad Keeper Disable Register (HW_PINCTRL_PULL6)	32	R/W	0000_0000h	9.4.42/798
8001_8700	PINCTRL Bank 0 Data Output Register (HW_PINCTRL_DOUT0)	32	R/W	0000_0000h	9.4.43/800
8001_8710	PINCTRL Bank 1 Data Output Register (HW_PINCTRL_DOUT1)	32	R/W	0000_0000h	9.4.44/801

HW_PINCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_8720	PINCTRL Bank 2 Data Output Register (HW_PINCTRL_DOUT2)	32	R/W	0000_0000h	9.4.45/801
8001_8730	PINCTRL Bank 3 Data Output Register (HW_PINCTRL_DOUT3)	32	R/W	0000_0000h	9.4.46/802
8001_8740	PINCTRL Bank 4 Data Output Register (HW_PINCTRL_DOUT4)	32	R/W	0000_0000h	9.4.47/803
8001_8900	PINCTRL Bank 0 Data Input Register (HW_PINCTRL_DIN0)	32	R	0000_0000h	9.4.48/804
8001_8910	PINCTRL Bank 1 Data Input Register (HW_PINCTRL_DIN1)	32	R	0000_0000h	9.4.49/805
8001_8920	PINCTRL Bank 2 Data Input Register (HW_PINCTRL_DIN2)	32	R	0000_0000h	9.4.50/806
8001_8930	PINCTRL Bank 3 Data Input Register (HW_PINCTRL_DIN3)	32	R	0000_0000h	9.4.51/806
8001_8940	PINCTRL Bank 4 Data Input Register (HW_PINCTRL_DIN4)	32	R	0000_0000h	9.4.52/807
8001_8B00	PINCTRL Bank 0 Data Output Enable Register (HW_PINCTRL_DOE0)	32	R/W	0000_0000h	9.4.53/808
8001_8B10	PINCTRL Bank 1 Data Output Enable Register (HW_PINCTRL_DOE1)	32	R/W	0000_0000h	9.4.54/809
8001_8B20	PINCTRL Bank 2 Data Output Enable Register (HW_PINCTRL_DOE2)	32	R/W	0000_0000h	9.4.55/810
8001_8B30	PINCTRL Bank 3 Data Output Enable Register (HW_PINCTRL_DOE3)	32	R/W	0000_0000h	9.4.56/810
8001_8B40	PINCTRL Bank 4 Data Output Enable Register (HW_PINCTRL_DOE4)	32	R/W	0000_0000h	9.4.57/811
8001_9000	PINCTRL Bank 0 Interrupt Select Register (HW_PINCTRL_PIN2IRQ0)	32	R/W	0000_0000h	9.4.58/812
8001_9010	PINCTRL Bank 1 Interrupt Select Register (HW_PINCTRL_PIN2IRQ1)	32	R/W	0000_0000h	9.4.59/813
8001_9020	PINCTRL Bank 2 Interrupt Select Register (HW_PINCTRL_PIN2IRQ2)	32	R/W	0000_0000h	9.4.60/814
8001_9030	PINCTRL Bank 3 Interrupt Select Register (HW_PINCTRL_PIN2IRQ3)	32	R/W	0000_0000h	9.4.61/815
8001_9040	PINCTRL Bank 4 Interrupt Select Register (HW_PINCTRL_PIN2IRQ4)	32	R/W	0000_0000h	9.4.62/816
8001_9100	PINCTRL Bank 0 Interrupt Mask Register (HW_PINCTRL_IRQEN0)	32	R/W	0000_0000h	9.4.63/817
8001_9110	PINCTRL Bank 1 Interrupt Mask Register (HW_PINCTRL_IRQEN1)	32	R/W	0000_0000h	9.4.64/818
8001_9120	PINCTRL Bank 2 Interrupt Mask Register (HW_PINCTRL_IRQEN2)	32	R/W	0000_0000h	9.4.65/819
8001_9130	PINCTRL Bank 3 Interrupt Mask Register (HW_PINCTRL_IRQEN3)	32	R/W	0000_0000h	9.4.66/820

HW_PINCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_9140	PINCTRL Bank 4 Interrupt Mask Register (HW_PINCTRL_IRQEN4)	32	R/W	0000_0000h	9.4.67/821
8001_9200	PINCTRL Bank 0 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL0)	32	R/W	0000_0000h	9.4.68/822
8001_9210	PINCTRL Bank 1 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL1)	32	R/W	0000_0000h	9.4.69/823
8001_9220	PINCTRL Bank 2 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL2)	32	R/W	0000_0000h	9.4.70/823
8001_9230	PINCTRL Bank 3 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL3)	32	R/W	0000_0000h	9.4.71/824
8001_9240	PINCTRL Bank 4 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL4)	32	R/W	0000_0000h	9.4.72/825
8001_9300	PINCTRL Bank 0 Interrupt Polarity Register (HW_PINCTRL_IRQPOL0)	32	R/W	0000_0000h	9.4.73/826
8001_9310	PINCTRL Bank 1 Interrupt Polarity Register (HW_PINCTRL_IRQPOL1)	32	R/W	0000_0000h	9.4.74/827
8001_9320	PINCTRL Bank 2 Interrupt Polarity Register (HW_PINCTRL_IRQPOL2)	32	R/W	0000_0000h	9.4.75/828
8001_9330	PINCTRL Bank 3 Interrupt Polarity Register (HW_PINCTRL_IRQPOL3)	32	R/W	0000_0000h	9.4.76/829
8001_9340	PINCTRL Bank 4 Interrupt Polarity Register (HW_PINCTRL_IRQPOL4)	32	R/W	0000_0000h	9.4.77/830
8001_9400	PINCTRL Bank 0 Interrupt Status Register (HW_PINCTRL_IRQSTAT0)	32	R/W	0000_0000h	9.4.78/831
8001_9410	PINCTRL Bank 1 Interrupt Status Register (HW_PINCTRL_IRQSTAT1)	32	R/W	0000_0000h	9.4.79/832
8001_9420	PINCTRL Bank 2 Interrupt Status Register (HW_PINCTRL_IRQSTAT2)	32	R/W	0000_0000h	9.4.80/833
8001_9430	PINCTRL Bank 3 Interrupt Status Register (HW_PINCTRL_IRQSTAT3)	32	R/W	0000_0000h	9.4.81/834
8001_9440	PINCTRL Bank 4 Interrupt Status Register (HW_PINCTRL_IRQSTAT4)	32	R/W	0000_0000h	9.4.82/835
8001_9A40	PINCTRL EMI Slice ODT Control (HW_PINCTRL_EMI_ODT_CTRL)	32	R/W	0888_8888h	9.4.83/836
8001_9B80	PINCTRL EMI Slice DS Control (HW_PINCTRL_EMI_DS_CTRL)	32	R/W	0003_0000h	9.4.84/840

9.4.1 PINCTRL Block Control Register (HW_PINCTRL_CTRL)

The PINCTRL Block Control Register contains the block control bits and combined interrupt output status for each PINCTRL bank.

HW_PINCTRL_CTRL: 0x000

HW_PINCTRL_CTRL_SET: 0x004

HW_PINCTRL_CTRL_CLR: 0x008

HW_PINCTRL_CTRL_TOG: 0x00C

This register contains block-wide control bits and combined bank interrupt status bits. For normal operation, write a 0x0 into this register.

Address: HW_PINCTRL_CTRL – 8001_8000h base + 0h offset = 8001_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSRVD2					PRESENT4	PRESENT3	PRESENT2	PRESENT1	PRESENT0	RSRVD1[19:16]			
W																
Reset	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:5]											IRQOUT4	IRQOUT3	IRQOUT2	IRQOUT1	IRQOUT0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_CTRL field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable operation of any of the PINCTRL banks. When set to one, it forces a block-level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one, it disables the block clock.
29–25 RSRVD2	Always write zeroes to this field.
24 PRESENT4	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 4 is not present in this product. 1: GPIO functionality for Bank 4 is present.
23 PRESENT3	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 3 is not present in this product. 1: GPIO functionality for Bank 3 is present.
22 PRESENT2	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 2 is not present in this product. 1: GPIO functionality for Bank 2 is present.

HW_PINCTRL_CTRL field descriptions (continued)

Field	Description
21 PRESENT1	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 1 is not present in this product. 1: GPIO functionality for Bank 1 is present.
20 PRESENT0	GPIO Functionality Present. 0: GPIO functionality for Pin Control Bank 0 is not present in this product. 1: GPIO functionality for Bank 0 is present.
19–5 RSRVD1	Always write zeroes to this field.
4 IRQOUT4	Read-only view of the interrupt collector GPIO4 signal, sourced from the combined IRQ outputs from bank 4.
3 IRQOUT3	Read-only view of the interrupt collector GPIO3 signal, sourced from the combined IRQ outputs from bank 3.
2 IRQOUT2	Read-only view of the interrupt collector GPIO2 signal, sourced from the combined IRQ outputs from bank 2.
1 IRQOUT1	Read-only view of the interrupt collector GPIO1 signal, sourced from the combined IRQ outputs from bank 1.
0 IRQOUT0	Read-only view of the interrupt collector GPIO0 signal, sourced from the combined IRQ outputs from bank 0.

9.4.2 PINCTRL Pin Mux Select Register 0 (HW_PINCTRL_MUXSEL0)

The PINCTRL Pin Mux Select Register provides pin function selection for 8 pins in bank0.

HW_PINCTRL_MUXSEL0: 0x100

HW_PINCTRL_MUXSEL0_SET: 0x104

HW_PINCTRL_MUXSEL0_CLR: 0x108

HW_PINCTRL_MUXSEL0_TOG: 0x10C

This register allows the programmer to select which hardware interface blocks drive the 8 pins shown above.

Address: HW_PINCTRL_MUXSEL0 – 8001_8000h base + 100h offset = 8001_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK0_ PIN07		BANK0_ PIN06		BANK0_ PIN05		BANK0_ PIN04		BANK0_ PIN03		BANK0_ PIN02		BANK0_ PIN01		BANK0_ PIN00	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL0 field descriptions

Field	Description
31–16 RSRVD0	Always write zeroes to this field.

15–14
BANK0_PIN07
	Pin 124, GPMI_D07 pin function selection:
	00= gpmi_d7;
	01= ssp1_d7;
	10= reserved;
	11= GPIO.

13–12
BANK0_PIN06
	Pin 130, GPMI_D06 pin function selection:
	00= gpmi_d6;
	01= ssp1_d6;
	10= reserved;
	11= GPIO.

11–10
BANK0_PIN05
	Pin 116, GPMI_D05 pin function selection:
	00= gpmi_d5;
	01= ssp1_d5;
	10= reserved;
	11= GPIO.

9–8
BANK0_PIN04
	Pin 128, GPMI_D04 pin function selection:
	00= gpmi_d4;
	01= ssp1_d4;
	10= reserved;
	11= GPIO.

7–6
BANK0_PIN03
	Pin 132, GPMI_D03 pin function selection:
	00= gpmi_d3;
	01= ssp1_d3;
	10= reserved;
	11= GPIO.

5–4
BANK0_PIN02
	Pin 126, GPMI_D02 pin function selection:
	00= gpmi_d2;
	01= ssp1_d2;

HW_PINCTRL_MUXSEL0 field descriptions (continued)

Field	Description
	10= reserved; 11= GPIO.

3–2
BANK0_PIN01
	Pin 136, GPMI_D01 pin function selection:
	00= gpmi_d1;
	01= ssp1_d1;
	10= reserved;
	11= GPIO.

1–0
BANK0_PIN00
	Pin 134, GPMI_D00 pin function selection:
	00= gpmi_d0;
	01= ssp1_d0;
	10= reserved;
	11= GPIO.

9.4.3 PINCTRL Pin Mux Select Register 1 (HW_PINCTRL_MUXSEL1)

The PINCTRL Pin Mux Select Register provides pin function selection for 13 pins in bank0.

HW_PINCTRL_MUXSEL1: 0x110

HW_PINCTRL_MUXSEL1_SET: 0x114

HW_PINCTRL_MUXSEL1_CLR: 0x118

HW_PINCTRL_MUXSEL1_TOG: 0x11C

This register allows the programmer to select which hardware interface blocks drive the 13 pins shown above.

Address: HW_PINCTRL_MUXSEL1 – 8001_8000h base + 110h offset = 8001_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0						BANK0_ PIN28		BANK0_ PIN27		BANK0_ PIN26		BANK0_ PIN25		BANK0_ PIN24	
W	▨															
Reset	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK0_PIN23		BANK0_PIN22		BANK0_PIN21		BANK0_PIN20		BANK0_PIN19		BANK0_PIN18		BANK0_PIN17		BANK0_PIN16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL1 field descriptions

Field	Description
31–26 RSRVD0	Always write zeroes to this field.

25–24
BANK0_PIN28
	Pin 129, GPMI_RESETN pin function selection:
	00= gpmi_resetn;
	01= ssp3_cmd;
	10= reserved;
	11= GPIO.

23–22
BANK0_PIN27
	Pin 123, GPMI_CLE pin function selection:
	00= gpmi_cle;
	01= ssp3_d2;
	10= ssp3_d5;
	11= GPIO.

21–20
BANK0_PIN26
	Pin 117, GPMI_ALE pin function selection:
	00= gpmi_ale;
	01= ssp3_d1;
	10= ssp3_d4;
	11= GPIO.

19–18
BANK0_PIN25
	Pin 127, GPMI_WRN pin function selection:
	00= gpmi_wrn;
	01= ssp1_sck;
	10= reserved;
	11= GPIO.

17–16
BANK0_PIN24
	Pin 110, GPMI_RDN pin function selection:
	00= gpmi_rdn;
	01= ssp3_sck;
	10= reserved;
	11= GPIO.

15–14
BANK0_PIN23
	Pin 80, GPMI_RDY3 pin function selection:
	00= gpmi_ready3;
	01= can0_rx;

HW_PINCTRL_MUXSEL1 field descriptions (continued)

Field	Description
	10= hsadc_trigger; 11= GPIO.

13–12
BANK0_PIN22
	Pin 88, GPMI_RDY2 pin function selection:
	00= gpmi_ready2;
	01= can0_tx;
	10= enet0_tx_er;
	11= GPIO.

11–10
BANK0_PIN21
	Pin 119, GPMI_RDY1 pin function selection:
	00= gpmi_ready1;
	01= ssp1_cmd;
	10= reserved;
	11= GPIO.

9–8
BANK0_PIN20
	Pin 103, GPMI_RDY0 pin function selection:
	00= gpmi_ready0;
	01= ssp1_card_detect;
	10= usb0_id;
	11= GPIO.

7–6
BANK0_PIN19
	Pin 135, GPMI_CE3N pin function selection:
	00= gpmi_ce3n;
	01= can1_rx;
	10= saif1_mclk;
	11= GPIO.

5–4
BANK0_PIN18
	Pin 92, GPMI_CE2N pin function selection:
	00= gpmi_ce2n;
	01= can1_tx;
	10= enet0_rx_er;
	11= GPIO.

3–2
BANK0_PIN17
	Pin 131, GPMI_CE1N pin function selection:
	00= gpmi_ce1n;
	01= ssp3_d3;
	10= reserved;
	11= GPIO.

1–0
BANK0_PIN16
	Pin 115, GPMI_CE0N pin function selection:
	00= gpmi_ce0n;
	01= ssp3_d0;

HW_PINCTRL_MUXSEL1 field descriptions (continued)

Field	Description
	10= reserved; 11= GPIO.

9.4.4 PINCTRL Pin Mux Select Register 2 (HW_PINCTRL_MUXSEL2)

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank1.

HW_PINCTRL_MUXSEL2: 0x120

HW_PINCTRL_MUXSEL2_SET: 0x124

HW_PINCTRL_MUXSEL2_CLR: 0x128

HW_PINCTRL_MUXSEL2_TOG: 0x12C

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

Address: HW_PINCTRL_MUXSEL2 – 8001_8000h base + 120h offset = 8001_8120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK1_PIN15		BANK1_PIN14		BANK1_PIN13		BANK1_PIN12		BANK1_PIN11		BANK1_PIN10		BANK1_PIN09		BANK1_PIN08	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK1_PIN07		BANK1_PIN06		BANK1_PIN05		BANK1_PIN04		BANK1_PIN03		BANK1_PIN02		BANK1_PIN01		BANK1_PIN00	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL2 field descriptions

Field	Description

31–30
BANK1_PIN15
	Pin 114, LCD_D15 pin function selection:
	00= lcd_d15;
	01= reserved;
	10= etm_da15;
	11= GPIO.

29–28 BANK1_PIN14	Pin 106, LCD_D14 pin function selection:

HW_PINCTRL_MUXSEL2 field descriptions (continued)

Field	Description

	00= lcd_d14;
	01= reserved;
	10= etm_da14;
	11= GPIO.

27–26
BANK1_PIN13
	Pin 102, LCD_D13 pin function selection:
	00= lcd_d13;
	01= reserved;
	10= etm_da13;
	11= GPIO.

25–24
BANK1_PIN12
	Pin 87, LCD_D12 pin function selection:
	00= lcd_d12;
	01= reserved;
	10= etm_da12;
	11= GPIO.

23–22
BANK1_PIN11
	Pin 95, LCD_D11 pin function selection:
	00= lcd_d11;
	01= reserved;
	10= etm_da11;
	11= GPIO.

21–20
BANK1_PIN10
	Pin 85, LCD_D10 pin function selection:
	00= lcd_d10;
	01= reserved;
	10= etm_da10;
	11= GPIO.

19–18
BANK1_PIN09
	Pin 99, LCD_D09 pin function selection:
	00= lcd_d9;
	01= etm_da4;
	10= etm_da9;
	11= GPIO.

17–16
BANK1_PIN08
	Pin 93, LCD_D08 pin function selection:
	00= lcd_d8;
	01= etm_da3;
	10= etm_da8;
	11= GPIO.

HW_PINCTRL_MUXSEL2 field descriptions (continued)

Field	Description

15–14
BANK1_PIN07
	Pin 75, LCD_D07 pin function selection:
	00= lcd_d7;
	01= reserved;
	10= etm_da7;
	11= GPIO.

13–12
BANK1_PIN06
	Pin 91, LCD_D06 pin function selection:
	00= lcd_d6;
	01= reserved;
	10= etm_da6;
	11= GPIO.

11–10
BANK1_PIN05
	Pin 89, LCD_D05 pin function selection:
	00= lcd_d5;
	01= reserved;
	10= etm_da5;
	11= GPIO.

9–8
BANK1_PIN04
	Pin 79, LCD_D04 pin function selection:
	00= lcd_d4;
	01= etm_da9;
	10= etm_da4;
	11= GPIO.

7–6
BANK1_PIN03
	Pin 77, LCD_D03 pin function selection:
	00= lcd_d3;
	01= etm_da8;
	10= etm_da3;
	11= GPIO.

5–4
BANK1_PIN02
	Pin 67, LCD_D02 pin function selection:
	00= lcd_d2;
	01= reserved;
	10= etm_da2;
	11= GPIO.

3–2
BANK1_PIN01
	Pin 69, LCD_D01 pin function selection:
	00= lcd_d1;
	01= reserved;
	10= etm_da1;
	11= GPIO.

HW_PINCTRL_MUXSEL2 field descriptions (continued)

Field	Description

1–0
BANK1_PIN00
	Pin 63, LCD_D00 pin function selection:
	00= lcd_d0;
	01= reserved;
	10= etm_da0;
	11= GPIO.

9.4.5 PINCTRL Pin Mux Select Register 3 (HW_PINCTRL_MUXSEL3)

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank1.

HW_PINCTRL_MUXSEL3: 0x130

HW_PINCTRL_MUXSEL3_SET: 0x134

HW_PINCTRL_MUXSEL3_CLR: 0x138

HW_PINCTRL_MUXSEL3_TOG: 0x13C

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

Address: HW_PINCTRL_MUXSEL3 – 8001_8000h base + 130h offset = 8001_8130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK1_PIN31		BANK1_PIN30		BANK1_PIN29		BANK1_PIN28		BANK1_PIN27		BANK1_PIN26		BANK1_PIN25		BANK1_PIN24	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK1_PIN23		BANK1_PIN22		BANK1_PIN21		BANK1_PIN20		BANK1_PIN19		BANK1_PIN18		BANK1_PIN17		BANK1_PIN16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL3 field descriptions

Field	Description

31–30
BANK1_PIN31
	Pin 111, LCD_ENABLE pin function selection:
	00= lcd_enable;
	01= reserved;
	10= reserved;

HW_PINCTRL_MUXSEL3 field descriptions (continued)

Field	Description
	11= GPIO.

29–28
BANK1_PIN30
	Pin 73, LCD_DOTCLK pin function selection:
	00= lcd_dotclk;
	01= saif1_mclk;
	10= etm_tclk;
	11= GPIO.

27–26
BANK1_PIN29
	Pin 71, LCD_HSYNC pin function selection:
	00= lcd_hsync;
	01= saif1_sdata1;
	10= etm_tctl;
	11= GPIO.

25–24
BANK1_PIN28
	Pin 59, LCD_VSYNC pin function selection:
	00= lcd_vsync;
	01= saif1_sdata0;
	10= reserved;
	11= GPIO.

23–22
BANK1_PIN27
	Pin 113, LCD_CS pin function selection:
	00= lcd_cs;
	01= lcd_enable;
	10= reserved;
	11= GPIO.

21–20
BANK1_PIN26
	Pin 94, LCD_RS pin function selection:
	00= lcd_rs;
	01= lcd_dotclk;
	10= reserved;
	11= GPIO.

19–18
BANK1_PIN25
	Pin 55, LCD_WR_RWN pin function selection:
	00= lcd_wr_rwn;
	01= lcd_hsync;
	10= etm_tclk;
	11= GPIO.

17–16
BANK1_PIN24
	Pin 96, LCD_RD_E pin function selection:
	00= lcd_rd_e;
	01= lcd_vsync;
	10= etm_tctl;

HW_PINCTRL_MUXSEL3 field descriptions (continued)

Field	Description
	11= GPIO.

15–14 BANK1_PIN23	Pin 108, LCD_D23 pin function selection:
	00= lcd_d23;
	01= enet1_1588_event3_in;
	10= etm_da0;
	11= GPIO.

13–12 BANK1_PIN22	Pin 120, LCD_D22 pin function selection:
	00= lcd_d22;
	01= enet1_1588_event3_out;
	10= etm_da1;
	11= GPIO.

11–10 BANK1_PIN21	Pin 122, LCD_D21 pin function selection:
	00= lcd_d21;
	01= enet1_1588_event2_in;
	10= etm_da2;
	11= GPIO.

9–8 BANK1_PIN20	Pin 107, LCD_D20 pin function selection:
	00= lcd_d20;
	01= enet1_1588_event2_out;
	10= etm_da3;
	11= GPIO.

7–6 BANK1_PIN19	Pin 112, LCD_D19 pin function selection:
	00= lcd_d19;
	01= reserved;
	10= etm_da4;
	11= GPIO.

5–4 BANK1_PIN18	Pin 118, LCD_D18 pin function selection:
	00= lcd_d18;
	01= reserved;
	10= etm_da5;
	11= GPIO.

3–2 BANK1_PIN17	Pin 105, LCD_D17 pin function selection:
	00= lcd_d17;
	01= reserved;
	10= etm_da6;

HW_PINCTRL_MUXSEL3 field descriptions (continued)

Field	Description
	11= GPIO.

1–0
BANK1_PIN16
	Pin 104, LCD_D16 pin function selection:
	00= lcd_d16;
	01= reserved;
	10= etm_da7;
	11= GPIO.

9.4.6 PINCTRL Pin Mux Select Register 4 (HW_PINCTRL_MUXSEL4)

The PINCTRL Pin Mux Select Register provides pin function selection for 15 pins in bank2.

HW_PINCTRL_MUXSEL4: 0x140

HW_PINCTRL_MUXSEL4_SET: 0x144

HW_PINCTRL_MUXSEL4_CLR: 0x148

HW_PINCTRL_MUXSEL4_TOG: 0x14C

This register allows the programmer to select which hardware interface blocks drive the 15 pins shown above.

Address: HW_PINCTRL_MUXSEL4 – 8001_8000h base + 140h offset = 8001_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BANK2_PIN15		BANK2_PIN14		BANK2_PIN13		BANK2_PIN12		RSRVD0		BANK2_PIN10		BANK2_PIN09		BANK2_PIN08	
W									▨							
Reset	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK2_PIN07		BANK2_PIN06		BANK2_PIN05		BANK2_PIN04		BANK2_PIN03		BANK2_PIN02		BANK2_PIN01		BANK2_PIN00	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL4 field descriptions

Field	Description
31–30 BANK2_PIN15	Pin 23, SSP1_DATA3 pin function selection: 00= ssp1_d3;

HW_PINCTRL_MUXSEL4 field descriptions (continued)

Field	Description

	01= ssp2_d7;
	10= enet0_1588_event3_in;
	11= GPIO.

29–28
BANK2_PIN14
	Pin 21, SSP1_DATA0 pin function selection:
	00= ssp1_d0;
	01= ssp2_d6;
	10= enet0_1588_event3_out;
	11= GPIO.

27–26
BANK2_PIN13
	Pin 17, SSP1_CMD pin function selection:
	00= ssp1_cmd;
	01= ssp2_d2;
	10= enet0_1588_event2_in;
	11= GPIO.

25–24
BANK2_PIN12
	Pin 11, SSP1_SCK pin function selection:
	00= ssp1_sck;
	01= ssp2_d1;
	10= enet0_1588_event2_out;
	11= GPIO.

23–22 RSRVD0	Always write zeroes to this field.

21–20
BANK2_PIN10
	Pin 268, SSP0_SCK pin function selection:
	00= ssp0_sck;
	01= reserved;
	10= reserved;
	11= GPIO.

19–18
BANK2_PIN09
	Pin 275, SSP0_DETECT pin function selection:
	00= ssp0_card_detect;
	01= reserved;
	10= reserved;
	11= GPIO.

17–16
BANK2_PIN08
	Pin 276, SSP0_CMD pin function selection:
	00= ssp0_cmd;
	01= reserved;
	10= reserved;
	11= GPIO.

HW_PINCTRL_MUXSEL4 field descriptions (continued)

Field	Description

15–14
BANK2_PIN07
	Pin 282, SSP0_DATA7 pin function selection:
	00= ssp0_d7;
	01= ssp2_sck;
	10= reserved;
	11= GPIO.

13–12
BANK2_PIN06
	Pin 6, SSP0_DATA6 pin function selection:
	00= ssp0_d6;
	01= ssp2_cmd;
	10= reserved;
	11= GPIO.

11–10
BANK2_PIN05
	Pin 284, SSP0_DATA5 pin function selection:
	00= ssp0_d5;
	01= ssp2_d3;
	10= reserved;
	11= GPIO.

9–8
BANK2_PIN04
	Pin 278, SSP0_DATA4 pin function selection:
	00= ssp0_d4;
	01= ssp2_d0;
	10= reserved;
	11= GPIO.

7–6
BANK2_PIN03
	Pin 274, SSP0_DATA3 pin function selection:
	00= ssp0_d3;
	01= reserved;
	10= reserved;
	11= GPIO.

5–4
BANK2_PIN02
	Pin 2, SSP0_DATA2 pin function selection:
	00= ssp0_d2;
	01= reserved;
	10= reserved;
	11= GPIO.

3–2
BANK2_PIN01
	Pin 289, SSP0_DATA1 pin function selection:
	00= ssp0_d1;
	01= reserved;
	10= reserved;
	11= GPIO.

HW_PINCTRL_MUXSEL4 field descriptions (continued)

Field	Description

1–0
BANK2_PIN00
	Pin 270, SSP0_DATA0 pin function selection:
	00= ssp0_d0;
	01= reserved;
	10= reserved;
	11= GPIO.

9.4.7 PINCTRL Pin Mux Select Register 5 (HW_PINCTRL_MUXSEL5)

The PINCTRL Pin Mux Select Register provides pin function selection for 10 pins in bank2.

HW_PINCTRL_MUXSEL5: 0x150

HW_PINCTRL_MUXSEL5_SET: 0x154

HW_PINCTRL_MUXSEL5_CLR: 0x158

HW_PINCTRL_MUXSEL5_TOG: 0x15C

This register allows the programmer to select which hardware interface blocks drive the 10 pins shown above.

Address: HW_PINCTRL_MUXSEL5 – 8001_8000h base + 150h offset = 8001_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1								BANK2_ PIN27		BANK2_ PIN26		BANK2_ PIN25		BANK2_ PIN24	
W	▨															
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0				BANK2_ PIN21		BANK2_ PIN20		BANK2_ PIN19		BANK2_ PIN18		BANK2_ PIN17		BANK2_ PIN16	
W	▨															
Reset	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL5 field descriptions

Field	Description
31–24 RSRVD1	Always write zeroes to this field.
23–22 BANK2_PIN27	Pin 15, SSP3_SS0 pin function selection: 00= ssp3_d3;

HW_PINCTRL_MUXSEL5 field descriptions (continued)

Field	Description

	01= auart4_cts;
	10= enet1_1588_event1_in;
	11= GPIO.

21–20
BANK2_PIN26
	Pin 3, SSP3_MISO pin function selection:
	00= ssp3_d0;
	01= auart4_rts;
	10= enet1_1588_event1_out;
	11= GPIO.

19–18
BANK2_PIN25
	Pin 9, SSP3_MOSI pin function selection:
	00= ssp3_cmd;
	01= auart4_rx;
	10= enet1_1588_event0_in;
	11= GPIO.

17–16
BANK2_PIN24
	Pin 286, SSP3_SCK pin function selection:
	00= ssp3_sck;
	01= auart4_tx;
	10= enet1_1588_event0_out;
	11= GPIO.

15–12 RSRVD0	Always write zeroes to this field.

11–10
BANK2_PIN21
	Pin 18, SSP2_SS2 pin function selection:
	00= ssp2_d5;
	01= ssp2_d2;
	10= usb0_overcurrent;
	11= GPIO.

9–8
BANK2_PIN20
	Pin 7, SSP2_SS1 pin function selection:
	00= ssp2_d4;
	01= ssp2_d1;
	10= usb1_overcurrent;
	11= GPIO.

7–6
BANK2_PIN19
	Pin 4, SSP2_SS0 pin function selection:
	00= ssp2_d3;
	01= auart3_tx;
	10= saif1_sdata2;
	11= GPIO.

HW_PINCTRL_MUXSEL5 field descriptions (continued)

Field	Description

5–4 BANK2_PIN18	Pin 288, SSP2_MISO pin function selection:
	00= ssp2_d0;
	01= auart3_rx;
	10= saif1_sdata1;
	11= GPIO.

3–2 BANK2_PIN17	Pin 1, SSP2_MOSI pin function selection:
	00= ssp2_cmd;
	01= auart2_tx;
	10= saif0_sdata2;
	11= GPIO.

1–0 BANK2_PIN16	Pin 280, SSP2_SCK pin function selection:
	00= ssp2_sck;
	01= auart2_rx;
	10= saif0_sdata1;
	11= GPIO.

9.4.8 PINCTRL Pin Mux Select Register 6 (HW_PINCTRL_MUXSEL6)

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank3.

HW_PINCTRL_MUXSEL6: 0x160

HW_PINCTRL_MUXSEL6_SET: 0x164

HW_PINCTRL_MUXSEL6_CLR: 0x168

HW_PINCTRL_MUXSEL6_TOG: 0x16C

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

Address: HW_PINCTRL_MUXSEL6 – 8001_8000h base + 160h offset = 8001_8160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK3_ PIN15		BANK3_ PIN14		BANK3_ PIN13		BANK3_ PIN12		BANK3_ PIN11		BANK3_ PIN10		BANK3_ PIN09		BANK3_ PIN08	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK3_PIN07		BANK3_PIN06		BANK3_PIN05		BANK3_PIN04		BANK3_PIN03		BANK3_PIN02		BANK3_PIN01		BANK3_PIN00	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL6 field descriptions

Field	Description

31–30
BANK3_PIN15
	Pin 82, AUART3_RTS pin function selection:
	00= auart3_rts;
	01= can1_rx;
	10= enet0_1588_event1_in;
	11= GPIO.

29–28
BANK3_PIN14
	Pin 90, AUART3_CTS pin function selection:
	00= auart3_cts;
	01= can1_tx;
	10= enet0_1588_event1_out;
	11= GPIO.

27–26
BANK3_PIN13
	Pin 86, AUART3_TX pin function selection:
	00= auart3_tx;
	01= can0_rx;
	10= enet0_1588_event0_in;
	11= GPIO.

25–24
BANK3_PIN12
	Pin 98, AUART3_RX pin function selection:
	00= auart3_rx;
	01= can0_tx;
	10= enet0_1588_event0_out;
	11= GPIO.

23–22
BANK3_PIN11
	Pin 56, AUART2_RTS pin function selection:
	00= auart2_rts;
	01= i2c1_sda;
	10= saif1_lrclk;
	11= GPIO.

21–20
BANK3_PIN10
	Pin 50, AUART2_CTS pin function selection:
	00= auart2_cts;
	01= i2c1_scl;
	10= saif1_bitclk;
	11= GPIO.

HW_PINCTRL_MUXSEL6 field descriptions (continued)

Field	Description

19–18
BANK3_PIN09
	Pin 26, AUART2_TX pin function selection:
	00= auart2_tx;
	01= ssp3_d2;
	10= ssp3_d5;
	11= GPIO.

17–16
BANK3_PIN08
	Pin 22, AUART2_RX pin function selection:
	00= auart2_rx;
	01= ssp3_d1;
	10= ssp3_d4;
	11= GPIO.

15–14
BANK3_PIN07
	Pin 74, AUART1_RTS pin function selection:
	00= auart1_rts;
	01= usb0_id;
	10= timrot_rotaryb;
	11= GPIO.

13–12
BANK3_PIN06
	Pin 78, AUART1_CTS pin function selection:
	00= auart1_cts;
	01= usb0_overcurrent;
	10= timrot_rotarya;
	11= GPIO.

11–10
BANK3_PIN05
	Pin 65, AUART1_TX pin function selection:
	00= auart1_tx;
	01= ssp3_card_detect;
	10= pwm_1;
	11= GPIO.

9–8
BANK3_PIN04
	Pin 81, AUART1_RX pin function selection:
	00= auart1_rx;
	01= ssp2_card_detect;
	10= pwm_0;
	11= GPIO.

7–6
BANK3_PIN03
	Pin 66, AUART0_RTS pin function selection:
	00= auart0_rts;
	01= auart4_tx;
	10= duart_tx;
	11= GPIO.

HW_PINCTRL_MUXSEL6 field descriptions (continued)

Field	Description

5–4
BANK3_PIN02
	Pin 70, AUART0_CTS pin function selection:
	00= auart0_cts;
	01= auart4_rx;
	10= duart_rx;
	11= GPIO.

3–2
BANK3_PIN01
	Pin 38, AUART0_TX pin function selection:
	00= auart0_tx;
	01= i2c0_sda;
	10= duart_rts;
	11= GPIO.

1–0
BANK3_PIN00
	Pin 30, AUART0_RX pin function selection:
	00= auart0_rx;
	01= i2c0_scl;
	10= duart_cts;
	11= GPIO.

9.4.9 PINCTRL Pin Mux Select Register 7 (HW_PINCTRL_MUXSEL7)

The PINCTRL Pin Mux Select Register provides pin function selection for 14 pins in bank3.

HW_PINCTRL_MUXSEL7: 0x170

HW_PINCTRL_MUXSEL7_SET: 0x174

HW_PINCTRL_MUXSEL7_CLR: 0x178

HW_PINCTRL_MUXSEL7_TOG: 0x17C

This register allows the programmer to select which hardware interface blocks drive the 14 pins shown above.

Address: HW_PINCTRL_MUXSEL7 – 8001_8000h base + 170h offset = 8001_8170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1		BANK3_ PIN30		BANK3_ PIN29		BANK3_ PIN28		BANK3_ PIN27		BANK3_ PIN26		BANK3_ PIN25		BANK3_ PIN24	
W	▨	▨														
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK3_PIN23		BANK3_PIN22		BANK3_PIN21		BANK3_PIN20		RSRVD0		BANK3_PIN18		BANK3_PIN17		BANK3_PIN16	
W									▨							
Reset	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1

HW_PINCTRL_MUXSEL7 field descriptions

Field	Description
31–30 RSRVD1	Always write zeroes to this field.

29–28
BANK3_PIN30
	Pin 101, LCD_RESET pin function selection:
	00= lcd_reset;
	01= lcd_vsync;
	10= reserved;
	11= GPIO.

27–26
BANK3_PIN29
	Pin 279, PWM4 pin function selection:
	00= pwm_4;
	01= reserved;
	10= reserved;
	11= GPIO.

25–24
BANK3_PIN28
	Pin 287, PWM3 pin function selection:
	00= pwm_3;
	01= reserved;
	10= reserved;
	11= GPIO.

23–22
BANK3_PIN27
	Pin 285, SPDIF pin function selection:
	00= spdif_tx;
	01= reserved;
	10= enet1_rx_er;
	11= GPIO.

21–20
BANK3_PIN26
	Pin 8, SAIF1_SDATA0 pin function selection:
	00= saif1_sdata0;
	01= pwm_7;
	10= saif0_sdata1;
	11= GPIO.

19–18
BANK3_PIN25
	Pin 281, I2C0_SDA pin function selection:
	00= i2c0_sda;
	01= timrot_rotaryb;

HW_PINCTRL_MUXSEL7 field descriptions (continued)

Field	Description
	10= duart_tx; 11= GPIO.

17–16
BANK3_PIN24
	Pin 272, I2C0_SCL pin function selection:
	00= i2c0_scl;
	01= timrot_rotarya;
	10= duart_rx;
	11= GPIO.

15–14
BANK3_PIN23
	Pin 12, SAIF0_SDATA0 pin function selection:
	00= saif0_sdata0;
	01= pwm_6;
	10= auart4_tx;
	11= GPIO.

13–12
BANK3_PIN22
	Pin 16, SAIF0_BITCLK pin function selection:
	00= saif0_bitclk;
	01= pwm_5;
	10= auart4_rx;
	11= GPIO.

11–10
BANK3_PIN21
	Pin 34, SAIF0_LRCLK pin function selection:
	00= saif0_lrclk;
	01= pwm_4;
	10= auart4_rts;
	11= GPIO.

9–8
BANK3_PIN20
	Pin 28, SAIF0_MCLK pin function selection:
	00= saif0_mclk;
	01= pwm_3;
	10= auart4_cts;
	11= GPIO.

7–6 RSRVD0	Always write zeroes to this field.

5–4
BANK3_PIN18
	Pin 68, PWM2 pin function selection:
	00= pwm_2;
	01= usb0_id;
	10= usb1_overcurrent;
	11= GPIO.

HW_PINCTRL_MUXSEL7 field descriptions (continued)

Field	Description

3–2
BANK3_PIN17
	Pin 84, PWM1 pin function selection:
	00= pwm_1;
	01= i2c1_sda;
	10= duart_tx;
	11= GPIO.

1–0
BANK3_PIN16
	Pin 72, PWM0 pin function selection:
	00= pwm_0;
	01= i2c1_scl;
	10= duart_rx;
	11= GPIO.

9.4.10 PINCTRL Pin Mux Select Register 8 (HW_PINCTRL_MUXSEL8)

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank4.

HW_PINCTRL_MUXSEL8: 0x180

HW_PINCTRL_MUXSEL8_SET: 0x184

HW_PINCTRL_MUXSEL8_CLR: 0x188

HW_PINCTRL_MUXSEL8_TOG: 0x18C

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

Address: HW_PINCTRL_MUXSEL8 – 8001_8000h base + 180h offset = 8001_8180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK4_ PIN15		BANK4_ PIN14		BANK4_ PIN13		BANK4_ PIN12		BANK4_ PIN11		BANK4_ PIN10		BANK4_ PIN09		BANK4_ PIN08	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK4_ PIN07		BANK4_ PIN06		BANK4_ PIN05		BANK4_ PIN04		BANK4_ PIN03		BANK4_ PIN02		BANK4_ PIN01		BANK4_ PIN00	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL8 field descriptions

Field	Description

31–30 BANK4_PIN15	Pin 61, ENET0_CRS pin function selection:
	00= enet0_crs;
	01= enet1_rx_en;
	10= enet0_1588_event3_in;
	11= GPIO.

29–28 BANK4_PIN14	Pin 57, ENET0_COL pin function selection:
	00= enet0_col;
	01= enet1_tx_en;
	10= enet0_1588_event3_out;
	11= GPIO.

27–26 BANK4_PIN13	Pin 31, ENET0_RX_CLK pin function selection:
	00= enet0_rx_clk;
	01= enet0_rx_er;
	10= enet0_1588_event2_in;
	11= GPIO.

25–24 BANK4_PIN12	Pin 41, ENET0_TXD3 pin function selection:
	00= enet0_txd3;
	01= enet1_txd1;
	10= enet0_1588_event1_in;
	11= GPIO.

23–22 BANK4_PIN11	Pin 43, ENET0_TXD2 pin function selection:
	00= enet0_txd2;
	01= enet1_txd0;
	10= enet0_1588_event1_out;
	11= GPIO.

21–20 BANK4_PIN10	Pin 53, ENET0_RXD3 pin function selection:
	00= enet0_rxd3;
	01= enet1_rxd1;
	10= enet0_1588_event0_in;
	11= GPIO.

19–18 BANK4_PIN09	Pin 51, ENET0_RXD2 pin function selection:
	00= enet0_rxd2;
	01= enet1_rxd0;
	10= enet0_1588_event0_out;
	11= GPIO.

HW_PINCTRL_MUXSEL8 field descriptions (continued)

Field	Description

17–16
BANK4_PIN08
	Pin 35, ENET0_TXD1 pin function selection:
	00= enet0_txd1;
	01= gpmi_ready7;
	10= reserved;
	11= GPIO.

15–14
BANK4_PIN07
	Pin 37, ENET0_TXD0 pin function selection:
	00= enet0_txd0;
	01= gpmi_ready6;
	10= reserved;
	11= GPIO.

13–12
BANK4_PIN06
	Pin 29, ENET0_TX_EN pin function selection:
	00= enet0_tx_en;
	01= gpmi_ready5;
	10= reserved;
	11= GPIO.

11–10
BANK4_PIN05
	Pin 13, ENET0_TX_CLK pin function selection:
	00= enet0_tx_clk;
	01= hsadc_trigger;
	10= enet0_1588_event2_out;
	11= GPIO.

9–8
BANK4_PIN04
	Pin 47, ENET0_RXD1 pin function selection:
	00= enet0_rxd1;
	01= gpmi_ready4;
	10= reserved;
	11= GPIO.

7–6
BANK4_PIN03
	Pin 45, ENET0_RXD0 pin function selection:
	00= enet0_rxd0;
	01= gpmi_ce7n;
	10= saif1_sdata2;
	11= GPIO.

5–4
BANK4_PIN02
	Pin 27, ENET0_RX_EN pin function selection:
	00= enet0_rx_en;
	01= gpmi_ce6n;
	10= saif1_sdata1;
	11= GPIO.

HW_PINCTRL_MUXSEL8 field descriptions (continued)

Field	Description

3–2
BANK4_PIN01
	Pin 39, ENET0_MDIO pin function selection:
	00= enet0_mdio;
	01= gpmi_ce5n;
	10= saif0_sdata2;
	11= GPIO.

1–0
BANK4_PIN00
	Pin 54, ENET0_MDC pin function selection:
	00= enet0_mdc;
	01= gpmi_ce4n;
	10= saif0_sdata1;
	11= GPIO.

9.4.11 PINCTRL Pin Mux Select Register 9 (HW_PINCTRL_MUXSEL9)

The PINCTRL Pin Mux Select Register provides pin function selection for 2 pins in bank4.

HW_PINCTRL_MUXSEL9: 0x190

HW_PINCTRL_MUXSEL9_SET: 0x194

HW_PINCTRL_MUXSEL9_CLR: 0x198

HW_PINCTRL_MUXSEL9_TOG: 0x19C

This register allows the programmer to select which hardware interface blocks drive the 2 pins shown above.

Address: HW_PINCTRL_MUXSEL9 – 8001_8000h base + 190h offset = 8001_8190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:10]						BANK4_ PIN20		RSRVD0						BANK4_ PIN16	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

HW_PINCTRL_MUXSEL9 field descriptions

Field	Description
31–10 RSRVD1	Always write zeroes to this field.

9–8
BANK4_PIN20
	Pin 230, JTAG_RTCK pin function selection:
	00= jtag_rtck;
	01= reserved;
	10= reserved;
	11= GPIO.

7–2 RSRVD0	Always write zeroes to this field.

1–0
BANK4_PIN16
	Pin 19, ENET_CLK pin function selection:
	00= clkctrl_enet;
	01= reserved;
	10= reserved;
	11= GPIO.

9.4.12 PINCTRL Pin Mux Select Register 10 (HW_PINCTRL_MUXSEL10)

The PINCTRL Pin Mux Select Register provides pin function selection for 16 pins in bank5.

HW_PINCTRL_MUXSEL10: 0x1a0

HW_PINCTRL_MUXSEL10_SET: 0x1a4

HW_PINCTRL_MUXSEL10_CLR: 0x1a8

HW_PINCTRL_MUXSEL10_TOG: 0x1aC

This register allows the programmer to select which hardware interface blocks drive the 16 pins shown above.

Address: HW_PINCTRL_MUXSEL10 – 8001_8000h base + 1A0h offset = 8001_81A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK5_PIN15		BANK5_PIN14		BANK5_PIN13		BANK5_PIN12		BANK5_PIN11		BANK5_PIN10		BANK5_PIN09		BANK5_PIN08	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK5_ PIN07		BANK5_ PIN06		BANK5_ PIN05		BANK5_ PIN04		BANK5_ PIN03		BANK5_ PIN02		BANK5_ PIN01		BANK5_ PIN00	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL10 field descriptions

Field	Description

31–30
BANK5_PIN15
	Pin 218, EMI_D15 pin function selection:
	00= emi_data15;
	01= reserved;
	10= reserved;
	11= disabled.

29–28
BANK5_PIN14
	Pin 223, EMI_D14 pin function selection:
	00= emi_data14;
	01= reserved;
	10= reserved;
	11= disabled.

27–26
BANK5_PIN13
	Pin 208, EMI_D13 pin function selection:
	00= emi_data13;
	01= reserved;
	10= reserved;
	11= disabled.

25–24
BANK5_PIN12
	Pin 213, EMI_D12 pin function selection:
	00= emi_data12;
	01= reserved;
	10= reserved;
	11= disabled.

23–22
BANK5_PIN11
	Pin 207, EMI_D11 pin function selection:
	00= emi_data11;
	01= reserved;
	10= reserved;
	11= disabled.

21–20
BANK5_PIN10
	Pin 217, EMI_D10 pin function selection:
	00= emi_data10;
	01= reserved;
	10= reserved;
	11= disabled.

HW_PINCTRL_MUXSEL10 field descriptions (continued)

Field	Description

19–18
BANK5_PIN09
	Pin 212, EMI_D09 pin function selection:
	00= emi_data9;
	01= reserved;
	10= reserved;
	11= disabled.

17–16
BANK5_PIN08
	Pin 216, EMI_D08 pin function selection:
	00= emi_data8;
	01= reserved;
	10= reserved;
	11= disabled.

15–14
BANK5_PIN07
	Pin 193, EMI_D07 pin function selection:
	00= emi_data7;
	01= reserved;
	10= reserved;
	11= disabled.

13–12
BANK5_PIN06
	Pin 189, EMI_D06 pin function selection:
	00= emi_data6;
	01= reserved;
	10= reserved;
	11= disabled.

11–10
BANK5_PIN05
	Pin 182, EMI_D05 pin function selection:
	00= emi_data5;
	01= reserved;
	10= reserved;
	11= disabled.

9–8
BANK5_PIN04
	Pin 180, EMI_D04 pin function selection:
	00= emi_data4;
	01= reserved;
	10= reserved;
	11= disabled.

7–6
BANK5_PIN03
	Pin 184, EMI_D03 pin function selection:
	00= emi_data3;
	01= reserved;
	10= reserved;
	11= disabled.

HW_PINCTRL_MUXSEL10 field descriptions (continued)

Field	Description

5–4
BANK5_PIN02
	Pin 177, EMI_D02 pin function selection:
	00= emi_data2;
	01= reserved;
	10= reserved;
	11= disabled.

3–2
BANK5_PIN01
	Pin 188, EMI_D01 pin function selection:
	00= emi_data1;
	01= reserved;
	10= reserved;
	11= disabled.

1–0
BANK5_PIN00
	Pin 185, EMI_D00 pin function selection:
	00= emi_data0;
	01= reserved;
	10= reserved;
	11= disabled.

9.4.13 PINCTRL Pin Mux Select Register 11 (HW_PINCTRL_MUXSEL11)

The PINCTRL Pin Mux Select Register provides pin function selection for 9 pins in bank5.

HW_PINCTRL_MUXSEL11: 0x1b0

HW_PINCTRL_MUXSEL11_SET: 0x1b4

HW_PINCTRL_MUXSEL11_CLR: 0x1b8

HW_PINCTRL_MUXSEL11_TOG: 0x1bC

This register allows the programmer to select which hardware interface blocks drive the 9 pins shown above.

Address: HW_PINCTRL_MUXSEL11 – 8001_8000h base + 1B0h offset = 8001_81B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1										BANK5_PIN26		RSRVD0			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	BANK5_ PIN23		BANK5_ PIN22		BANK5_ PIN21		BANK5_ PIN20		BANK5_ PIN19		BANK5_ PIN18		BANK5_ PIN17		BANK5_ PIN16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL11 field descriptions

Field	Description
31–22 RSRVD1	Always write zeroes to this field.

21–20
BANK5_PIN26
	Pin 196, EMI_DDR_OPEN pin function selection:
	00= emi_ddr_open;
	01= reserved;
	10= reserved;
	11= disabled.

19–16 RSRVD0	Always write zeroes to this field.

15–14
BANK5_PIN23
	Pin 204, EMI_DQS1 pin function selection:
	00= emi_dqs1;
	01= reserved;
	10= reserved;
	11= disabled.

13–12
BANK5_PIN22
	Pin 202, EMI_DQS0 pin function selection:
	00= emi_dqs0;
	01= reserved;
	10= reserved;
	11= disabled.

11–10
BANK5_PIN21
	Pin 200, EMI_CLK pin function selection:
	00= emi_clk;
	01= reserved;
	10= reserved;
	11= disabled.

9–8
BANK5_PIN20
	Pin 195, EMI_DDR_OPEN_FB pin function selection:
	00= emi_ddr_open_feedback;
	01= reserved;
	10= reserved;
	11= disabled.

7–6 BANK5_PIN19	Pin 222, EMI_DQM1 pin function selection:

HW_PINCTRL_MUXSEL11 field descriptions (continued)

Field	Description

	00= emi_dqm1;
	01= reserved;
	10= reserved;
	11= disabled.

5–4
BANK5_PIN18
	Pin 171, EMI_ODT1 pin function selection:
	00= emi_odt1;
	01= reserved;
	10= reserved;
	11= disabled.

3–2
BANK5_PIN17
	Pin 183, EMI_DQM0 pin function selection:
	00= emi_dqm0;
	01= reserved;
	10= reserved;
	11= disabled.

1–0
BANK5_PIN16
	Pin 173, EMI_ODT0 pin function selection:
	00= emi_odt0;
	01= reserved;
	10= reserved;
	11= disabled.

9.4.14 PINCTRL Pin Mux Select Register 12 (HW_PINCTRL_MUXSEL12)

The PINCTRL Pin Mux Select Register provides pin function selection for 15 pins in bank6.

HW_PINCTRL_MUXSEL12: 0x1c0

HW_PINCTRL_MUXSEL12_SET: 0x1c4

HW_PINCTRL_MUXSEL12_CLR: 0x1c8

HW_PINCTRL_MUXSEL12_TOG: 0x1cC

This register allows the programmer to select which hardware interface blocks drive the 15 pins shown above.

Address: HW_PINCTRL_MUXSEL12 – 8001_8000h base + 1C0h offset = 8001_81C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0		BANK6_PIN14		BANK6_PIN13		BANK6_PIN12		BANK6_PIN11		BANK6_PIN10		BANK6_PIN09		BANK6_PIN08	
W	▨															
Reset	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK6_PIN07		BANK6_PIN06		BANK6_PIN05		BANK6_PIN04		BANK6_PIN03		BANK6_PIN02		BANK6_PIN01		BANK6_PIN00	
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL12 field descriptions

Field	Description
31–30 RSRVD0	Always write zeroes to this field.

29–28
BANK6_PIN14
	Pin 147, EMI_A14 pin function selection:
	00= emi_addr14;
	01= reserved;
	10= reserved;
	11= disabled.

27–26
BANK6_PIN13
	Pin 142, EMI_A13 pin function selection:
	00= emi_addr13;
	01= reserved;
	10= reserved;
	11= disabled.

25–24
BANK6_PIN12
	Pin 150, EMI_A12 pin function selection:
	00= emi_addr12;
	01= reserved;
	10= reserved;
	11= disabled.

23–22
BANK6_PIN11
	Pin 146, EMI_A11 pin function selection:
	00= emi_addr11;
	01= reserved;
	10= reserved;
	11= disabled.

21–20 BANK6_PIN10	Pin 162, EMI_A10 pin function selection:

HW_PINCTRL_MUXSEL12 field descriptions (continued)

Field	Description

	00= emi_addr10;
	01= reserved;
	10= reserved;
	11= disabled.

19–18
BANK6_PIN09
	Pin 143, EMI_A09 pin function selection:
	00= emi_addr9;
	01= reserved;
	10= reserved;
	11= disabled.

17–16
BANK6_PIN08
	Pin 140, EMI_A08 pin function selection:
	00= emi_addr8;
	01= reserved;
	10= reserved;
	11= disabled.

15–14
BANK6_PIN07
	Pin 153, EMI_A07 pin function selection:
	00= emi_addr7;
	01= reserved;
	10= reserved;
	11= disabled.

13–12
BANK6_PIN06
	Pin 138, EMI_A06 pin function selection:
	00= emi_addr6;
	01= reserved;
	10= reserved;
	11= disabled.

11–10
BANK6_PIN05
	Pin 154, EMI_A05 pin function selection:
	00= emi_addr5;
	01= reserved;
	10= reserved;
	11= disabled.

9–8
BANK6_PIN04
	Pin 144, EMI_A04 pin function selection:
	00= emi_addr4;
	01= reserved;
	10= reserved;
	11= disabled.

HW_PINCTRL_MUXSEL12 field descriptions (continued)

Field	Description

7–6
BANK6_PIN03
	Pin 152, EMI_A03 pin function selection:
	00= emi_addr3;
	01= reserved;
	10= reserved;
	11= disabled.

5–4
BANK6_PIN02
	Pin 164, EMI_A02 pin function selection:
	00= emi_addr2;
	01= reserved;
	10= reserved;
	11= disabled.

3–2
BANK6_PIN01
	Pin 158, EMI_A01 pin function selection:
	00= emi_addr1;
	01= reserved;
	10= reserved;
	11= disabled.

1–0
BANK6_PIN00
	Pin 170, EMI_A00 pin function selection:
	00= emi_addr0;
	01= reserved;
	10= reserved;
	11= disabled.

9.4.15 PINCTRL Pin Mux Select Register 13 (HW_PINCTRL_MUXSEL13)

The PINCTRL Pin Mux Select Register provides pin function selection for 9 pins in bank6.

HW_PINCTRL_MUXSEL13: 0x1d0

HW_PINCTRL_MUXSEL13_SET: 0x1d4

HW_PINCTRL_MUXSEL13_CLR: 0x1d8

HW_PINCTRL_MUXSEL13_TOG: 0x1dC

This register allows the programmer to select which hardware interface blocks drive the 9 pins shown above.

Address: HW_PINCTRL_MUXSEL13 – 8001_8000h base + 1D0h offset = 8001_81D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0														BANK6_PIN24	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	BANK6_PIN23		BANK6_PIN22		BANK6_PIN21		BANK6_PIN20		BANK6_PIN19		BANK6_PIN18		BANK6_PIN17		BANK6_PIN16	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PINCTRL_MUXSEL13 field descriptions

Field	Description
31–18 RSRVD0	Always write zeroes to this field.

17–16
BANK6_PIN24
	Pin 166, EMI_CKE pin function selection:
	00= emi_cke;
	01= reserved;
	10= reserved;
	11= disabled.

15–14
BANK6_PIN23
	Pin 139, EMI_CE1N pin function selection:
	00= emi_ce1n;
	01= reserved;
	10= reserved;
	11= disabled.

13–12
BANK6_PIN22
	Pin 151, EMI_CE0N pin function selection:
	00= emi_ce0n;
	01= reserved;
	10= reserved;
	11= disabled.

11–10
BANK6_PIN21
	Pin 176, EMI_WEN pin function selection:
	00= emi_wen;
	01= reserved;
	10= reserved;
	11= disabled.

9–8 BANK6_PIN20	Pin 167, EMI_RASN pin function selection:

HW_PINCTRL_MUXSEL13 field descriptions (continued)

Field	Description

	00= emi_rasn;
	01= reserved;
	10= reserved;
	11= disabled.

7–6
BANK6_PIN19
	Pin 172, EMI_CASN pin function selection:
	00= emi_casn;
	01= reserved;
	10= reserved;
	11= disabled.

5–4
BANK6_PIN18
	Pin 165, EMI_BA2 pin function selection:
	00= emi_ba2;
	01= reserved;
	10= reserved;
	11= disabled.

3–2
BANK6_PIN17
	Pin 160, EMI_BA1 pin function selection:
	00= emi_ba1;
	01= reserved;
	10= reserved;
	11= disabled.

1–0
BANK6_PIN16
	Pin 157, EMI_BA0 pin function selection:
	00= emi_ba0;
	01= reserved;
	10= reserved;
	11= disabled.

9.4.16 PINCTRL Drive Strength and Voltage Register 0 (HW_PINCTRL_DRIVE0)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 0.

HW_PINCTRL_DRIVE0: 0x300

HW_PINCTRL_DRIVE0_SET: 0x304

HW_PINCTRL_DRIVE0_CLR: 0x308

HW_PINCTRL_DRIVE0_TOG: 0x30C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE0 – 8001_8000h base + 300h offset = 8001_8300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK0_PIN07_V	BANK0_ PIN07_MA		RSRVD6	BANK0_PIN06_V	BANK0_ PIN06_MA		RSRVD5	BANK0_PIN05_V	BANK0_ PIN05_MA		RSRVD4	BANK0_PIN04_V	BANK0_ PIN04_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK0_PIN03_V	BANK0_ PIN03_MA		RSRVD2	BANK0_PIN02_V	BANK0_ PIN02_MA		RSRVD1	BANK0_PIN01_V	BANK0_ PIN01_MA		RSRVD0	BANK0_PIN00_V	BANK0_ PIN00_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE0 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK0_PIN07_V	Pin 124, GPMI_D07 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK0_PIN07_
MA
	Pin 124, GPMI_D07 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK0_PIN06_V	Pin 130, GPMI_D06 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE0 field descriptions (continued)

Field	Description

25–24
BANK0_PIN06_
MA
	Pin 130, GPMI_D06 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK0_PIN05_V	Pin 116, GPMI_D05 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK0_PIN05_
MA
	Pin 116, GPMI_D05 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK0_PIN04_V	Pin 128, GPMI_D04 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK0_PIN04_
MA
	Pin 128, GPMI_D04 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK0_PIN03_V	Pin 132, GPMI_D03 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK0_PIN03_
MA
	Pin 132, GPMI_D03 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE0 field descriptions (continued)

Field	Description
11 RSRVD2	Always write zeroes to this field.
10 BANK0_PIN02_V	Pin 126, GPMI_D02 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK0_PIN02_
MA
	Pin 126, GPMI_D02 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK0_PIN01_V	Pin 136, GPMI_D01 pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK0_PIN01_
MA
	Pin 136, GPMI_D01 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK0_PIN00_V	Pin 134, GPMI_D00 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK0_PIN00_
MA
	Pin 134, GPMI_D00 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.17 PINCTRL Drive Strength and Voltage Register 1 (HW_PINCTRL_DRIVE1)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 0 pins of bank 0.

HW_PINCTRL_DRIVE1: 0x310

HW_PINCTRL_DRIVE1_SET: 0x314

HW_PINCTRL_DRIVE1_CLR: 0x318

HW_PINCTRL_DRIVE1_TOG: 0x31C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE1 – 8001_8000h base + 310h offset = 8001_8310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DRIVE1 field descriptions

Field	Description
31–0 RSRVD0	Always write zeroes to this field.

9.4.18 PINCTRL Drive Strength and Voltage Register 2 (HW_PINCTRL_DRIVE2)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 0.

HW_PINCTRL_DRIVE2: 0x320

HW_PINCTRL_DRIVE2_SET: 0x324

HW_PINCTRL_DRIVE2_CLR: 0x328

HW_PINCTRL_DRIVE2_TOG: 0x32C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE2 – 8001_8000h base + 320h offset = 8001_8320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK0_PIN23_V	BANK0_ PIN23_MA		RSRVD6	BANK0_PIN22_V	BANK0_ PIN22_MA		RSRVD5	BANK0_PIN21_V	BANK0_ PIN21_MA		RSRVD4	BANK0_PIN20_V	BANK0_ PIN20_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK0_PIN19_V	BANK0_ PIN19_MA		RSRVD2	BANK0_PIN18_V	BANK0_ PIN18_MA		RSRVD1	BANK0_PIN17_V	BANK0_ PIN17_MA		RSRVD0	BANK0_PIN16_V	BANK0_ PIN16_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE2 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK0_PIN23_V	Pin 80, GPMI_RDY3 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK0_PIN23_
MA
	Pin 80, GPMI_RDY3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK0_PIN22_V	Pin 88, GPMI_RDY2 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK0_PIN22_
MA
	Pin 88, GPMI_RDY2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE2 field descriptions (continued)

Field	Description
23 RSRVD5	Always write zeroes to this field.
22 BANK0_PIN21_V	Pin 119, GPMI_RDY1 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK0_PIN21_
MA
	Pin 119, GPMI_RDY1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK0_PIN20_V	Pin 103, GPMI_RDY0 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK0_PIN20_
MA
	Pin 103, GPMI_RDY0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK0_PIN19_V	Pin 135, GPMI_CE3N pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK0_PIN19_
MA
	Pin 135, GPMI_CE3N pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK0_PIN18_V	Pin 92, GPMI_CE2N pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE2 field descriptions (continued)

Field	Description

9–8
BANK0_PIN18_
MA
	Pin 92, GPMI_CE2N pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK0_PIN17_V	Pin 131, GPMI_CE1N pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK0_PIN17_
MA
	Pin 131, GPMI_CE1N pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK0_PIN16_V	Pin 115, GPMI_CE0N pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK0_PIN16_
MA
	Pin 115, GPMI_CE0N pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.19 PINCTRL Drive Strength and Voltage Register 3 (HW_PINCTRL_DRIVE3)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 5 pins of bank 0.

HW_PINCTRL_DRIVE3: 0x330

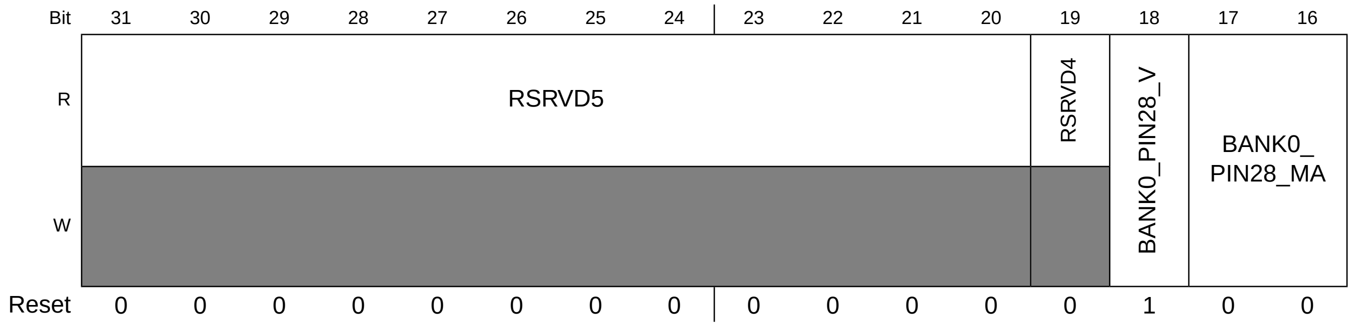
HW_PINCTRL_DRIVE3_SET: 0x334

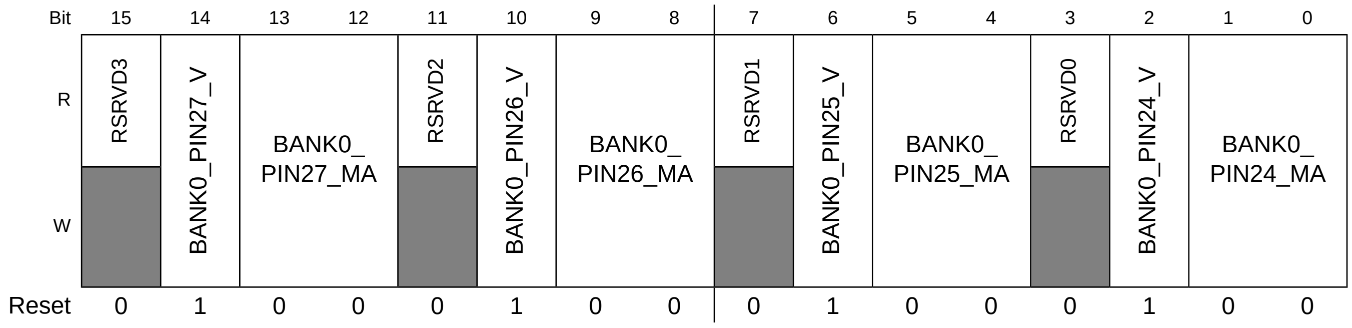
HW_PINCTRL_DRIVE3_CLR: 0x338

HW_PINCTRL_DRIVE3_TOG: 0x33C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE3 – 8001_8000h base + 330h offset = 8001_8330h





HW_PINCTRL_DRIVE3 field descriptions

Field	Description
31–20 RSRVD5	Always write zeroes to this field.
19 RSRVD4	Always write zeroes to this field.
18 BANK0_PIN28_V	Pin 129, GPMI_RESETN pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK0_PIN28_MA
	Pin 129, GPMI_RESETN pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.

HW_PINCTRL_DRIVE3 field descriptions (continued)

Field	Description
14 BANK0_PIN27_V	Pin 123, GPMI_CLE pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK0_PIN27_
MA
	Pin 123, GPMI_CLE pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK0_PIN26_V	Pin 117, GPMI_ALE pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK0_PIN26_
MA
	Pin 117, GPMI_ALE pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK0_PIN25_V	Pin 127, GPMI_WRN pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK0_PIN25_
MA
	Pin 127, GPMI_WRN pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK0_PIN24_V	Pin 110, GPMI_RDN pin voltage selection: 0= 1.8V; 1= 3.3V.
1–0 BANK0_PIN24_ MA	Pin 110, GPMI_RDN pin output drive strength selection: 00= 4 mA;

HW_PINCTRL_DRIVE3 field descriptions (continued)

Field	Description

	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.20 PINCTRL Drive Strength and Voltage Register 4 (HW_PINCTRL_DRIVE4)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 1.

HW_PINCTRL_DRIVE4: 0x340

HW_PINCTRL_DRIVE4_SET: 0x344

HW_PINCTRL_DRIVE4_CLR: 0x348

HW_PINCTRL_DRIVE4_TOG: 0x34C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE4 – 8001_8000h base + 340h offset = 8001_8340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK1_PIN07_V	BANK1_PIN07_MA		RSRVD6	BANK1_PIN06_V	BANK1_PIN06_MA		RSRVD5	BANK1_PIN05_V	BANK1_PIN05_MA		RSRVD4	BANK1_PIN04_V	BANK1_PIN04_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK1_PIN03_V	BANK1_PIN03_MA		RSRVD2	BANK1_PIN02_V	BANK1_PIN02_MA		RSRVD1	BANK1_PIN01_V	BANK1_PIN01_MA		RSRVD0	BANK1_PIN00_V	BANK1_PIN00_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE4 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK1_PIN07_V	Pin 75, LCD_D07 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK1_PIN07_
MA
	Pin 75, LCD_D07 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK1_PIN06_V	Pin 91, LCD_D06 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK1_PIN06_
MA
	Pin 91, LCD_D06 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK1_PIN05_V	Pin 89, LCD_D05 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK1_PIN05_
MA
	Pin 89, LCD_D05 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK1_PIN04_V	Pin 79, LCD_D04 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE4 field descriptions (continued)

Field	Description

17–16
BANK1_PIN04_MA
	Pin 79, LCD_D04 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK1_PIN03_V	Pin 77, LCD_D03 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK1_PIN03_MA
	Pin 77, LCD_D03 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK1_PIN02_V	Pin 67, LCD_D02 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK1_PIN02_MA
	Pin 67, LCD_D02 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK1_PIN01_V	Pin 69, LCD_D01 pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK1_PIN01_MA
	Pin 69, LCD_D01 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE4 field descriptions (continued)

Field	Description
3 RSRVD0	Always write zeroes to this field.
2 BANK1_PIN00_V	Pin 63, LCD_D00 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK1_PIN00_
MA
	Pin 63, LCD_D00 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.21 PINCTRL Drive Strength and Voltage Register 5 (HW_PINCTRL_DRIVE5)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 1.

HW_PINCTRL_DRIVE5: 0x350

HW_PINCTRL_DRIVE5_SET: 0x354

HW_PINCTRL_DRIVE5_CLR: 0x358

HW_PINCTRL_DRIVE5_TOG: 0x35C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE5 – 8001_8000h base + 350h offset = 8001_8350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK1_PIN15_V	BANK1_ PIN15_MA		RSRVD6	BANK1_PIN14_V	BANK1_ PIN14_MA		RSRVD5	BANK1_PIN13_V	BANK1_ PIN13_MA		RSRVD4	BANK1_PIN12_V	BANK1_ PIN12_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK1_PIN11_V	BANK1_PIN11_MA		RSRVD2	BANK1_PIN10_V	BANK1_PIN10_MA		RSRVD1	BANK1_PIN09_V	BANK1_PIN09_MA		RSRVD0	BANK1_PIN08_V	BANK1_PIN08_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE5 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK1_PIN15_V	Pin 114, LCD_D15 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK1_PIN15_MA
	Pin 114, LCD_D15 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK1_PIN14_V	Pin 106, LCD_D14 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK1_PIN14_MA
	Pin 106, LCD_D14 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK1_PIN13_V	Pin 102, LCD_D13 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK1_PIN13_MA
	Pin 102, LCD_D13 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;

HW_PINCTRL_DRIVE5 field descriptions (continued)

Field	Description

	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK1_PIN12_V	Pin 87, LCD_D12 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK1_PIN12_
MA
	Pin 87, LCD_D12 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK1_PIN11_V	Pin 95, LCD_D11 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK1_PIN11_
MA
	Pin 95, LCD_D11 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK1_PIN10_V	Pin 85, LCD_D10 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK1_PIN10_
MA
	Pin 85, LCD_D10 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK1_PIN09_V	Pin 99, LCD_D09 pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE5 field descriptions (continued)

Field	Description
	1= 3.3V.

5–4
BANK1_PIN09_
MA
	Pin 99, LCD_D09 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK1_PIN08_V	Pin 93, LCD_D08 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK1_PIN08_
MA
	Pin 93, LCD_D08 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.22 PINCTRL Drive Strength and Voltage Register 6 (HW_PINCTRL_DRIVE6)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 1.

HW_PINCTRL_DRIVE6: 0x360

HW_PINCTRL_DRIVE6_SET: 0x364

HW_PINCTRL_DRIVE6_CLR: 0x368

HW_PINCTRL_DRIVE6_TOG: 0x36C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE6 – 8001_8000h base + 360h offset = 8001_8360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK1_PIN23_V	BANK1_PIN23_MA		RSRVD6	BANK1_PIN22_V	BANK1_PIN22_MA		RSRVD5	BANK1_PIN21_V	BANK1_PIN21_MA		RSRVD4	BANK1_PIN20_V	BANK1_PIN20_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK1_PIN19_V	BANK1_PIN19_MA		RSRVD2	BANK1_PIN18_V	BANK1_PIN18_MA		RSRVD1	BANK1_PIN17_V	BANK1_PIN17_MA		RSRVD0	BANK1_PIN16_V	BANK1_PIN16_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE6 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK1_PIN23_V	Pin 108, LCD_D23 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK1_PIN23_
MA
	Pin 108, LCD_D23 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK1_PIN22_V	Pin 120, LCD_D22 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK1_PIN22_
MA
	Pin 120, LCD_D22 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE6 field descriptions (continued)

Field	Description
23 RSRVD5	Always write zeroes to this field.
22 BANK1_PIN21_V	Pin 122, LCD_D21 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK1_PIN21_
MA
	Pin 122, LCD_D21 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK1_PIN20_V	Pin 107, LCD_D20 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK1_PIN20_
MA
	Pin 107, LCD_D20 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK1_PIN19_V	Pin 112, LCD_D19 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK1_PIN19_
MA
	Pin 112, LCD_D19 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK1_PIN18_V	Pin 118, LCD_D18 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE6 field descriptions (continued)

Field	Description

9–8
BANK1_PIN18_MA
	Pin 118, LCD_D18 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK1_PIN17_V	Pin 105, LCD_D17 pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK1_PIN17_MA
	Pin 105, LCD_D17 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK1_PIN16_V	Pin 104, LCD_D16 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK1_PIN16_MA
	Pin 104, LCD_D16 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.23 PINCTRL Drive Strength and Voltage Register 7 (HW_PINCTRL_DRIVE7)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 1.

HW_PINCTRL_DRIVE7: 0x370

HW_PINCTRL_DRIVE7_SET: 0x374

HW_PINCTRL_DRIVE7_CLR: 0x378

HW_PINCTRL_DRIVE7_TOG: 0x37C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE7 – 8001_8000h base + 370h offset = 8001_8370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK1_PIN31_V	BANK1_ PIN31_MA		RSRVD6	BANK1_PIN30_V	BANK1_ PIN30_MA		RSRVD5	BANK1_PIN29_V	BANK1_ PIN29_MA		RSRVD4	BANK1_PIN28_V	BANK1_ PIN28_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK1_PIN27_V	BANK1_ PIN27_MA		RSRVD2	BANK1_PIN26_V	BANK1_ PIN26_MA		RSRVD1	BANK1_PIN25_V	BANK1_ PIN25_MA		RSRVD0	BANK1_PIN24_V	BANK1_ PIN24_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE7 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK1_PIN31_V	Pin 111, LCD_ENABLE pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK1_PIN31_
MA
	Pin 111, LCD_ENABLE pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK1_PIN30_V	Pin 73, LCD_DOTCLK pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE7 field descriptions (continued)

Field	Description
	1= 3.3V.

25–24 BANK1_PIN30_MA	Pin 73, LCD_DOTCLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK1_PIN29_V	Pin 71, LCD_HSYNC pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20 BANK1_PIN29_MA	Pin 71, LCD_HSYNC pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK1_PIN28_V	Pin 59, LCD_VSYNC pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16 BANK1_PIN28_MA	Pin 59, LCD_VSYNC pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK1_PIN27_V	Pin 113, LCD_CS pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12 BANK1_PIN27_MA	Pin 113, LCD_CS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;

HW_PINCTRL_DRIVE7 field descriptions (continued)

Field	Description
	11= Reserved;
11 RSRVD2	Always write zeroes to this field.
10 BANK1_PIN26_V	Pin 94, LCD_RS pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK1_PIN26_
MA
	Pin 94, LCD_RS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK1_PIN25_V	Pin 55, LCD_WR_RWN pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK1_PIN25_
MA
	Pin 55, LCD_WR_RWN pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK1_PIN24_V	Pin 96, LCD_RD_E pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK1_PIN24_
MA
	Pin 96, LCD_RD_E pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.24 PINCTRL Drive Strength and Voltage Register 8 (HW_PINCTRL_DRIVE8)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 2.

HW_PINCTRL_DRIVE8: 0x380

HW_PINCTRL_DRIVE8_SET: 0x384

HW_PINCTRL_DRIVE8_CLR: 0x388

HW_PINCTRL_DRIVE8_TOG: 0x38C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE8 – 8001_8000h base + 380h offset = 8001_8380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK2_PIN07_V	BANK2_PIN07_MA		RSRVD6	BANK2_PIN06_V	BANK2_PIN06_MA		RSRVD5	BANK2_PIN05_V	BANK2_PIN05_MA		RSRVD4	BANK2_PIN04_V	BANK2_PIN04_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK2_PIN03_V	BANK2_PIN03_MA		RSRVD2	BANK2_PIN02_V	BANK2_PIN02_MA		RSRVD1	BANK2_PIN01_V	BANK2_PIN01_MA		RSRVD0	BANK2_PIN00_V	BANK2_PIN00_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE8 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK2_PIN07_V	Pin 282, SSP0_DATA7 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE8 field descriptions (continued)

Field	Description

29–28 BANK2_PIN07_MA	Pin 282, SSP0_DATA7 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK2_PIN06_V	Pin 6, SSP0_DATA6 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24 BANK2_PIN06_MA	Pin 6, SSP0_DATA6 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK2_PIN05_V	Pin 284, SSP0_DATA5 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20 BANK2_PIN05_MA	Pin 284, SSP0_DATA5 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK2_PIN04_V	Pin 278, SSP0_DATA4 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16 BANK2_PIN04_MA	Pin 278, SSP0_DATA4 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE8 field descriptions (continued)

Field	Description
15 RSRVD3	Always write zeroes to this field.
14 BANK2_PIN03_V	Pin 274, SSP0_DATA3 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK2_PIN03_
MA
	Pin 274, SSP0_DATA3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK2_PIN02_V	Pin 2, SSP0_DATA2 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK2_PIN02_
MA
	Pin 2, SSP0_DATA2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK2_PIN01_V	Pin 289, SSP0_DATA1 pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK2_PIN01_
MA
	Pin 289, SSP0_DATA1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK2_PIN00_V	Pin 270, SSP0_DATA0 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE8 field descriptions (continued)

Field	Description

1–0
BANK2_PIN00_
MA
	Pin 270, SSP0_DATA0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.25 PINCTRL Drive Strength and Voltage Register 9 (HW_PINCTRL_DRIVE9)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 7 pins of bank 2.

HW_PINCTRL_DRIVE9: 0x390

HW_PINCTRL_DRIVE9_SET: 0x394

HW_PINCTRL_DRIVE9_CLR: 0x398

HW_PINCTRL_DRIVE9_TOG: 0x39C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE9 – 8001_8000h base + 390h offset = 8001_8390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK2_PIN15_V	BANK2_PIN15_MA		RSRVD6	BANK2_PIN14_V	BANK2_PIN14_MA		RSRVD5	BANK2_PIN13_V	BANK2_PIN13_MA		RSRVD4	BANK2_PIN12_V	BANK2_PIN12_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3				RSRVD2	BANK2_PIN10_V	BANK2_ PIN10_MA		RSRVD1	BANK2_PIN09_V	BANK2_ PIN09_MA		RSRVD0	BANK2_PIN08_V	BANK2_ PIN08_MA	
W	▨	▨	▨	▨	▨				▨				▨			
Reset	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE9 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK2_PIN15_V	Pin 23, SSP1_DATA3 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK2_PIN15_
MA
	Pin 23, SSP1_DATA3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK2_PIN14_V	Pin 21, SSP1_DATA0 pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK2_PIN14_
MA
	Pin 21, SSP1_DATA0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK2_PIN13_V	Pin 17, SSP1_CMD pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK2_PIN13_
MA
	Pin 17, SSP1_CMD pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;

HW_PINCTRL_DRIVE9 field descriptions (continued)

Field	Description

	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK2_PIN12_V	Pin 11, SSP1_SCK pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK2_PIN12_
MA
	Pin 11, SSP1_SCK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15–12 RSRVD3	Always write zeroes to this field.
11 RSRVD2	Always write zeroes to this field.
10 BANK2_PIN10_V	Pin 268, SSP0_SCK pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK2_PIN10_
MA
	Pin 268, SSP0_SCK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK2_PIN09_V	Pin 275, SSP0_DETECT pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK2_PIN09_
MA
	Pin 275, SSP0_DETECT pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.

HW_PINCTRL_DRIVE9 field descriptions (continued)

Field	Description
2 BANK2_PIN08_V	Pin 276, SSP0_CMD pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK2_PIN08_
MA
	Pin 276, SSP0_CMD pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.26 PINCTRL Drive Strength and Voltage Register 10 (HW_PINCTRL_DRIVE10)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 6 pins of bank 2.

HW_PINCTRL_DRIVE10: 0x3a0

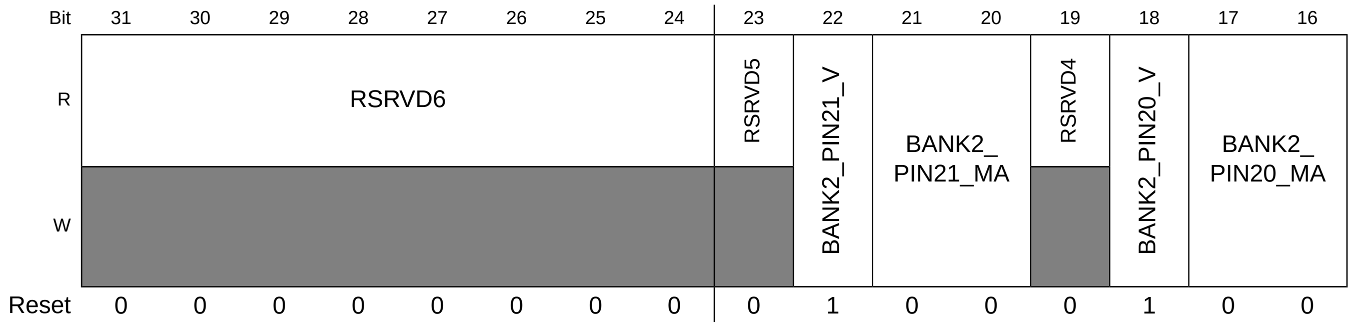
HW_PINCTRL_DRIVE10_SET: 0x3a4

HW_PINCTRL_DRIVE10_CLR: 0x3a8

HW_PINCTRL_DRIVE10_TOG: 0x3aC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE10 – 8001_8000h base + 3A0h offset = 8001_83A0h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK2_PIN19_V	BANK2_ PIN19_MA		RSRVD2	BANK2_PIN18_V	BANK2_ PIN18_MA		RSRVD1	BANK2_PIN17_V	BANK2_ PIN17_MA		RSRVD0	BANK2_PIN16_V	BANK2_ PIN16_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE10 field descriptions

Field	Description
31–24 RSRVD6	Always write zeroes to this field.
23 RSRVD5	Always write zeroes to this field.
22 BANK2_PIN21_V	Pin 18, SSP2_SS2 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK2_PIN21_
MA
	Pin 18, SSP2_SS2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK2_PIN20_V	Pin 7, SSP2_SS1 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK2_PIN20_
MA
	Pin 7, SSP2_SS1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK2_PIN19_V	Pin 4, SSP2_SS0 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE10 field descriptions (continued)

Field	Description

13–12 BANK2_PIN19_MA	Pin 4, SSP2_SS0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK2_PIN18_V	Pin 288, SSP2_MISO pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8 BANK2_PIN18_MA	Pin 288, SSP2_MISO pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK2_PIN17_V	Pin 1, SSP2_MOSI pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4 BANK2_PIN17_MA	Pin 1, SSP2_MOSI pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK2_PIN16_V	Pin 280, SSP2_SCK pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0 BANK2_PIN16_MA	Pin 280, SSP2_SCK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.27 PINCTRL Drive Strength and Voltage Register 11 (HW_PINCTRL_DRIVE11)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 4 pins of bank 2.

HW_PINCTRL_DRIVE11: 0x3b0

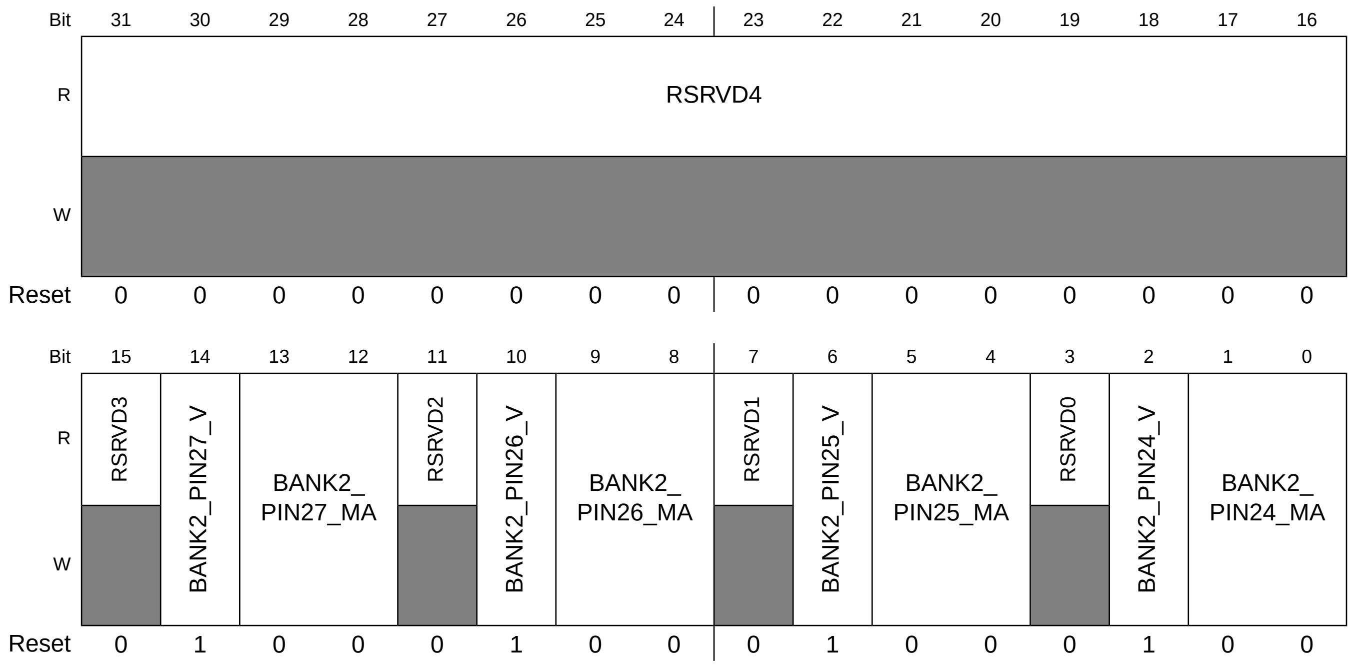
HW_PINCTRL_DRIVE11_SET: 0x3b4

HW_PINCTRL_DRIVE11_CLR: 0x3b8

HW_PINCTRL_DRIVE11_TOG: 0x3bC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE11 – 8001_8000h base + 3B0h offset = 8001_83B0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD4
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD3
BANK2_PIN27_V
BANK2_ PIN27_MA
RSRVD2
BANK2_PIN26_V
BANK2_ PIN26_MA
RSRVD1
BANK2_PIN25_V
BANK2_ PIN25_MA
RSRVD0
BANK2_PIN24_V
BANK2_ PIN24_MA
Reset 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0

HW_PINCTRL_DRIVE11 field descriptions

Field	Description
31–16 RSRVD4	Always write zeroes to this field.
15 RSRVD3	Always write zeroes to this field.
14 BANK2_PIN27_V	Pin 15, SSP3_SS0 pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE11 field descriptions (continued)

Field	Description

13–12
BANK2_PIN27_
MA
	Pin 15, SSP3_SS0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK2_PIN26_V	Pin 3, SSP3_MISO pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK2_PIN26_
MA
	Pin 3, SSP3_MISO pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK2_PIN25_V	Pin 9, SSP3_MOSI pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK2_PIN25_
MA
	Pin 9, SSP3_MOSI pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK2_PIN24_V	Pin 286, SSP3_SCK pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK2_PIN24_
MA
	Pin 286, SSP3_SCK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.28 PINCTRL Drive Strength and Voltage Register 12 (HW_PINCTRL_DRIVE12)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 3.

HW_PINCTRL_DRIVE12: 0x3c0

HW_PINCTRL_DRIVE12_SET: 0x3c4

HW_PINCTRL_DRIVE12_CLR: 0x3c8

HW_PINCTRL_DRIVE12_TOG: 0x3cC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE12 – 8001_8000h base + 3C0h offset = 8001_83C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK3_PIN07_V	BANK3_PIN07_MA		RSRVD6	BANK3_PIN06_V	BANK3_PIN06_MA		RSRVD5	BANK3_PIN05_V	BANK3_PIN05_MA		RSRVD4	BANK3_PIN04_V	BANK3_PIN04_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK3_PIN03_V	BANK3_PIN03_MA		RSRVD2	BANK3_PIN02_V	BANK3_PIN02_MA		RSRVD1	BANK3_PIN01_V	BANK3_PIN01_MA		RSRVD0	BANK3_PIN00_V	BANK3_PIN00_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE12 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK3_PIN07_V	Pin 74, AUART1_RTS pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE12 field descriptions (continued)

Field	Description

29–28
BANK3_PIN07_
MA
	Pin 74, AUART1_RTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK3_PIN06_V	Pin 78, AUART1_CTS pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK3_PIN06_
MA
	Pin 78, AUART1_CTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK3_PIN05_V	Pin 65, AUART1_TX pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK3_PIN05_
MA
	Pin 65, AUART1_TX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK3_PIN04_V	Pin 81, AUART1_RX pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK3_PIN04_
MA
	Pin 81, AUART1_RX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE12 field descriptions (continued)

Field	Description
15 RSRVD3	Always write zeroes to this field.
14 BANK3_PIN03_V	Pin 66, AUART0_RTS pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK3_PIN03_
MA
	Pin 66, AUART0_RTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK3_PIN02_V	Pin 70, AUART0_CTS pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK3_PIN02_
MA
	Pin 70, AUART0_CTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK3_PIN01_V	Pin 38, AUART0_TX pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK3_PIN01_
MA
	Pin 38, AUART0_TX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK3_PIN00_V	Pin 30, AUART0_RX pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE12 field descriptions (continued)

Field	Description

1–0
BANK3_PIN00_
MA
	Pin 30, AUART0_RX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.29 PINCTRL Drive Strength and Voltage Register 13 (HW_PINCTRL_DRIVE13)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 3.

HW_PINCTRL_DRIVE13: 0x3d0

HW_PINCTRL_DRIVE13_SET: 0x3d4

HW_PINCTRL_DRIVE13_CLR: 0x3d8

HW_PINCTRL_DRIVE13_TOG: 0x3dC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE13 – 8001_8000h base + 3D0h offset = 8001_83D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK3_PIN15_V	BANK3_PIN15_MA		RSRVD6	BANK3_PIN14_V	BANK3_PIN14_MA		RSRVD5	BANK3_PIN13_V	BANK3_PIN13_MA		RSRVD4	BANK3_PIN12_V	BANK3_PIN12_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK3_PIN11_V	BANK3_ PIN11_MA		RSRVD2	BANK3_PIN10_V	BANK3_ PIN10_MA		RSRVD1	BANK3_PIN09_V	BANK3_ PIN09_MA		RSRVD0	BANK3_PIN08_V	BANK3_ PIN08_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE13 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK3_PIN15_V	Pin 82, AUART3_RTS pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK3_PIN15_
MA
	Pin 82, AUART3_RTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK3_PIN14_V	Pin 90, AUART3_CTS pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK3_PIN14_
MA
	Pin 90, AUART3_CTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK3_PIN13_V	Pin 86, AUART3_TX pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK3_PIN13_
MA
	Pin 86, AUART3_TX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;

HW_PINCTRL_DRIVE13 field descriptions (continued)

Field	Description

	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK3_PIN12_V	Pin 98, AUART3_RX pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK3_PIN12_
MA
	Pin 98, AUART3_RX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK3_PIN11_V	Pin 56, AUART2_RTS pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK3_PIN11_
MA
	Pin 56, AUART2_RTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK3_PIN10_V	Pin 50, AUART2_CTS pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK3_PIN10_
MA
	Pin 50, AUART2_CTS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK3_PIN09_V	Pin 26, AUART2_TX pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE13 field descriptions (continued)

Field	Description
	1= 3.3V.

5–4
BANK3_PIN09_MA
	Pin 26, AUART2_TX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK3_PIN08_V	Pin 22, AUART2_RX pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK3_PIN08_MA
	Pin 22, AUART2_RX pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.30 PINCTRL Drive Strength and Voltage Register 14 (HW_PINCTRL_DRIVE14)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 7 pins of bank 3.

HW_PINCTRL_DRIVE14: 0x3e0

HW_PINCTRL_DRIVE14_SET: 0x3e4

HW_PINCTRL_DRIVE14_CLR: 0x3e8

HW_PINCTRL_DRIVE14_TOG: 0x3eC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE14 – 8001_8000h base + 3E0h offset = 8001_83E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK3_PIN23_V	BANK3_PIN23_MA		RSRVD6	BANK3_PIN22_V	BANK3_PIN22_MA		RSRVD5	BANK3_PIN21_V	BANK3_PIN21_MA		RSRVD4	BANK3_PIN20_V	BANK3_PIN20_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3				RSRVD2	BANK3_PIN18_V	BANK3_PIN18_MA		RSRVD1	BANK3_PIN17_V	BANK3_PIN17_MA		RSRVD0	BANK3_PIN16_V	BANK3_PIN16_MA	
W	▨	▨	▨	▨	▨				▨				▨			
Reset	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE14 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK3_PIN23_V	Pin 12, SAIF0_SDATA0 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK3_PIN23_
MA
	Pin 12, SAIF0_SDATA0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK3_PIN22_V	Pin 16, SAIF0_BITCLK pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK3_PIN22_
MA
	Pin 16, SAIF0_BITCLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE14 field descriptions (continued)

Field	Description
23 RSRVD5	Always write zeroes to this field.
22 BANK3_PIN21_V	Pin 34, SAIF0_LRCLK pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK3_PIN21_
MA
	Pin 34, SAIF0_LRCLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK3_PIN20_V	Pin 28, SAIF0_MCLK pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK3_PIN20_
MA
	Pin 28, SAIF0_MCLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15–12 RSRVD3	Always write zeroes to this field.
11 RSRVD2	Always write zeroes to this field.
10 BANK3_PIN18_V	Pin 68, PWM2 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK3_PIN18_
MA
	Pin 68, PWM2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK3_PIN17_V	Pin 84, PWM1 pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE14 field descriptions (continued)

Field	Description
	1= 3.3V.

5–4
BANK3_PIN17_MA
	Pin 84, PWM1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK3_PIN16_V	Pin 72, PWM0 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK3_PIN16_MA
	Pin 72, PWM0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.31 PINCTRL Drive Strength and Voltage Register 15 (HW_PINCTRL_DRIVE15)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 7 pins of bank 3.

HW_PINCTRL_DRIVE15: 0x3f0

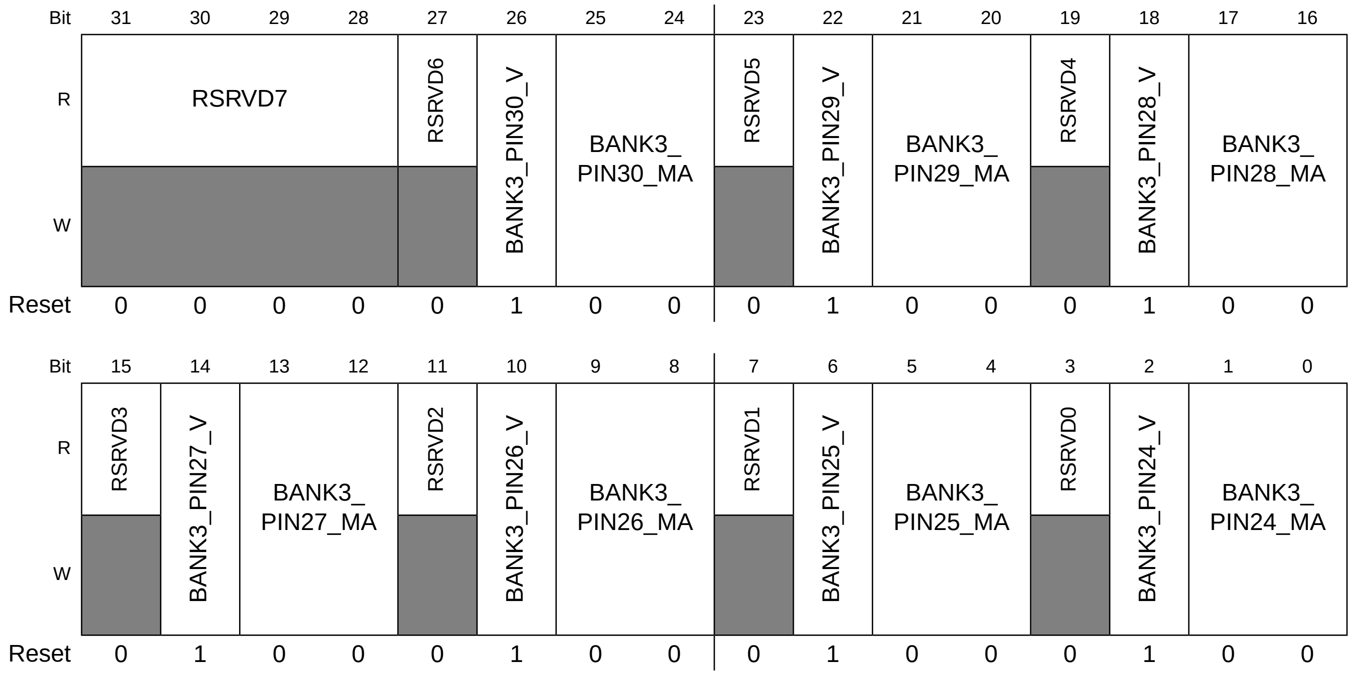
HW_PINCTRL_DRIVE15_SET: 0x3f4

HW_PINCTRL_DRIVE15_CLR: 0x3f8

HW_PINCTRL_DRIVE15_TOG: 0x3fC

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE15 – 8001_8000h base + 3F0h offset = 8001_83F0h



HW_PINCTRL_DRIVE15 field descriptions

Field	Description
31–28 RSRVD7	Always write zeroes to this field.
27 RSRVD6	Always write zeroes to this field.
26 BANK3_PIN30_V	Pin 101, LCD_RESET pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK3_PIN30_
MA
	Pin 101, LCD_RESET pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK3_PIN29_V	Pin 279, PWM4 pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK3_PIN29_
MA
	Pin 279, PWM4 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;

HW_PINCTRL_DRIVE15 field descriptions (continued)

Field	Description

	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK3_PIN28_V	Pin 287, PWM3 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK3_PIN28_
MA
	Pin 287, PWM3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK3_PIN27_V	Pin 285, SPDIF pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK3_PIN27_
MA
	Pin 285, SPDIF pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK3_PIN26_V	Pin 8, SAIF1_SDATA0 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK3_PIN26_
MA
	Pin 8, SAIF1_SDATA0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK3_PIN25_V	Pin 281, I2C0_SDA pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE15 field descriptions (continued)

Field	Description
	1= 3.3V.

5–4
BANK3_PIN25_MA
	Pin 281, I2C0_SDA pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK3_PIN24_V	Pin 272, I2C0_SCL pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK3_PIN24_MA
	Pin 272, I2C0_SCL pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.32 PINCTRL Drive Strength and Voltage Register 16 (HW_PINCTRL_DRIVE16)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 4.

HW_PINCTRL_DRIVE16: 0x400

HW_PINCTRL_DRIVE16_SET: 0x404

HW_PINCTRL_DRIVE16_CLR: 0x408

HW_PINCTRL_DRIVE16_TOG: 0x40C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE16 – 8001_8000h base + 400h offset = 8001_8400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK4_PIN07_V	BANK4_PIN07_MA		RSRVD6	BANK4_PIN06_V	BANK4_PIN06_MA		RSRVD5	BANK4_PIN05_V	BANK4_PIN05_MA		RSRVD4	BANK4_PIN04_V	BANK4_PIN04_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK4_PIN03_V	BANK4_PIN03_MA		RSRVD2	BANK4_PIN02_V	BANK4_PIN02_MA		RSRVD1	BANK4_PIN01_V	BANK4_PIN01_MA		RSRVD0	BANK4_PIN00_V	BANK4_PIN00_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE16 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK4_PIN07_V	Pin 37, ENET0_TXD0 pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK4_PIN07_MA
	Pin 37, ENET0_TXD0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK4_PIN06_V	Pin 29, ENET0_TX_EN pin voltage selection: 0= 1.8V; 1= 3.3V.

25–24
BANK4_PIN06_MA
	Pin 29, ENET0_TX_EN pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

HW_PINCTRL_DRIVE16 field descriptions (continued)

Field	Description
23 RSRVD5	Always write zeroes to this field.
22 BANK4_PIN05_V	Pin 13, ENET0_TX_CLK pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK4_PIN05_
MA
	Pin 13, ENET0_TX_CLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK4_PIN04_V	Pin 47, ENET0_RXD1 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK4_PIN04_
MA
	Pin 47, ENET0_RXD1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK4_PIN03_V	Pin 45, ENET0_RXD0 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK4_PIN03_
MA
	Pin 45, ENET0_RXD0 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

11 RSRVD2	Always write zeroes to this field.
10 BANK4_PIN02_V	Pin 27, ENET0_RX_EN pin voltage selection: 0= 1.8V; 1= 3.3V.

HW_PINCTRL_DRIVE16 field descriptions (continued)

Field	Description

9–8
BANK4_PIN02_
MA
	Pin 27, ENET0_RX_EN pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK4_PIN01_V	Pin 39, ENET0_MDIO pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK4_PIN01_
MA
	Pin 39, ENET0_MDIO pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK4_PIN00_V	Pin 54, ENET0_MDC pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK4_PIN00_
MA
	Pin 54, ENET0_MDC pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.33 PINCTRL Drive Strength and Voltage Register 17 (HW_PINCTRL_DRIVE17)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 8 pins of bank 4.

HW_PINCTRL_DRIVE17: 0x410

HW_PINCTRL_DRIVE17_SET: 0x414

HW_PINCTRL_DRIVE17_CLR: 0x418

HW_PINCTRL_DRIVE17_TOG: 0x41C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE17 – 8001_8000h base + 410h offset = 8001_8410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD7	BANK4_PIN15_V	BANK4_PIN15_MA		RSRVD6	BANK4_PIN14_V	BANK4_PIN14_MA		RSRVD5	BANK4_PIN13_V	BANK4_PIN13_MA		RSRVD4	BANK4_PIN12_V	BANK4_PIN12_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	BANK4_PIN11_V	BANK4_PIN11_MA		RSRVD2	BANK4_PIN10_V	BANK4_PIN10_MA		RSRVD1	BANK4_PIN09_V	BANK4_PIN09_MA		RSRVD0	BANK4_PIN08_V	BANK4_PIN08_MA	
W	▨				▨				▨				▨			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

HW_PINCTRL_DRIVE17 field descriptions

Field	Description
31 RSRVD7	Always write zeroes to this field.
30 BANK4_PIN15_V	Pin 61, ENET0_CRS pin voltage selection: 0= 1.8V; 1= 3.3V.

29–28
BANK4_PIN15_
MA
	Pin 61, ENET0_CRS pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

27 RSRVD6	Always write zeroes to this field.
26 BANK4_PIN14_V	Pin 57, ENET0_COL pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE17 field descriptions (continued)

Field	Description
	1= 3.3V.

25–24
BANK4_PIN14_
MA
	Pin 57, ENET0_COL pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

23 RSRVD5	Always write zeroes to this field.
22 BANK4_PIN13_V	Pin 31, ENET0_RX_CLK pin voltage selection: 0= 1.8V; 1= 3.3V.

21–20
BANK4_PIN13_
MA
	Pin 31, ENET0_RX_CLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

19 RSRVD4	Always write zeroes to this field.
18 BANK4_PIN12_V	Pin 41, ENET0_TXD3 pin voltage selection: 0= 1.8V; 1= 3.3V.

17–16
BANK4_PIN12_
MA
	Pin 41, ENET0_TXD3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15 RSRVD3	Always write zeroes to this field.
14 BANK4_PIN11_V	Pin 43, ENET0_TXD2 pin voltage selection: 0= 1.8V; 1= 3.3V.

13–12
BANK4_PIN11_
MA
	Pin 43, ENET0_TXD2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;

HW_PINCTRL_DRIVE17 field descriptions (continued)

Field	Description
	11= Reserved;
11 RSRVD2	Always write zeroes to this field.
10 BANK4_PIN10_V	Pin 53, ENET0_RXD3 pin voltage selection: 0= 1.8V; 1= 3.3V.

9–8
BANK4_PIN10_
MA
	Pin 53, ENET0_RXD3 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

7 RSRVD1	Always write zeroes to this field.
6 BANK4_PIN09_V	Pin 51, ENET0_RXD2 pin voltage selection: 0= 1.8V; 1= 3.3V.

5–4
BANK4_PIN09_
MA
	Pin 51, ENET0_RXD2 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

3 RSRVD0	Always write zeroes to this field.
2 BANK4_PIN08_V	Pin 35, ENET0_TXD1 pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0
BANK4_PIN08_
MA
	Pin 35, ENET0_TXD1 pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.34 PINCTRL Drive Strength and Voltage Register 18 (HW_PINCTRL_DRIVE18)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 2 pins of bank 4.

HW_PINCTRL_DRIVE18: 0x420

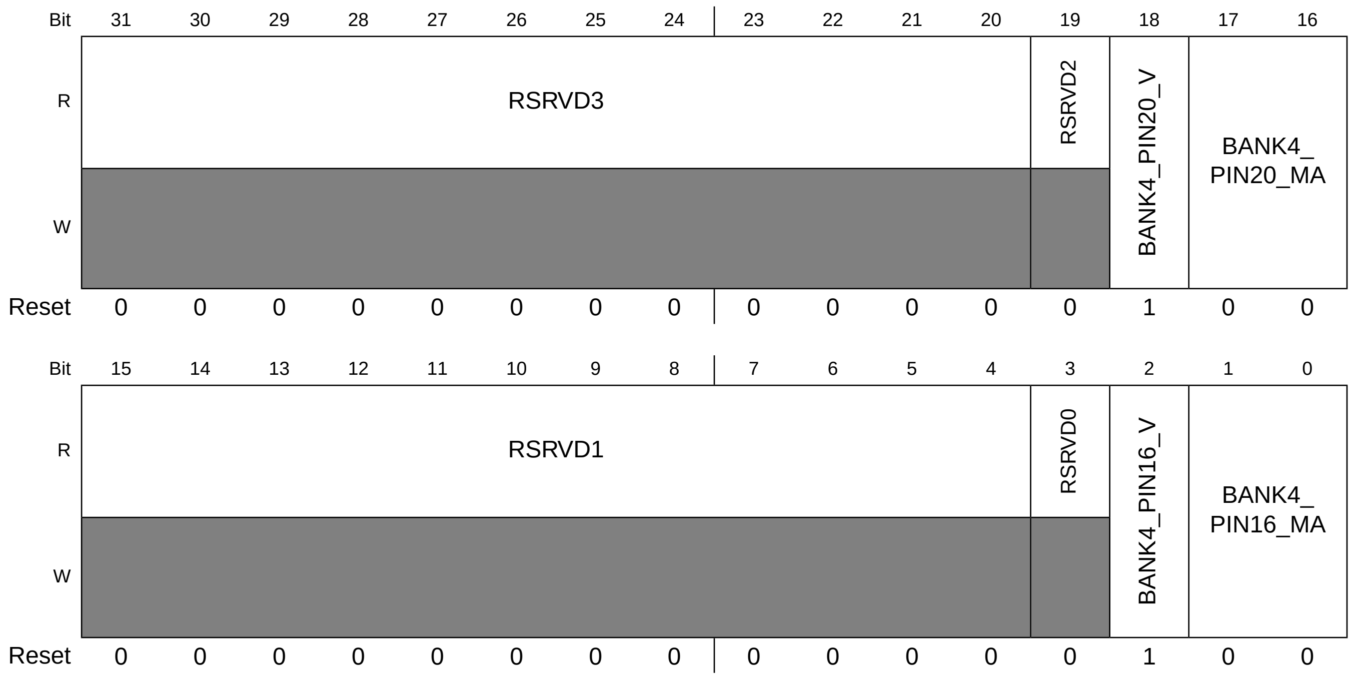
HW_PINCTRL_DRIVE18_SET: 0x424

HW_PINCTRL_DRIVE18_CLR: 0x428

HW_PINCTRL_DRIVE18_TOG: 0x42C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE18 – 8001_8000h base + 420h offset = 8001_8420h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD3
RSRVD2
BANK4_PIN20_V
BANK4_ PIN20_MA
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1
RSRVD0
BANK4_PIN16_V
BANK4_ PIN16_MA
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

HW_PINCTRL_DRIVE18 field descriptions

Field	Description
31–20 RSRVD3	Always write zeroes to this field.
19 RSRVD2	Always write zeroes to this field.
18 BANK4_PIN20_V	Pin 230, JTAG_RTCK pin voltage selection: 0= 1.8V;

HW_PINCTRL_DRIVE18 field descriptions (continued)

Field	Description
	1= 3.3V.

17–16 BANK4_PIN20_MA	Pin 230, JTAG_RTCK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

15–4 RSRVD1	Always write zeroes to this field.
3 RSRVD0	Always write zeroes to this field.
2 BANK4_PIN16_V	Pin 19, ENET_CLK pin voltage selection: 0= 1.8V; 1= 3.3V.

1–0 BANK4_PIN16_MA	Pin 19, ENET_CLK pin output drive strength selection:
	00= 4 mA;
	01= 8 mA;
	10= 12 mA;
	11= Reserved;

9.4.35 PINCTRL Drive Strength and Voltage Register 19 (HW_PINCTRL_DRIVE19)

The PINCTRL Drive Strength and Voltage Register selects the current drive strength for 0 pins of bank 4.

HW_PINCTRL_DRIVE19: 0x430

HW_PINCTRL_DRIVE19_SET: 0x434

HW_PINCTRL_DRIVE19_CLR: 0x438

HW_PINCTRL_DRIVE19_TOG: 0x43C

The Drive Strength and Voltage Register selects the drive strength and voltage for pins that are configured for output.

Address: HW_PINCTRL_DRIVE19 – 8001_8000h base + 430h offset = 8001_8430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																															
W	▨																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DRIVE19 field descriptions

Field	Description
31–0 RSRVD0	Always write zeroes to this field.

9.4.36 PINCTRL Bank 0 Pull Up Resistor Enable Register (HW_PINCTRL_PULL0)

The PINCTRL Bank 0 PULL Register enables/disables the internal pull up resistors for those pins in bank 0 which support this operation.

HW_PINCTRL_PULL0: 0x600

HW_PINCTRL_PULL0_SET: 0x604

HW_PINCTRL_PULL0_CLR: 0x608

HW_PINCTRL_PULL0_TOG: 0x60C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL0 – 8001_8000h base + 600h offset = 8001_8600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1			BANK0_PIN28	BANK0_PIN27	BANK0_PIN26	BANK0_PIN25	BANK0_PIN24	BANK0_PIN23	BANK0_PIN22	BANK0_PIN21	BANK0_PIN20	BANK0_PIN19	BANK0_PIN18	BANK0_PIN17	BANK0_PIN16
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0								BANK0_PIN07	BANK0_PIN06	BANK0_PIN05	BANK0_PIN04	BANK0_PIN03	BANK0_PIN02	BANK0_PIN01	BANK0_PIN00
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL0 field descriptions

Field	Description
31–29 RSRVD1	Always write zeroes to this field.
28 BANK0_PIN28	Set this bit to one to enable the internal pullup on pin 129, GPMI_RESETN.
27 BANK0_PIN27	Set this bit to one to enable the internal pullup on pin 123, GPMI_CLE.
26 BANK0_PIN26	Set this bit to one to enable the internal pullup on pin 117, GPMI_ALE.
25 BANK0_PIN25	Set this bit to one to disable the internal keeper on pin 127, GPMI_WRN.
24 BANK0_PIN24	Set this bit to one to enable the internal pullup on pin 110, GPMI_RDN.
23 BANK0_PIN23	Set this bit to one to enable the internal pullup on pin 80, GPMI_RDY3.
22 BANK0_PIN22	Set this bit to one to enable the internal pullup on pin 88, GPMI_RDY2.
21 BANK0_PIN21	Set this bit to one to enable the internal pullup on pin 119, GPMI_RDY1.
20 BANK0_PIN20	Set this bit to one to enable the internal pullup on pin 103, GPMI_RDY0.
19 BANK0_PIN19	Set this bit to one to enable the internal pullup on pin 135, GPMI_CE3N.
18 BANK0_PIN18	Set this bit to one to enable the internal pullup on pin 92, GPMI_CE2N.
17 BANK0_PIN17	Set this bit to one to enable the internal pullup on pin 131, GPMI_CE1N.
16 BANK0_PIN16	Set this bit to one to enable the internal pullup on pin 115, GPMI_CE0N.
15–8 RSRVD0	Always write zeroes to this field.
7 BANK0_PIN07	Set this bit to one to enable the internal pullup on pin 124, GPMI_D07.
6 BANK0_PIN06	Set this bit to one to enable the internal pullup on pin 130, GPMI_D06.
5 BANK0_PIN05	Set this bit to one to enable the internal pullup on pin 116, GPMI_D05.
4 BANK0_PIN04	Set this bit to one to enable the internal pullup on pin 128, GPMI_D04.
3 BANK0_PIN03	Set this bit to one to enable the internal pullup on pin 132, GPMI_D03.
2 BANK0_PIN02	Set this bit to one to enable the internal pullup on pin 126, GPMI_D02.

HW_PINCTRL_PULL0 field descriptions (continued)

Field	Description
1 BANK0_PIN01	Set this bit to one to enable the internal pullup on pin 136, GPMI_D01.
0 BANK0_PIN00	Set this bit to one to enable the internal pullup on pin 134, GPMI_D00.

9.4.37 PINCTRL Bank 1 Pull Up Resistor Enable Register (HW_PINCTRL_PULL1)

The PINCTRL Bank 1 PULL Register enables/disables the internal pull up resistors for those pins in bank 1 which support this operation.

HW_PINCTRL_PULL1: 0x610

HW_PINCTRL_PULL1_SET: 0x614

HW_PINCTRL_PULL1_CLR: 0x618

HW_PINCTRL_PULL1_TOG: 0x61C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL1 – 8001_8000h base + 610h offset = 8001_8610h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BANK1_PIN31	BANK1_PIN30	BANK1_PIN29	BANK1_PIN28	BANK1_PIN27	BANK1_PIN26	BANK1_PIN25	BANK1_PIN24	BANK1_PIN23	BANK1_PIN22	BANK1_PIN21	BANK1_PIN20	BANK1_PIN19	BANK1_PIN18	BANK1_PIN17	BANK1_PIN16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BANK1_PIN15	BANK1_PIN14	BANK1_PIN13	BANK1_PIN12	BANK1_PIN11	BANK1_PIN10	BANK1_PIN09	BANK1_PIN08	BANK1_PIN07	BANK1_PIN06	BANK1_PIN05	BANK1_PIN04	BANK1_PIN03	BANK1_PIN02	BANK1_PIN01	BANK1_PIN00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL1 field descriptions

Field	Description
31 BANK1_PIN31	Set this bit to one to disable the internal keeper on pin 111, LCD_ENABLE.
30 BANK1_PIN30	Set this bit to one to disable the internal keeper on pin 73, LCD_DOTCLK.

HW_PINCTRL_PULL1 field descriptions (continued)

Field	Description
29 BANK1_PIN29	Set this bit to one to disable the internal keeper on pin 71, LCD_HSYNC.
28 BANK1_PIN28	Set this bit to one to disable the internal keeper on pin 59, LCD_VSYNC.
27 BANK1_PIN27	Set this bit to one to disable the internal keeper on pin 113, LCD_CS.
26 BANK1_PIN26	Set this bit to one to disable the internal keeper on pin 94, LCD_RS.
25 BANK1_PIN25	Set this bit to one to disable the internal keeper on pin 55, LCD_WR_RWN.
24 BANK1_PIN24	Set this bit to one to disable the internal keeper on pin 96, LCD_RD_E.
23 BANK1_PIN23	Set this bit to one to disable the internal keeper on pin 108, LCD_D23.
22 BANK1_PIN22	Set this bit to one to disable the internal keeper on pin 120, LCD_D22.
21 BANK1_PIN21	Set this bit to one to disable the internal keeper on pin 122, LCD_D21.
20 BANK1_PIN20	Set this bit to one to disable the internal keeper on pin 107, LCD_D20.
19 BANK1_PIN19	Set this bit to one to disable the internal keeper on pin 112, LCD_D19.
18 BANK1_PIN18	Set this bit to one to disable the internal keeper on pin 118, LCD_D18.
17 BANK1_PIN17	Set this bit to one to disable the internal keeper on pin 105, LCD_D17.
16 BANK1_PIN16	Set this bit to one to disable the internal keeper on pin 104, LCD_D16.
15 BANK1_PIN15	Set this bit to one to disable the internal keeper on pin 114, LCD_D15.
14 BANK1_PIN14	Set this bit to one to disable the internal keeper on pin 106, LCD_D14.
13 BANK1_PIN13	Set this bit to one to disable the internal keeper on pin 102, LCD_D13.
12 BANK1_PIN12	Set this bit to one to disable the internal keeper on pin 87, LCD_D12.
11 BANK1_PIN11	Set this bit to one to disable the internal keeper on pin 95, LCD_D11.
10 BANK1_PIN10	Set this bit to one to disable the internal keeper on pin 85, LCD_D10.
9 BANK1_PIN09	Set this bit to one to disable the internal keeper on pin 99, LCD_D09.

HW_PINCTRL_PULL1 field descriptions (continued)

Field	Description
8 BANK1_PIN08	Set this bit to one to disable the internal keeper on pin 93, LCD_D08.
7 BANK1_PIN07	Set this bit to one to disable the internal keeper on pin 75, LCD_D07.
6 BANK1_PIN06	Set this bit to one to disable the internal keeper on pin 91, LCD_D06.
5 BANK1_PIN05	Set this bit to one to disable the internal keeper on pin 89, LCD_D05.
4 BANK1_PIN04	Set this bit to one to disable the internal keeper on pin 79, LCD_D04.
3 BANK1_PIN03	Set this bit to one to disable the internal keeper on pin 77, LCD_D03.
2 BANK1_PIN02	Set this bit to one to disable the internal keeper on pin 67, LCD_D02.
1 BANK1_PIN01	Set this bit to one to disable the internal keeper on pin 69, LCD_D01.
0 BANK1_PIN00	Set this bit to one to disable the internal keeper on pin 63, LCD_D00.

9.4.38 PINCTRL Bank 2 Pull Up Resistor Enable Register (HW_PINCTRL_PULL2)

The PINCTRL Bank 2 PULL Register enables/disables the internal pull up resistors for those pins in bank 2 which support this operation.

HW_PINCTRL_PULL2: 0x620

HW_PINCTRL_PULL2_SET: 0x624

HW_PINCTRL_PULL2_CLR: 0x628

HW_PINCTRL_PULL2_TOG: 0x62C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL2 – 8001_8000h base + 620h offset = 8001_8620h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2				BANK2_PIN27	BANK2_PIN26	BANK2_PIN25	BANK2_PIN24	RSRVD1		BANK2_PIN21	BANK2_PIN20	BANK2_PIN19	BANK2_PIN18	BANK2_PIN17	BANK2_PIN16
W	▨	▨	▨	▨					▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK2_PIN15	BANK2_PIN14	BANK2_PIN13	BANK2_PIN12	RSRVD0	BANK2_PIN10	BANK2_PIN09	BANK2_PIN08	BANK2_PIN07	BANK2_PIN06	BANK2_PIN05	BANK2_PIN04	BANK2_PIN03	BANK2_PIN02	BANK2_PIN01	BANK2_PIN00
W					▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL2 field descriptions

Field	Description
31–28 RSRVD2	Always write zeroes to this field.
27 BANK2_PIN27	Set this bit to one to enable the internal pullup on pin 15, SSP3_SS0.
26 BANK2_PIN26	Set this bit to one to enable the internal pullup on pin 3, SSP3_MISO.
25 BANK2_PIN25	Set this bit to one to enable the internal pullup on pin 9, SSP3_MOSI.
24 BANK2_PIN24	Set this bit to one to disable the internal keeper on pin 286, SSP3_SCK.
23–22 RSRVD1	Always write zeroes to this field.
21 BANK2_PIN21	Set this bit to one to enable the internal pullup on pin 18, SSP2_SS2.
20 BANK2_PIN20	Set this bit to one to enable the internal pullup on pin 7, SSP2_SS1.
19 BANK2_PIN19	Set this bit to one to enable the internal pullup on pin 4, SSP2_SS0.
18 BANK2_PIN18	Set this bit to one to enable the internal pullup on pin 288, SSP2_MISO.
17 BANK2_PIN17	Set this bit to one to enable the internal pullup on pin 1, SSP2_MOSI.
16 BANK2_PIN16	Set this bit to one to disable the internal keeper on pin 280, SSP2_SCK.

HW_PINCTRL_PULL2 field descriptions (continued)

Field	Description
15 BANK2_PIN15	Set this bit to one to enable the internal pullup on pin 23, SSP1_DATA3.
14 BANK2_PIN14	Set this bit to one to enable the internal pullup on pin 21, SSP1_DATA0.
13 BANK2_PIN13	Set this bit to one to enable the internal pullup on pin 17, SSP1_CMD.
12 BANK2_PIN12	Set this bit to one to enable the internal pullup on pin 11, SSP1_SCK.
11 RSRVD0	Always write zeroes to this field.
10 BANK2_PIN10	Set this bit to one to disable the internal keeper on pin 268, SSP0_SCK.
9 BANK2_PIN09	Set this bit to one to enable the internal pullup on pin 275, SSP0_DETECT.
8 BANK2_PIN08	Set this bit to one to enable the internal pullup on pin 276, SSP0_CMD.
7 BANK2_PIN07	Set this bit to one to enable the internal pullup on pin 282, SSP0_DATA7.
6 BANK2_PIN06	Set this bit to one to enable the internal pullup on pin 6, SSP0_DATA6.
5 BANK2_PIN05	Set this bit to one to enable the internal pullup on pin 284, SSP0_DATA5.
4 BANK2_PIN04	Set this bit to one to enable the internal pullup on pin 278, SSP0_DATA4.
3 BANK2_PIN03	Set this bit to one to enable the internal pullup on pin 274, SSP0_DATA3.
2 BANK2_PIN02	Set this bit to one to enable the internal pullup on pin 2, SSP0_DATA2.
1 BANK2_PIN01	Set this bit to one to enable the internal pullup on pin 289, SSP0_DATA1.
0 BANK2_PIN00	Set this bit to one to enable the internal pullup on pin 270, SSP0_DATA0.

9.4.39 PINCTRL Bank 3 Pull Up Resistor Enable Register (HW_PINCTRL_PULL3)

The PINCTRL Bank 3 PULL Register enables/disables the internal pull up resistors for those pins in bank 3 which support this operation.

HW_PINCTRL_PULL3: 0x630

HW_PINCTRL_PULL3_SET: 0x634

HW_PINCTRL_PULL3_CLR: 0x638

HW_PINCTRL_PULL3_TOG: 0x63C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL3 – 8001_8000h base + 630h offset = 8001_8630h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1	BANK3_PIN30	BANK3_PIN29	BANK3_PIN28	BANK3_PIN27	BANK3_PIN26	BANK3_PIN25	BANK3_PIN24	BANK3_PIN23	BANK3_PIN22	BANK3_PIN21	BANK3_PIN20	RSRVD0	BANK3_PIN18	BANK3_PIN17	BANK3_PIN16
W	▨												▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK3_PIN15	BANK3_PIN14	BANK3_PIN13	BANK3_PIN12	BANK3_PIN11	BANK3_PIN10	BANK3_PIN09	BANK3_PIN08	BANK3_PIN07	BANK3_PIN06	BANK3_PIN05	BANK3_PIN04	BANK3_PIN03	BANK3_PIN02	BANK3_PIN01	BANK3_PIN00
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL3 field descriptions

Field	Description
31 RSRVD1	Always write zeroes to this field.
30 BANK3_PIN30	Set this bit to one to disable the internal keeper on pin 101, LCD_RESET.
29 BANK3_PIN29	Set this bit to one to disable the internal keeper on pin 279, PWM4.
28 BANK3_PIN28	Set this bit to one to disable the internal keeper on pin 287, PWM3.
27 BANK3_PIN27	Set this bit to one to disable the internal keeper on pin 285, SPDIF.
26 BANK3_PIN26	Set this bit to one to disable the internal keeper on pin 8, SAIF1_SDATA0.
25 BANK3_PIN25	Set this bit to one to disable the internal keeper on pin 281, I2C0_SDA.
24 BANK3_PIN24	Set this bit to one to disable the internal keeper on pin 272, I2C0_SCL.

HW_PINCTRL_PULL3 field descriptions (continued)

Field	Description
23 BANK3_PIN23	Set this bit to one to disable the internal keeper on pin 12, SAIF0_SDATA0.
22 BANK3_PIN22	Set this bit to one to disable the internal keeper on pin 16, SAIF0_BITCLK.
21 BANK3_PIN21	Set this bit to one to disable the internal keeper on pin 34, SAIF0_LRCLK.
20 BANK3_PIN20	Set this bit to one to disable the internal keeper on pin 28, SAIF0_MCLK.
19 RSRVD0	Always write zeroes to this field.
18 BANK3_PIN18	Set this bit to one to enable the internal pullup on pin 68, PWM2.
17 BANK3_PIN17	Set this bit to one to disable the internal keeper on pin 84, PWM1.
16 BANK3_PIN16	Set this bit to one to disable the internal keeper on pin 72, PWM0.
15 BANK3_PIN15	Set this bit to one to disable the internal keeper on pin 82, AUART3_RTS.
14 BANK3_PIN14	Set this bit to one to disable the internal keeper on pin 90, AUART3_CTS.
13 BANK3_PIN13	Set this bit to one to disable the internal keeper on pin 86, AUART3_TX.
12 BANK3_PIN12	Set this bit to one to disable the internal keeper on pin 98, AUART3_RX.
11 BANK3_PIN11	Set this bit to one to disable the internal keeper on pin 56, AUART2_RTS.
10 BANK3_PIN10	Set this bit to one to disable the internal keeper on pin 50, AUART2_CTS.
9 BANK3_PIN09	Set this bit to one to enable the internal pullup on pin 26, AUART2_TX.
8 BANK3_PIN08	Set this bit to one to enable the internal pullup on pin 22, AUART2_RX.
7 BANK3_PIN07	Set this bit to one to enable the internal pullup on pin 74, AUART1_RTS.
6 BANK3_PIN06	Set this bit to one to enable the internal pullup on pin 78, AUART1_CTS.
5 BANK3_PIN05	Set this bit to one to disable the internal keeper on pin 65, AUART1_TX.
4 BANK3_PIN04	Set this bit to one to disable the internal keeper on pin 81, AUART1_RX.
3 BANK3_PIN03	Set this bit to one to disable the internal keeper on pin 66, AUART0_RTS.

HW_PINCTRL_PULL3 field descriptions (continued)

Field	Description
2 BANK3_PIN02	Set this bit to one to disable the internal keeper on pin 70, AUART0_CTS.
1 BANK3_PIN01	Set this bit to one to disable the internal keeper on pin 38, AUART0_TX.
0 BANK3_PIN00	Set this bit to one to disable the internal keeper on pin 30, AUART0_RX.

9.4.40 PINCTRL Bank 4 Pull Up Resistor Enable Register (HW_PINCTRL_PULL4)

The PINCTRL Bank 4 PULL Register enables/disables the internal pull up resistors for those pins in bank 4 which support this operation.

HW_PINCTRL_PULL4: 0x640

HW_PINCTRL_PULL4_SET: 0x644

HW_PINCTRL_PULL4_CLR: 0x648

HW_PINCTRL_PULL4_TOG: 0x64C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL4 – 8001_8000h base + 640h offset = 8001_8640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1											BANK4_PIN20	RSRVD0			BANK4_PIN16
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	BANK4_PIN15	BANK4_PIN14	BANK4_PIN13	BANK4_PIN12	BANK4_PIN11	BANK4_PIN10	BANK4_PIN09	BANK4_PIN08	BANK4_PIN07	BANK4_PIN06	BANK4_PIN05	BANK4_PIN04	BANK4_PIN03	BANK4_PIN02	BANK4_PIN01	BANK4_PIN00
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL4 field descriptions

Field	Description
31–21 RSRVD1	Always write zeroes to this field.

HW_PINCTRL_PULL4 field descriptions (continued)

Field	Description
20 BANK4_PIN20	Set this bit to one to disable the internal keeper on pin 230, JTAG_RTCK.
19–17 RSRVD0	Always write zeroes to this field.
16 BANK4_PIN16	Set this bit to one to disable the internal keeper on pin 19, ENET_CLK.
15 BANK4_PIN15	Set this bit to one to disable the internal keeper on pin 61, ENET0_CRS.
14 BANK4_PIN14	Set this bit to one to disable the internal keeper on pin 57, ENET0_COL.
13 BANK4_PIN13	Set this bit to one to disable the internal keeper on pin 31, ENET0_RX_CLK.
12 BANK4_PIN12	Set this bit to one to disable the internal keeper on pin 41, ENET0_TXD3.
11 BANK4_PIN11	Set this bit to one to disable the internal keeper on pin 43, ENET0_TXD2.
10 BANK4_PIN10	Set this bit to one to disable the internal keeper on pin 53, ENET0_RXD3.
9 BANK4_PIN09	Set this bit to one to disable the internal keeper on pin 51, ENET0_RXD2.
8 BANK4_PIN08	Set this bit to one to enable the internal pullup on pin 35, ENET0_TXD1.
7 BANK4_PIN07	Set this bit to one to enable the internal pullup on pin 37, ENET0_TXD0.
6 BANK4_PIN06	Set this bit to one to enable the internal pullup on pin 29, ENET0_TX_EN.
5 BANK4_PIN05	Set this bit to one to disable the internal keeper on pin 13, ENET0_TX_CLK.
4 BANK4_PIN04	Set this bit to one to enable the internal pullup on pin 47, ENET0_RXD1.
3 BANK4_PIN03	Set this bit to one to enable the internal pullup on pin 45, ENET0_RXD0.
2 BANK4_PIN02	Set this bit to one to enable the internal pullup on pin 27, ENET0_RX_EN.
1 BANK4_PIN01	Set this bit to one to enable the internal pullup on pin 39, ENET0_MDIO.
0 BANK4_PIN00	Set this bit to one to enable the internal pullup on pin 54, ENET0_MDC.

9.4.41 PINCTRL Bank 5 Pad Keeper Disable Register (HW_PINCTRL_PULL5)

The PINCTRL Bank 5 PULL Register enables/disables the pad keepers for those pins in bank 5 which support this operation.

HW_PINCTRL_PULL5: 0x650

HW_PINCTRL_PULL5_SET: 0x654

HW_PINCTRL_PULL5_CLR: 0x658

HW_PINCTRL_PULL5_TOG: 0x65C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL5 – 8001_8000h base + 650h offset = 8001_8650h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1					BANK5_PIN26	RSRVD0		BANK5_PIN23	BANK5_PIN22	BANK5_PIN21	BANK5_PIN20	BANK5_PIN19	BANK5_PIN18	BANK5_PIN17	BANK5_PIN16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BANK5_PIN15	BANK5_PIN14	BANK5_PIN13	BANK5_PIN12	BANK5_PIN11	BANK5_PIN10	BANK5_PIN09	BANK5_PIN08	BANK5_PIN07	BANK5_PIN06	BANK5_PIN05	BANK5_PIN04	BANK5_PIN03	BANK5_PIN02	BANK5_PIN01	BANK5_PIN00
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL5 field descriptions

Field	Description
31–27 RSRVD1	Always write zeroes to this field.
26 BANK5_PIN26	Set this bit to one to disable the internal keeper on pin 196, EMI_DDR_OPEN.
25–24 RSRVD0	Always write zeroes to this field.
23 BANK5_PIN23	Set this bit to one to disable the internal keeper on pin 204, EMI_DQS1.
22 BANK5_PIN22	Set this bit to one to disable the internal keeper on pin 202, EMI_DQS0.
21 BANK5_PIN21	Set this bit to one to disable the internal keeper on pin 200, EMI_CLK.

HW_PINCTRL_PULL5 field descriptions (continued)

Field	Description
20 BANK5_PIN20	Set this bit to one to disable the internal keeper on pin 195, EMI_DDR_OPEN_FB.
19 BANK5_PIN19	Set this bit to one to disable the internal keeper on pin 222, EMI_DQM1.
18 BANK5_PIN18	Set this bit to one to disable the internal keeper on pin 171, EMI_ODT1.
17 BANK5_PIN17	Set this bit to one to disable the internal keeper on pin 183, EMI_DQM0.
16 BANK5_PIN16	Set this bit to one to disable the internal keeper on pin 173, EMI_ODT0.
15 BANK5_PIN15	Set this bit to one to disable the internal keeper on pin 218, EMI_D15.
14 BANK5_PIN14	Set this bit to one to disable the internal keeper on pin 223, EMI_D14.
13 BANK5_PIN13	Set this bit to one to disable the internal keeper on pin 208, EMI_D13.
12 BANK5_PIN12	Set this bit to one to disable the internal keeper on pin 213, EMI_D12.
11 BANK5_PIN11	Set this bit to one to disable the internal keeper on pin 207, EMI_D11.
10 BANK5_PIN10	Set this bit to one to disable the internal keeper on pin 217, EMI_D10.
9 BANK5_PIN09	Set this bit to one to disable the internal keeper on pin 212, EMI_D09.
8 BANK5_PIN08	Set this bit to one to disable the internal keeper on pin 216, EMI_D08.
7 BANK5_PIN07	Set this bit to one to disable the internal keeper on pin 193, EMI_D07.
6 BANK5_PIN06	Set this bit to one to disable the internal keeper on pin 189, EMI_D06.
5 BANK5_PIN05	Set this bit to one to disable the internal keeper on pin 182, EMI_D05.
4 BANK5_PIN04	Set this bit to one to disable the internal keeper on pin 180, EMI_D04.
3 BANK5_PIN03	Set this bit to one to disable the internal keeper on pin 184, EMI_D03.
2 BANK5_PIN02	Set this bit to one to disable the internal keeper on pin 177, EMI_D02.
1 BANK5_PIN01	Set this bit to one to disable the internal keeper on pin 188, EMI_D01.
0 BANK5_PIN00	Set this bit to one to disable the internal keeper on pin 185, EMI_D00.

9.4.42 PINCTRL Bank 6 Pad Keeper Disable Register (HW_PINCTRL_PULL6)

The PINCTRL Bank 6 PULL Register enables/disables the pad keepers for those pins in bank 6 which support this operation.

HW_PINCTRL_PULL6: 0x660

HW_PINCTRL_PULL6_SET: 0x664

HW_PINCTRL_PULL6_CLR: 0x668

HW_PINCTRL_PULL6_TOG: 0x66C

The Pull register enables/disables integrated on-chip pull up resistors or pad keepers for the pins.

Address: HW_PINCTRL_PULL6 – 8001_8000h base + 660h offset = 8001_8660h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1							BANK6_PIN24	BANK6_PIN23	BANK6_PIN22	BANK6_PIN21	BANK6_PIN20	BANK6_PIN19	BANK6_PIN18	BANK6_PIN17	BANK6_PIN16
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0	BANK6_PIN14	BANK6_PIN13	BANK6_PIN12	BANK6_PIN11	BANK6_PIN10	BANK6_PIN09	BANK6_PIN08	BANK6_PIN07	BANK6_PIN06	BANK6_PIN05	BANK6_PIN04	BANK6_PIN03	BANK6_PIN02	BANK6_PIN01	BANK6_PIN00
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PULL6 field descriptions

Field	Description
31–25 RSRVD1	Always write zeroes to this field.
24 BANK6_PIN24	Set this bit to one to disable the internal keeper on pin 166, EMI_CKE.
23 BANK6_PIN23	Set this bit to one to disable the internal keeper on pin 139, EMI_CE1N.

HW_PINCTRL_PULL6 field descriptions (continued)

Field	Description
22 BANK6_PIN22	Set this bit to one to disable the internal keeper on pin 151, EMI_CE0N.
21 BANK6_PIN21	Set this bit to one to disable the internal keeper on pin 176, EMI_WEN.
20 BANK6_PIN20	Set this bit to one to disable the internal keeper on pin 167, EMI_RASN.
19 BANK6_PIN19	Set this bit to one to disable the internal keeper on pin 172, EMI_CASN.
18 BANK6_PIN18	Set this bit to one to disable the internal keeper on pin 165, EMI_BA2.
17 BANK6_PIN17	Set this bit to one to disable the internal keeper on pin 160, EMI_BA1.
16 BANK6_PIN16	Set this bit to one to disable the internal keeper on pin 157, EMI_BA0.
15 RSRVD0	Always write zeroes to this field.
14 BANK6_PIN14	Set this bit to one to disable the internal keeper on pin 147, EMI_A14.
13 BANK6_PIN13	Set this bit to one to disable the internal keeper on pin 142, EMI_A13.
12 BANK6_PIN12	Set this bit to one to disable the internal keeper on pin 150, EMI_A12.
11 BANK6_PIN11	Set this bit to one to disable the internal keeper on pin 146, EMI_A11.
10 BANK6_PIN10	Set this bit to one to disable the internal keeper on pin 162, EMI_A10.
9 BANK6_PIN09	Set this bit to one to disable the internal keeper on pin 143, EMI_A09.
8 BANK6_PIN08	Set this bit to one to disable the internal keeper on pin 140, EMI_A08.
7 BANK6_PIN07	Set this bit to one to disable the internal keeper on pin 153, EMI_A07.
6 BANK6_PIN06	Set this bit to one to disable the internal keeper on pin 138, EMI_A06.
5 BANK6_PIN05	Set this bit to one to disable the internal keeper on pin 154, EMI_A05.
4 BANK6_PIN04	Set this bit to one to disable the internal keeper on pin 144, EMI_A04.
3 BANK6_PIN03	Set this bit to one to disable the internal keeper on pin 152, EMI_A03.
2 BANK6_PIN02	Set this bit to one to disable the internal keeper on pin 164, EMI_A02.

HW_PINCTRL_PULL6 field descriptions (continued)

Field	Description
1 BANK6_PIN01	Set this bit to one to disable the internal keeper on pin 158, EMI_A01.
0 BANK6_PIN00	Set this bit to one to disable the internal keeper on pin 170, EMI_A00.

9.4.43 PINCTRL Bank 0 Data Output Register (HW_PINCTRL_DOUT0)

The Bank 0 Data Output register provides data for all pins in bank 0 that are configured for GPIO output mode.

HW_PINCTRL_DOUT0: 0x700

HW_PINCTRL_DOUT0_SET: 0x704

HW_PINCTRL_DOUT0_CLR: 0x708

HW_PINCTRL_DOUT0_TOG: 0x70C

This register contains the data that will be driven out all bank 0 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

Address: HW_PINCTRL_DOUT0 – 8001_8000h base + 700h offset = 8001_8700h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			DOUT																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOUT0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 DOUT	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 29 pins in bank 0.

9.4.44 PINCTRL Bank 1 Data Output Register (HW_PINCTRL_DOUT1)

The Bank 1 Data Output register provides data for all pins in bank 1 that are configured for GPIO output mode.

HW_PINCTRL_DOUT1: 0x710

HW_PINCTRL_DOUT1_SET: 0x714

HW_PINCTRL_DOUT1_CLR: 0x718

HW_PINCTRL_DOUT1_TOG: 0x71C

This register contains the data that will be driven out all bank 0 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

Address: HW_PINCTRL_DOUT1 – 8001_8000h base + 710h offset = 8001_8710h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOUT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOUT1 field descriptions

Field	Description
31–0 DOUT	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 32 pins in bank 1.

9.4.45 PINCTRL Bank 2 Data Output Register (HW_PINCTRL_DOUT2)

The Bank 2 Data Output register provides data for all pins in bank 2 that are configured for GPIO output mode.

HW_PINCTRL_DOUT2: 0x720

HW_PINCTRL_DOUT2_SET: 0x724

HW_PINCTRL_DOUT2_CLR: 0x728

HW_PINCTRL_DOUT2_TOG: 0x72C

This register contains the data that will be driven out all bank 0 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

Address: HW_PINCTRL_DOUT2 – 8001_8000h base + 720h offset = 8001_8720h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				DOUT																											
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOUT2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 DOUT	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 28 pins in bank 2.

9.4.46 PINCTRL Bank 3 Data Output Register (HW_PINCTRL_DOUT3)

The Bank 3 Data Output register provides data for all pins in bank 3 that are configured for GPIO output mode.

HW_PINCTRL_DOUT3: 0x730

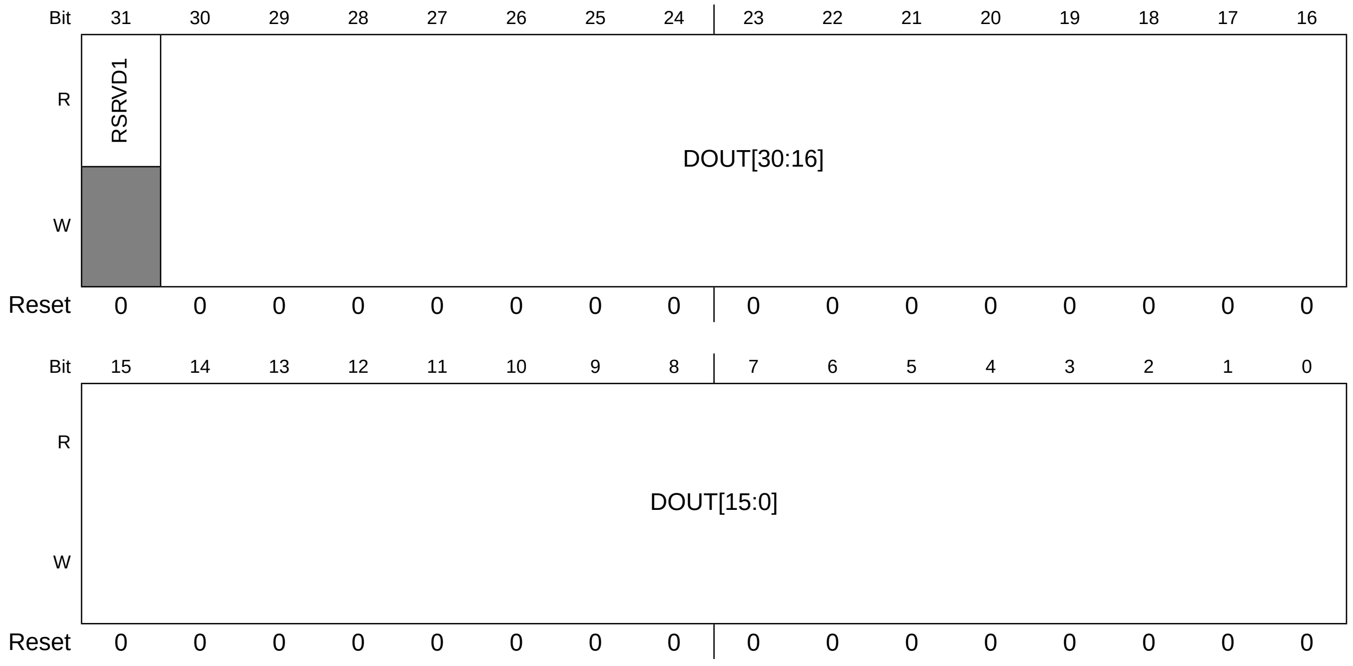
HW_PINCTRL_DOUT3_SET: 0x734

HW_PINCTRL_DOUT3_CLR: 0x738

HW_PINCTRL_DOUT3_TOG: 0x73C

This register contains the data that will be driven out all bank 0 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

Address: HW_PINCTRL_DOUT3 – 8001_8000h base + 730h offset = 8001_8730h



HW_PINCTRL_DOUT3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 DOUT	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 31 pins in bank 3.

9.4.47 PINCTRL Bank 4 Data Output Register (HW_PINCTRL_DOUT4)

The Bank 4 Data Output register provides data for all pins in bank 4 that are configured for GPIO output mode.

HW_PINCTRL_DOUT4: 0x740

HW_PINCTRL_DOUT4_SET: 0x744

HW_PINCTRL_DOUT4_CLR: 0x748

HW_PINCTRL_DOUT4_TOG: 0x74C

This register contains the data that will be driven out all bank 0 pins which are configured for GPIO output mode. For example, if HW_PINCTRL_MUXSEL0 contains 0x0000000F and HW_PINCTRL_DOE0 contains 0x00000001, then GPIO0[0] will be driven with the value from bit 0 of this register, GPIO0[1] will not be driven, and GPIO0[2:15] will be controlled by the associated primary interfaces.

Address: HW_PINCTRL_DOUT4 – 8001_8000h base + 740h offset = 8001_8740h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											DOUT																				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOUT4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 DOUT	This field selects the output value (0 or 1) for pins configured as GPIO outputs. Each bit in this register corresponds to one of the 21 pins in bank 4.

9.4.48 PINCTRL Bank 0 Data Input Register (HW_PINCTRL_DIN0)

The current value of all bank 0 pins may be read from the PINCTRL Bank 0 Data Input Register.

HW_PINCTRL_DIN0: 0x900

HW_PINCTRL_DIN0_SET: 0x904

HW_PINCTRL_DIN0_CLR: 0x908

HW_PINCTRL_DIN0_TOG: 0x90C

This register reflects the current values of all the bank 0 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers, but generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to insure that the chip is not driving the pin.

Address: HW_PINCTRL_DIN0 – 8001_8000h base + 900h offset = 8001_8900h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			DIN																												
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DIN0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 DIN	Each bit in this read-only register corresponds to one of the 29 pins in bank 0. The current state of each pin in bank 0, synchronized to HCLK, may be read here.

9.4.49 PINCTRL Bank 1 Data Input Register (HW_PINCTRL_DIN1)

The current value of all bank 1 pins may be read from the PINCTRL Bank 1 Data Input Register.

HW_PINCTRL_DIN1: 0x910

HW_PINCTRL_DIN1_SET: 0x914

HW_PINCTRL_DIN1_CLR: 0x918

HW_PINCTRL_DIN1_TOG: 0x91C

This register reflects the current values of all the bank 1 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers, but generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to insure that the chip is not driving the pin.

Address: HW_PINCTRL_DIN1 – 8001_8000h base + 910h offset = 8001_8910h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIN																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DIN1 field descriptions

Field	Description
31–0 DIN	Each bit in this read-only register corresponds to one of the 32 pins in bank 1. The current state of each pin in bank 1, synchronized to HCLK, may be read here.

9.4.50 PINCTRL Bank 2 Data Input Register (HW_PINCTRL_DIN2)

The current value of all bank 2 pins may be read from the PINCTRL Bank 2 Data Input Register.

HW_PINCTRL_DIN2: 0x920

HW_PINCTRL_DIN2_SET: 0x924

HW_PINCTRL_DIN2_CLR: 0x928

HW_PINCTRL_DIN2_TOG: 0x92C

This register reflects the current values of all the bank 2 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers, but generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to insure that the chip is not driving the pin.

Address: HW_PINCTRL_DIN2 – 8001_8000h base + 920h offset = 8001_8920h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				DIN																											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DIN2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 DIN	Each bit in this read-only register corresponds to one of the 28 pins in bank 2. The current state of each pin in bank 2, synchronized to HCLK, may be read here.

9.4.51 PINCTRL Bank 3 Data Input Register (HW_PINCTRL_DIN3)

The current value of all bank 3 pins may be read from the PINCTRL Bank 3 Data Input Register.

HW_PINCTRL_DIN3: 0x930

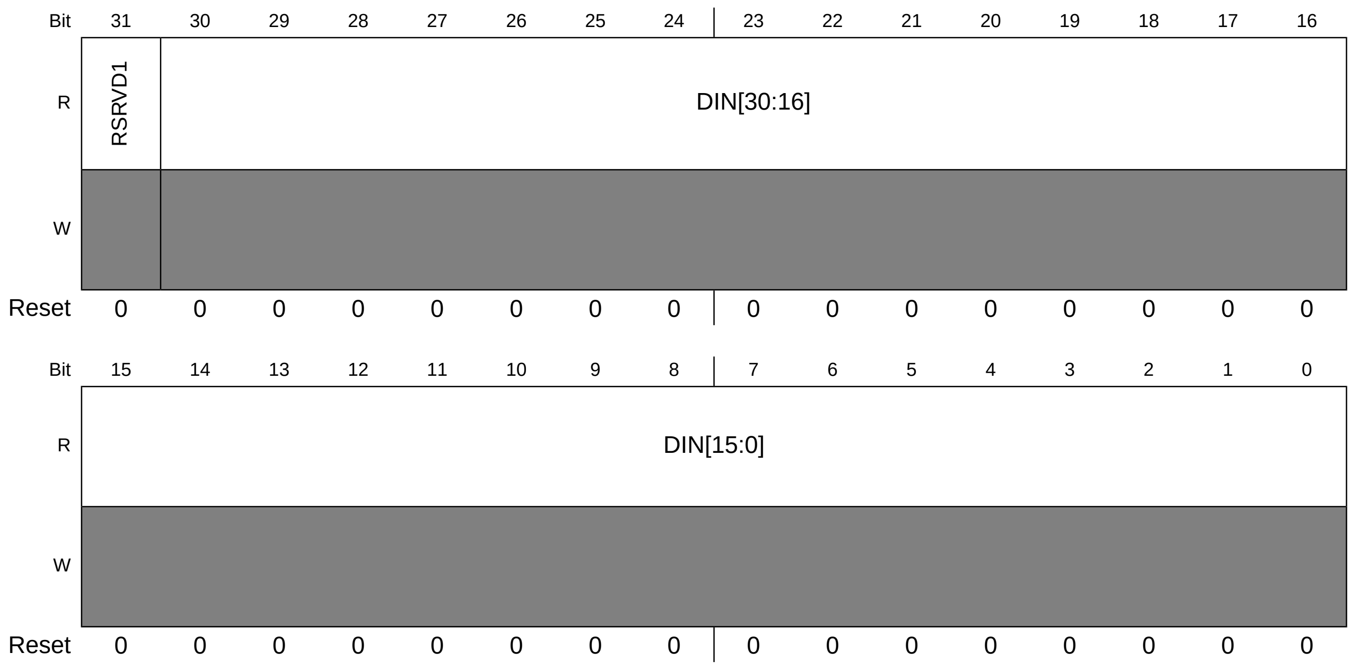
HW_PINCTRL_DIN3_SET: 0x934

HW_PINCTRL_DIN3_CLR: 0x938

HW_PINCTRL_DIN3_TOG: 0x93C

This register reflects the current values of all the bank 3 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers, but generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to insure that the chip is not driving the pin.

Address: HW_PINCTRL_DIN3 – 8001_8000h base + 930h offset = 8001_8930h



HW_PINCTRL_DIN3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 DIN	Each bit in this read-only register corresponds to one of the 31 pins in bank 3. The current state of each pin in bank 3, synchronized to HCLK, may be read here.

9.4.52 PINCTRL Bank 4 Data Input Register (HW_PINCTRL_DIN4)

The current value of all bank 4 pins may be read from the PINCTRL Bank 4 Data Input Register.

HW_PINCTRL_DIN4: 0x940

HW_PINCTRL_DIN4_SET: 0x944

HW_PINCTRL_DIN4_CLR: 0x948

HW_PINCTRL_DIN4_TOG: 0x94C

This register reflects the current values of all the bank 4 pins. The register accurately reflects the state of the pin regardless of the setting of the HW_PINCTRL_MUXSELx or HW_PINCTRL_DOEx registers, but generally, if it is desired to use a pin as a general purpose input, the pin's two bits in the HW_PINCTRL_MUXSELx register should be set to 3 (GPIO mode) and the pin's bit in the HW_PINCTRL_DOEx register should be set to 0 (disabled) to insure that the chip is not driving the pin.

Address: HW_PINCTRL_DIN4 – 8001_8000h base + 940h offset = 8001_8940h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											DIN																				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DIN4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 DIN	Each bit in this read-only register corresponds to one of the 21 pins in bank 4. The current state of each pin in bank 4, synchronized to HCLK, may be read here.

9.4.53 PINCTRL Bank 0 Data Output Enable Register (HW_PINCTRL_DOE0)

The PINCTRL Bank 0 Output Enable Register controls the output enable signal for all pins in bank 0 that are configured for GPIO mode.

HW_PINCTRL_DOE0: 0xb00

HW_PINCTRL_DOE0_SET: 0xb04

HW_PINCTRL_DOE0_CLR: 0xb08

HW_PINCTRL_DOE0_TOG: 0xb0C

For pins in bank 0 that are configured as GPIOs, a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUTxx register to be driven out the pin, and a 0 in this register will disable the corresponding driver.

Address: HW_PINCTRL_DOE0 – 8001_8000h base + B00h offset = 8001_8B00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			DOE																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOE0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 DOE	Each bit in this register corresponds to one of the 29 pins in bank 0. Setting a bit in this register to one allows the chip to drive the corresponding pin in GPIO mode.

9.4.54 PINCTRL Bank 1 Data Output Enable Register (HW_PINCTRL_DOE1)

The PINCTRL Bank 1 Output Enable Register controls the output enable signal for all pins in bank 1 that are configured for GPIO mode.

HW_PINCTRL_DOE1: 0xb10

HW_PINCTRL_DOE1_SET: 0xb14

HW_PINCTRL_DOE1_CLR: 0xb18

HW_PINCTRL_DOE1_TOG: 0xb1C

For pins in bank 1 that are configured as GPIOs, a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUTxx register to be driven out the pin, and a 0 in this register will disable the corresponding driver.

Address: HW_PINCTRL_DOE1 – 8001_8000h base + B10h offset = 8001_8B10h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DOE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOE1 field descriptions

Field	Description
31–0 DOE	Each bit in this register corresponds to one of the 32 pins in bank 1. Setting a bit in this register to one allows the chip to drive the corresponding pin in GPIO mode.

9.4.55 PINCTRL Bank 2 Data Output Enable Register (HW_PINCTRL_DOE2)

The PINCTRL Bank 2 Output Enable Register controls the output enable signal for all pins in bank 2 that are configured for GPIO mode.

HW_PINCTRL_DOE2: 0xb20

HW_PINCTRL_DOE2_SET: 0xb24

HW_PINCTRL_DOE2_CLR: 0xb28

HW_PINCTRL_DOE2_TOG: 0xb2C

For pins in bank 2 that are configured as GPIOs, a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUTxx register to be driven out the pin, and a 0 in this register will disable the corresponding driver.

Address: HW_PINCTRL_DOE2 – 8001_8000h base + B20h offset = 8001_8B20h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				DOE																											
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_DOE2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 DOE	Each bit in this register corresponds to one of the 28 pins in bank 2. Setting a bit in this register to one allows the chip to drive the corresponding pin in GPIO mode.

9.4.56 PINCTRL Bank 3 Data Output Enable Register (HW_PINCTRL_DOE3)

The PINCTRL Bank 3 Output Enable Register controls the output enable signal for all pins in bank 3 that are configured for GPIO mode.

HW_PINCTRL_DOE3: 0xb30

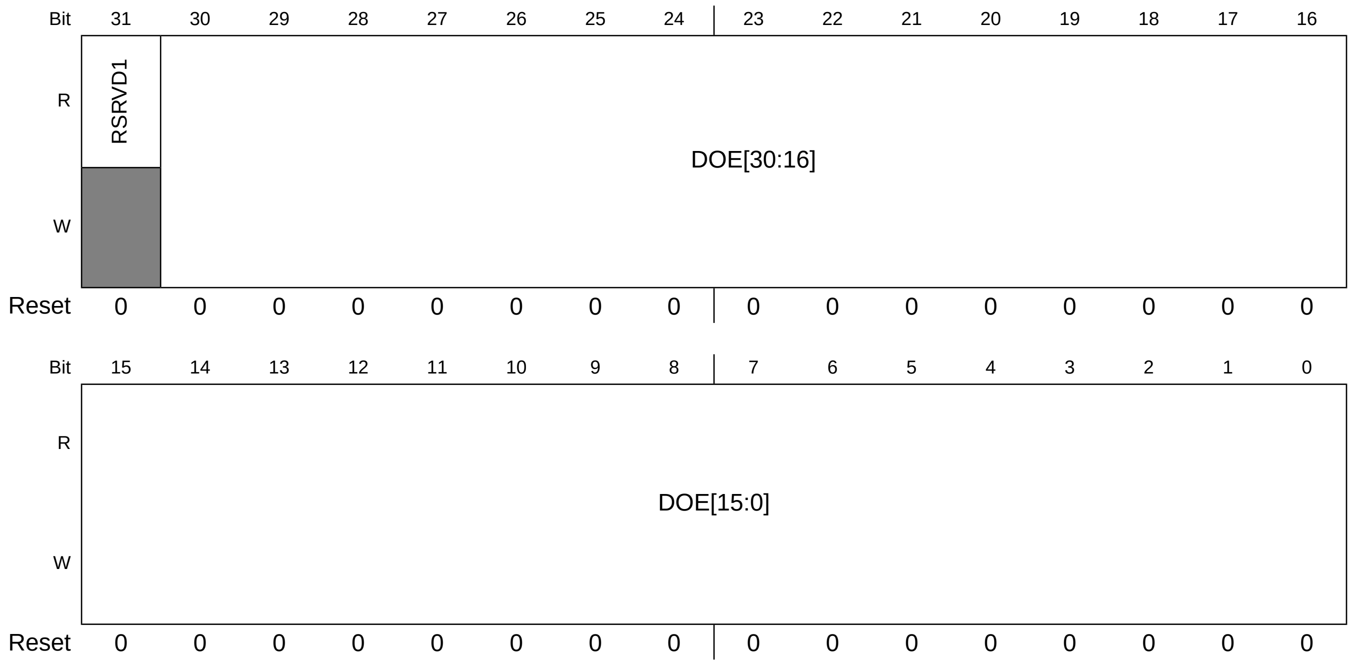
HW_PINCTRL_DOE3_SET: 0xb34

HW_PINCTRL_DOE3_CLR: 0xb38

HW_PINCTRL_DOE3_TOG: 0xb3C

For pins in bank 3 that are configured as GPIOs, a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUTxx register to be driven out the pin, and a 0 in this register will disable the corresponding driver.

Address: HW_PINCTRL_DOE3 – 8001_8000h base + B30h offset = 8001_8B30h



HW_PINCTRL_DOE3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 DOE	Each bit in this register corresponds to one of the 31 pins in bank 3. Setting a bit in this register to one allows the chip to drive the corresponding pin in GPIO mode.

9.4.57 PINCTRL Bank 4 Data Output Enable Register (HW_PINCTRL_DOE4)

The PINCTRL Bank 4 Output Enable Register controls the output enable signal for all pins in bank 4 that are configured for GPIO mode.

HW_PINCTRL_DOE4: 0xb40

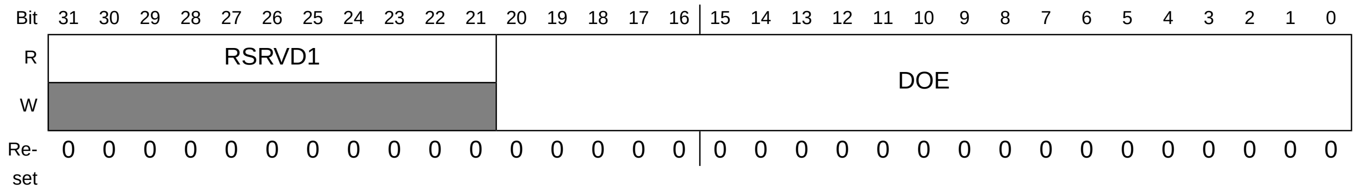
HW_PINCTRL_DOE4_SET: 0xb44

HW_PINCTRL_DOE4_CLR: 0xb48

HW_PINCTRL_DOE4_TOG: 0xb4C

For pins in bank 4 that are configured as GPIOs, a 1 in this register will enable the corresponding bit value from HW_PINCTRL_DOUTxx register to be driven out the pin, and a 0 in this register will disable the corresponding driver.

Address: HW_PINCTRL_DOE4 – 8001_8000h base + B40h offset = 8001_8B40h



HW_PINCTRL_DOE4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 DOE	Each bit in this register corresponds to one of the 21 pins in bank 4. Setting a bit in this register to one allows the chip to drive the corresponding pin in GPIO mode.

9.4.58 PINCTRL Bank 0 Interrupt Select Register (HW_PINCTRL_PIN2IRQ0)

The Bank 0 Interrupt Select register selects which of the bank 0 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ0: 0x1000

HW_PINCTRL_PIN2IRQ0_SET: 0x1004

HW_PINCTRL_PIN2IRQ0_CLR: 0x1008

HW_PINCTRL_PIN2IRQ0_TOG: 0x100C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 0 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL0 and HW_PINCTRL_IRQPOL0 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT0 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO0. For example, if this register contains 0x00000014, then pins GPIO0[2] and GPIO0[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT0 register.

Address: HW_PINCTRL_PIN2IRQ0 – 8001_8000h base + 1000h offset = 8001_9000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			PIN2IRQ																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PIN2IRQ0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 PIN2IRQ	Each bit in this register corresponds to one of the 29 pins in bank 0: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

9.4.59 PINCTRL Bank 1 Interrupt Select Register (HW_PINCTRL_PIN2IRQ1)

The Bank 1 Interrupt Select register selects which of the bank 1 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ1: 0x1010

HW_PINCTRL_PIN2IRQ1_SET: 0x1014

HW_PINCTRL_PIN2IRQ1_CLR: 0x1018

HW_PINCTRL_PIN2IRQ1_TOG: 0x101C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 1 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL1 and HW_PINCTRL_IRQPOL1 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT1 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO1. For example, if this register contains 0x00000014, then pins GPIO1[2] and GPIO1[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT1 register.

Address: HW_PINCTRL_PIN2IRQ1 – 8001_8000h base + 1010h offset = 8001_9010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PIN2IRQ																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PIN2IRQ1 field descriptions

Field	Description
31–0 PIN2IRQ	Each bit in this register corresponds to one of the 32 pins in bank 1: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

9.4.60 PINCTRL Bank 2 Interrupt Select Register (HW_PINCTRL_PIN2IRQ2)

The Bank 2 Interrupt Select register selects which of the bank 2 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ2: 0x1020

HW_PINCTRL_PIN2IRQ2_SET: 0x1024

HW_PINCTRL_PIN2IRQ2_CLR: 0x1028

HW_PINCTRL_PIN2IRQ2_TOG: 0x102C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 2 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL2 and HW_PINCTRL_IRQPOL2 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT2 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO2. For example, if this register contains 0x00000014, then pins GPIO2[2] and GPIO2[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT2 register.

Address: HW_PINCTRL_PIN2IRQ2 – 8001_8000h base + 1020h offset = 8001_9020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				PIN2IRQ																											
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PIN2IRQ2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 PIN2IRQ	Each bit in this register corresponds to one of the 28 pins in bank 2: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

9.4.61 PINCTRL Bank 3 Interrupt Select Register (HW_PINCTRL_PIN2IRQ3)

The Bank 3 Interrupt Select register selects which of the bank 3 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ3: 0x1030

HW_PINCTRL_PIN2IRQ3_SET: 0x1034

HW_PINCTRL_PIN2IRQ3_CLR: 0x1038

HW_PINCTRL_PIN2IRQ3_TOG: 0x103C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 3 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL3 and HW_PINCTRL_IRQPOL3 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT3 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO3. For example, if this register contains 0x00000014, then pins GPIO3[2] and GPIO3[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT3 register.

Address: HW_PINCTRL_PIN2IRQ3 – 8001_8000h base + 1030h offset = 8001_9030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1	PIN2IRQ[30:16]														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIN2IRQ[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PIN2IRQ3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 PIN2IRQ	Each bit in this register corresponds to one of the 31 pins in bank 3: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

9.4.62 PINCTRL Bank 4 Interrupt Select Register (HW_PINCTRL_PIN2IRQ4)

The Bank 4 Interrupt Select register selects which of the bank 4 pins may be used as interrupt sources.

HW_PINCTRL_PIN2IRQ4: 0x1040

HW_PINCTRL_PIN2IRQ4_SET: 0x1044

HW_PINCTRL_PIN2IRQ4_CLR: 0x1048

HW_PINCTRL_PIN2IRQ4_TOG: 0x104C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register selects which pins in bank 4 can be used to generate interrupts. If the pin is selected in this register by setting its bit to 1, then detection of the correct level or edge on the pin (as chosen by the HW_PINCTRL_IRQLEVEL4 and HW_PINCTRL_IRQPOL4 registers) will set the corresponding bit in the HW_PINCTRL_IRQSTAT4 register. If the pin is additionally enabled in the HW_PINCTRL_IRQEN0 register, then the interrupt will be propagated to the interrupt collector as interrupt GPIO4. For example, if this register contains 0x00000014, then pins GPIO4[2] and GPIO4[4] can be used as interrupt pins, and no other pins in bank 0 will cause bits to be set in the HW_PINCTRL_IRQSTAT4 register.

Address: HW_PINCTRL_PIN2IRQ4 – 8001_8000h base + 1040h offset = 8001_9040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											PIN2IRQ																				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_PIN2IRQ4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 PIN2IRQ	Each bit in this register corresponds to one of the 21 pins in bank 4: 0= Deselect the pin's interrupt functionality. 1= Select the pin to be used as an interrupt source.

9.4.63 PINCTRL Bank 0 Interrupt Mask Register (HW_PINCTRL_IRQEN0)

The PINCTRL Bank 0 Interrupt Mask Register contains interrupt enable masks for the pins in bank 0.

HW_PINCTRL_IRQEN0: 0x1100

HW_PINCTRL_IRQEN0_SET: 0x1104

HW_PINCTRL_IRQEN0_CLR: 0x1108

HW_PINCTRL_IRQEN0_TOG: 0x110C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 0. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT0, an interrupt will be propagated to the interrupt collector as interrupt GPIO0. For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT0 (corresponding to pins GPIO0[2] and GPIO0[4]) will cause interrupts from bank 0.

Address: HW_PINCTRL_IRQEN0 – 8001_8000h base + 1100h offset = 8001_9100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			IRQEN																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQEN0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 IRQEN	Each bit in this register corresponds to one of the 29 pins in bank 0: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT0.

9.4.64 PINCTRL Bank 1 Interrupt Mask Register (HW_PINCTRL_IRQEN1)

The PINCTRL Bank 1 Interrupt Mask Register contains interrupt enable masks for the pins in bank 1.

HW_PINCTRL_IRQEN1: 0x1110

HW_PINCTRL_IRQEN1_SET: 0x1114

HW_PINCTRL_IRQEN1_CLR: 0x1118

HW_PINCTRL_IRQEN1_TOG: 0x111C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 1. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT1, an interrupt will be propagated to the interrupt collector as interrupt GPIO1. For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT1 (corresponding to pins GPIO1[2] and GPIO1[4]) will cause interrupts from bank 1.

Address: HW_PINCTRL_IRQEN1 – 8001_8000h base + 1110h offset = 8001_9110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQEN																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQEN1 field descriptions

Field	Description
31–0 IRQEN	Each bit in this register corresponds to one of the 32 pins in bank 1: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT1.

9.4.65 PINCTRL Bank 2 Interrupt Mask Register (HW_PINCTRL_IRQEN2)

The PINCTRL Bank 2 Interrupt Mask Register contains interrupt enable masks for the pins in bank 2.

HW_PINCTRL_IRQEN2: 0x1120

HW_PINCTRL_IRQEN2_SET: 0x1124

HW_PINCTRL_IRQEN2_CLR: 0x1128

HW_PINCTRL_IRQEN2_TOG: 0x112C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 2. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT2, an interrupt will be propagated to the interrupt collector as interrupt GPIO2. For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT2 (corresponding to pins GPIO2[2] and GPIO2[4]) will cause interrupts from bank 2.

Address: HW_PINCTRL_IRQEN2 – 8001_8000h base + 1120h offset = 8001_9120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				IRQEN																											
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQEN2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 IRQEN	Each bit in this register corresponds to one of the 28 pins in bank 2: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT2. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT2.

9.4.66 PINCTRL Bank 3 Interrupt Mask Register (HW_PINCTRL_IRQEN3)

The PINCTRL Bank 3 Interrupt Mask Register contains interrupt enable masks for the pins in bank 3.

HW_PINCTRL_IRQEN3: 0x1130

HW_PINCTRL_IRQEN3_SET: 0x1134

HW_PINCTRL_IRQEN3_CLR: 0x1138

HW_PINCTRL_IRQEN3_TOG: 0x113C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 3. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT3, an interrupt will be propagated to the interrupt collector as interrupt GPIO3. For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT3 (corresponding to pins GPIO3[2] and GPIO3[4]) will cause interrupts from bank 3.

Address: HW_PINCTRL_IRQEN3 – 8001_8000h base + 1130h offset = 8001_9130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1	IRQEN[30:16]														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQEN[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQEN3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 IRQEN	Each bit in this register corresponds to one of the 31 pins in bank 3: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT3. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT3.

9.4.67 PINCTRL Bank 4 Interrupt Mask Register (HW_PINCTRL_IRQEN4)

The PINCTRL Bank 4 Interrupt Mask Register contains interrupt enable masks for the pins in bank 4.

HW_PINCTRL_IRQEN4: 0x1140

HW_PINCTRL_IRQEN4_SET: 0x1144

HW_PINCTRL_IRQEN4_CLR: 0x1148

HW_PINCTRL_IRQEN4_TOG: 0x114C

As described earlier in this chapter, any digital I/O pin can be used as an interrupt source. This register masks the interrupt sources from the pins in bank 4. If a bit is set in this register and the same bit is set in HW_PINCTRL_IRQSTAT4, an interrupt will be propagated to the interrupt collector as interrupt GPIO4. For example, if this register contains 0x00000014, then only bits 2 and 4 in HW_PINCTRL_IRQSTAT4 (corresponding to pins GPIO4[2] and GPIO4[4]) will cause interrupts from bank 4.

Address: HW_PINCTRL_IRQEN4 – 8001_8000h base + 1140h offset = 8001_9140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											IRQEN																				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQEN4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 IRQEN	Each bit in this register corresponds to one of the 21 pins in bank 4: 1= Enable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT4. 0= Disable interrupts from the corresponding bit in HW_PINCTRL_IRQSTAT4.

9.4.68 PINCTRL Bank 0 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL0)

The PINCTRL Bank 0 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 0.

HW_PINCTRL_IRQLEVEL0: 0x1200

HW_PINCTRL_IRQLEVEL0_SET: 0x1204

HW_PINCTRL_IRQLEVEL0_CLR: 0x1208

HW_PINCTRL_IRQLEVEL0_TOG: 0x120C

This register selects level or edge detection for interrupt generation. Each pin in bank 0 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL0 (see below) appropriately.

Address: HW_PINCTRL_IRQLEVEL0 – 8001_8000h base + 1200h offset = 8001_9200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			IRQLEVEL																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQLEVEL0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 IRQLEVEL	Each bit in this register corresponds to one of the 29 pins in bank 0: 1= Level detection; 0= Edge detection.

9.4.69 PINCTRL Bank 1 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL1)

The PINCTRL Bank 1 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 1.

HW_PINCTRL_IRQLEVEL1: 0x1210

HW_PINCTRL_IRQLEVEL1_SET: 0x1214

HW_PINCTRL_IRQLEVEL1_CLR: 0x1218

HW_PINCTRL_IRQLEVEL1_TOG: 0x121C

This register selects level or edge detection for interrupt generation. Each pin in bank 1 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL1 (see below) appropriately.

Address: HW_PINCTRL_IRQLEVEL1 – 8001_8000h base + 1210h offset = 8001_9210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IRQLEVEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQLEVEL1 field descriptions

Field	Description
31–0 IRQLEVEL	Each bit in this register corresponds to one of the 32 pins in bank 1: 1= Level detection; 0= Edge detection.

9.4.70 PINCTRL Bank 2 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL2)

The PINCTRL Bank 2 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 2.

HW_PINCTRL_IRQLEVEL2: 0x1220

HW_PINCTRL_IRQLEVEL2_SET: 0x1224

HW_PINCTRL_IRQLEVEL2_CLR: 0x1228

HW_PINCTRL_IRQLEVEL2_TOG: 0x122C

This register selects level or edge detection for interrupt generation. Each pin in bank 2 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL2 (see below) appropriately.

Address: HW_PINCTRL_IRQLEVEL2 – 8001_8000h base + 1220h offset = 8001_9220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				IRQLEVEL																											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQLEVEL2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 IRQLEVEL	Each bit in this register corresponds to one of the 28 pins in bank 2: 1= Level detection; 0= Edge detection.

9.4.71 PINCTRL Bank 3 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL3)

The PINCTRL Bank 3 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 3.

HW_PINCTRL_IRQLEVEL3: 0x1230

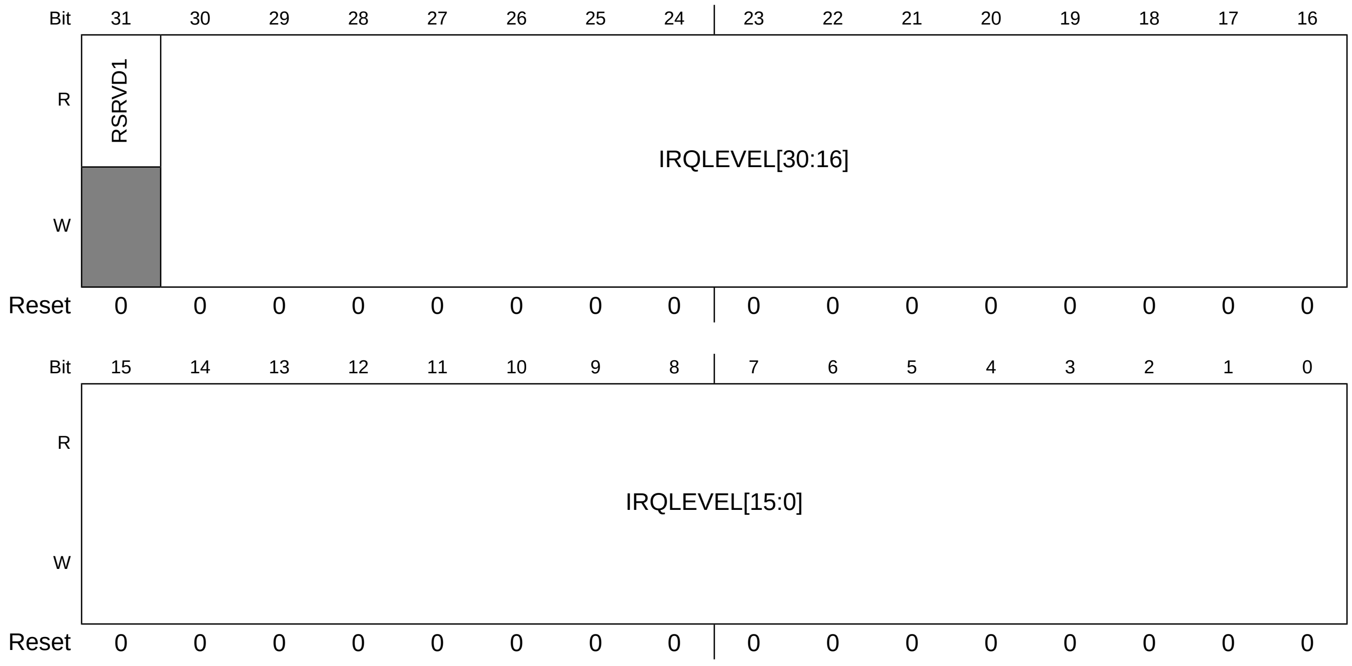
HW_PINCTRL_IRQLEVEL3_SET: 0x1234

HW_PINCTRL_IRQLEVEL3_CLR: 0x1238

HW_PINCTRL_IRQLEVEL3_TOG: 0x123C

This register selects level or edge detection for interrupt generation. Each pin in bank 3 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL3 (see below) appropriately.

Address: HW_PINCTRL_IRQLEVEL3 – 8001_8000h base + 1230h offset = 8001_9230h



HW_PINCTRL_IRQLEVEL3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 IRQLEVEL	Each bit in this register corresponds to one of the 31 pins in bank 3: 1= Level detection; 0= Edge detection.

9.4.72 PINCTRL Bank 4 Interrupt Level/Edge Register (HW_PINCTRL_IRQLEVEL4)

The PINCTRL Bank 4 Interrupt Level/Edge Register selects level or edge sensitivity for interrupt requests for the pins in bank 4.

HW_PINCTRL_IRQLEVEL4: 0x1240

HW_PINCTRL_IRQLEVEL4_SET: 0x1244

HW_PINCTRL_IRQLEVEL4_CLR: 0x1248

HW_PINCTRL_IRQLEVEL4_TOG: 0x124C

This register selects level or edge detection for interrupt generation. Each pin in bank 4 that is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting bits in this register and HW_PINCTRL_IRQPOL4 (see below) appropriately.

Address: HW_PINCTRL_IRQLEVEL4 – 8001_8000h base + 1240h offset = 8001_9240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											IRQLEVEL																				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQLEVEL4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 IRQLEVEL	Each bit in this register corresponds to one of the 21 pins in bank 4: 1= Level detection; 0= Edge detection.

9.4.73 PINCTRL Bank 0 Interrupt Polarity Register (HW_PINCTRL_IRQPOL0)

The PINCTRL Bank 0 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 0.

HW_PINCTRL_IRQPOL0: 0x1300

HW_PINCTRL_IRQPOL0_SET: 0x1304

HW_PINCTRL_IRQPOL0_CLR: 0x1308

HW_PINCTRL_IRQPOL0_TOG: 0x130C

This register selects the polarity for interrupt generation. Each pin in bank 0 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL0 (see above) appropriately.

Address: HW_PINCTRL_IRQPOL0 – 8001_8000h base + 1300h offset = 8001_9300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			IRQPOL																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQPOL0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 IRQPOL	Each bit in this register corresponds to one of the 29 pins in bank 0: 0= Low or falling edge; 1= High or rising edge.

9.4.74 PINCTRL Bank 1 Interrupt Polarity Register (HW_PINCTRL_IRQPOL1)

The PINCTRL Bank 1 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 1.

HW_PINCTRL_IRQPOL1: 0x1310

HW_PINCTRL_IRQPOL1_SET: 0x1314

HW_PINCTRL_IRQPOL1_CLR: 0x1318

HW_PINCTRL_IRQPOL1_TOG: 0x131C

This register selects the polarity for interrupt generation. Each pin in bank 1 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL1 (see above) appropriately.

Address: HW_PINCTRL_IRQPOL1 – 8001_8000h base + 1310h offset = 8001_9310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IRQPOL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQPOL1 field descriptions

Field	Description
31–0 IRQPOL	Each bit in this register corresponds to one of the 32 pins in bank 1: 0= Low or falling edge; 1= High or rising edge.

9.4.75 PINCTRL Bank 2 Interrupt Polarity Register (HW_PINCTRL_IRQPOL2)

The PINCTRL Bank 2 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 2.

HW_PINCTRL_IRQPOL2: 0x1320

HW_PINCTRL_IRQPOL2_SET: 0x1324

HW_PINCTRL_IRQPOL2_CLR: 0x1328

HW_PINCTRL_IRQPOL2_TOG: 0x132C

This register selects the polarity for interrupt generation. Each pin in bank 2 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL2 (see above) appropriately.

Address: HW_PINCTRL_IRQPOL2 – 8001_8000h base + 1320h offset = 8001_9320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				IRQPOL																											
W	▨	▨	▨	▨																												
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQPOL2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 IRQPOL	Each bit in this register corresponds to one of the 28 pins in bank 2: 0= Low or falling edge; 1= High or rising edge.

9.4.76 PINCTRL Bank 3 Interrupt Polarity Register (HW_PINCTRL_IRQPOL3)

The PINCTRL Bank 3 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 3.

HW_PINCTRL_IRQPOL3: 0x1330

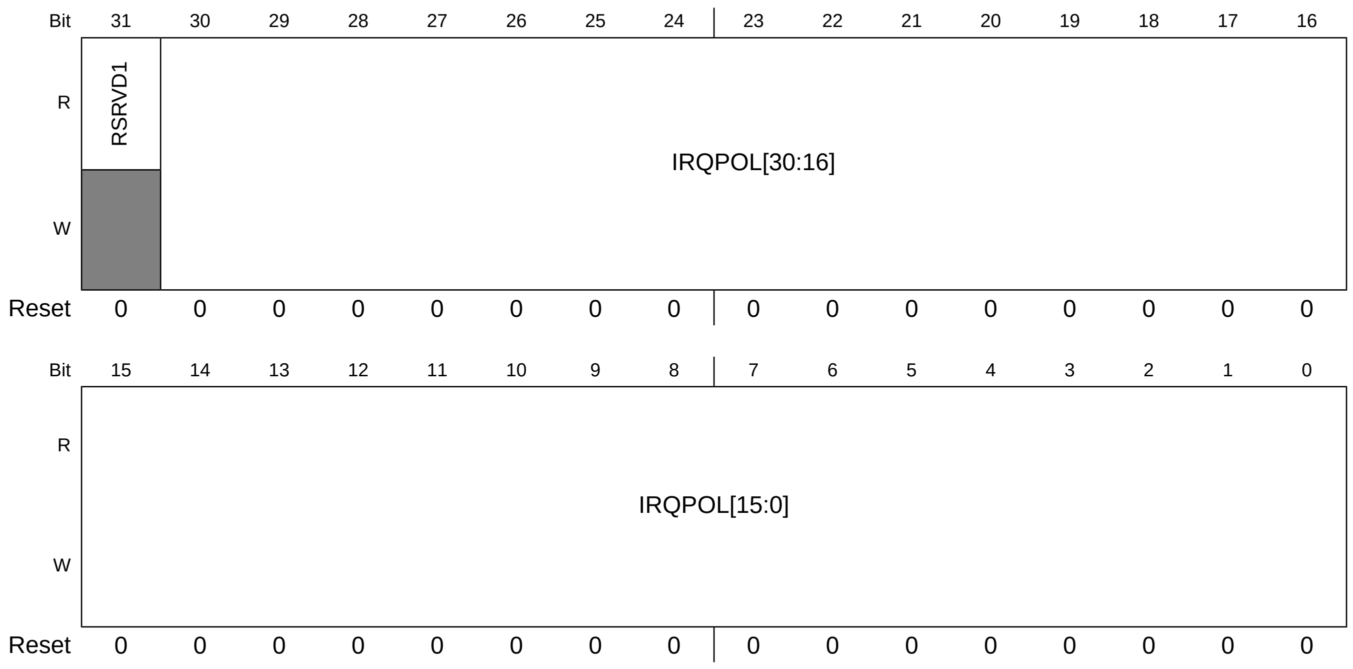
HW_PINCTRL_IRQPOL3_SET: 0x1334

HW_PINCTRL_IRQPOL3_CLR: 0x1338

HW_PINCTRL_IRQPOL3_TOG: 0x133C

This register selects the polarity for interrupt generation. Each pin in bank 3 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL3 (see above) appropriately.

Address: HW_PINCTRL_IRQPOL3 – 8001_8000h base + 1330h offset = 8001_9330h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD1
IRQPOL[30:16]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
IRQPOL[15:0]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_PINCTRL_IRQPOL3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 IRQPOL	Each bit in this register corresponds to one of the 31 pins in bank 3: 0= Low or falling edge;

HW_PINCTRL_IRQPOL3 field descriptions (continued)

Field	Description
	1= High or rising edge.

9.4.77 PINCTRL Bank 4 Interrupt Polarity Register (HW_PINCTRL_IRQPOL4)

The PINCTRL Bank 4 Interrupt Polarity Register selects the polarity for interrupt requests for the pins in bank 4.

HW_PINCTRL_IRQPOL4: 0x1340

HW_PINCTRL_IRQPOL4_SET: 0x1344

HW_PINCTRL_IRQPOL4_CLR: 0x1348

HW_PINCTRL_IRQPOL4_TOG: 0x134C

This register selects the polarity for interrupt generation. Each pin in bank 4 which is configured for interrupt generation can be independently set to interrupt on low level, high level, rising edge, or falling edge by setting this register and HW_PINCTRL_IRQLEVEL4 (see above) appropriately.

Address: HW_PINCTRL_IRQPOL4 – 8001_8000h base + 1340h offset = 8001_9340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											IRQPOL																				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQPOL4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 IRQPOL	Each bit in this register corresponds to one of the 21 pins in bank 4: 0= Low or falling edge; 1= High or rising edge.

9.4.78 PINCTRL Bank 0 Interrupt Status Register (HW_PINCTRL_IRQSTAT0)

The PINCTRL Bank 0 Interrupt Status Register reflects pending interrupt status for the pins in bank 0.

HW_PINCTRL_IRQSTAT0: 0x1400

HW_PINCTRL_IRQSTAT0_SET: 0x1404

HW_PINCTRL_IRQSTAT0_CLR: 0x1408

HW_PINCTRL_IRQSTAT0_TOG: 0x140C

This register reflects the pending interrupt status for pins in bank 0. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 0 pin which has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ0 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, for example, HW_PINCTRL_IRQSTAT0_CLR. Status bits for pins configured as level sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ0. If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN0 mask register, then the GPIO0 interrupt will be asserted to the interrupt collector.

Address: HW_PINCTRL_IRQSTAT0 – 8001_8000h base + 1400h offset = 8001_9400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			IRQSTAT																												
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQSTAT0 field descriptions

Field	Description
31–29 RSRVD1	Empty Description.
28–0 IRQSTAT	Each bit in this register corresponds to one of the 29 pins in bank 0: 0= No interrupt pending; 1= Interrupt pending.

9.4.79 PINCTRL Bank 1 Interrupt Status Register (HW_PINCTRL_IRQSTAT1)

The PINCTRL Bank 1 Interrupt Status Register reflects pending interrupt status for the pins in bank 1.

HW_PINCTRL_IRQSTAT1: 0x1410

HW_PINCTRL_IRQSTAT1_SET: 0x1414

HW_PINCTRL_IRQSTAT1_CLR: 0x1418

HW_PINCTRL_IRQSTAT1_TOG: 0x141C

This register reflects the pending interrupt status for pins in bank 1. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 1 pin which has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ1 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT1_CLR. Status bits for pins configured as level sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ1. If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN1 mask register, then the GPIO1 interrupt will be asserted to the interrupt collector.

Address: HW_PINCTRL_IRQSTAT1 – 8001_8000h base + 1410h offset = 8001_9410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IRQSTAT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQSTAT1 field descriptions

Field	Description
31–0 IRQSTAT	Each bit in this register corresponds to one of the 32 pins in bank 1: 0= No interrupt pending; 1= Interrupt pending.

9.4.80 PINCTRL Bank 2 Interrupt Status Register (HW_PINCTRL_IRQSTAT2)

The PINCTRL Bank 2 Interrupt Status Register reflects pending interrupt status for the pins in bank 2.

HW_PINCTRL_IRQSTAT2: 0x1420

HW_PINCTRL_IRQSTAT2_SET: 0x1424

HW_PINCTRL_IRQSTAT2_CLR: 0x1428

HW_PINCTRL_IRQSTAT2_TOG: 0x142C

This register reflects the pending interrupt status for pins in bank 2. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 2 pin which has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ2 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT2_CLR. Status bits for pins configured as level sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ2. If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN2 mask register, then the GPIO2 interrupt will be asserted to the interrupt collector.

Address: HW_PINCTRL_IRQSTAT2 – 8001_8000h base + 1420h offset = 8001_9420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				IRQSTAT																											
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQSTAT2 field descriptions

Field	Description
31–28 RSRVD1	Empty Description.
27–0 IRQSTAT	Each bit in this register corresponds to one of the 28 pins in bank 2: 0= No interrupt pending; 1= Interrupt pending.

9.4.81 PINCTRL Bank 3 Interrupt Status Register (HW_PINCTRL_IRQSTAT3)

The PINCTRL Bank 3 Interrupt Status Register reflects pending interrupt status for the pins in bank 3.

HW_PINCTRL_IRQSTAT3: 0x1430

HW_PINCTRL_IRQSTAT3_SET: 0x1434

HW_PINCTRL_IRQSTAT3_CLR: 0x1438

HW_PINCTRL_IRQSTAT3_TOG: 0x143C

This register reflects the pending interrupt status for pins in bank 3. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 3 pin which has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ3 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT3_CLR. Status bits for pins configured as level sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ3. If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN3 mask register, then the GPIO3 interrupt will be asserted to the interrupt collector.

Address: HW_PINCTRL_IRQSTAT3 – 8001_8000h base + 1430h offset = 8001_9430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1	IRQSTAT[30:16]														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQSTAT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQSTAT3 field descriptions

Field	Description
31 RSRVD1	Empty Description.
30–0 IRQSTAT	Each bit in this register corresponds to one of the 31 pins in bank 3: 0= No interrupt pending; 1= Interrupt pending.

9.4.82 PINCTRL Bank 4 Interrupt Status Register (HW_PINCTRL_IRQSTAT4)

The PINCTRL Bank 4 Interrupt Status Register reflects pending interrupt status for the pins in bank 4.

HW_PINCTRL_IRQSTAT4: 0x1440

HW_PINCTRL_IRQSTAT4_SET: 0x1444

HW_PINCTRL_IRQSTAT4_CLR: 0x1448

HW_PINCTRL_IRQSTAT4_TOG: 0x144C

This register reflects the pending interrupt status for pins in bank 4. Bits in this register are automatically set by hardware when an interrupt condition (level high, level low, rising edge, or falling edge) occurs on a bank 4 pin which has been enabled as an interrupts source in the HW_PINCTRL_PIN2IRQ4 register. Software may clear any bit in this register by writing a 1 to the bit at the SCT clear address, e.g., HW_PINCTRL_IRQSTAT4_CLR. Status bits for pins configured as level sensitive interrupts cannot be cleared unless either the actual pin is in the non-interrupting state, or the pin has been disabled as an interrupt source by clearing its bit in HW_PINCTRL_PIN2IRQ4. If a bit is set in this register, and the corresponding bit is also set in the HW_PINCNTRL_IRQEN4 mask register, then the GPIO4 interrupt will be asserted to the interrupt collector.

Address: HW_PINCTRL_IRQSTAT4 – 8001_8000h base + 1440h offset = 8001_9440h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1											IRQSTAT																				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_IRQSTAT4 field descriptions

Field	Description
31–21 RSRVD1	Empty Description.
20–0 IRQSTAT	Each bit in this register corresponds to one of the 21 pins in bank 4: 0= No interrupt pending; 1= Interrupt pending.

9.4.83 PINCTRL EMI Slice ODT Control (HW_PINCTRL_EMI_ODT_CTRL)

The PINCTRL EMI On Die Termination (ODT) Control Register Controls the load and calibration of each EMI slice.

HW_PINCTRL_EMI_ODT_CTRL: 0x1a40

HW_PINCTRL_EMI_ODT_CTRL_SET: 0x1a44

HW_PINCTRL_EMI_ODT_CTRL_CLR: 0x1a48

HW_PINCTRL_EMI_ODT_CTRL_TOG: 0x1a4C

This register controls On Die Termination for the EMI SLices 0-3.

Address: HW_PINCTRL_EMI_ODT_CTRL – 8001_8000h base + 1A40h offset = 8001_9A40h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1				ADDRESS_CALIB		ADDRESS_TLOAD		CONTROL_CALIB		CONTROL_TLOAD		DUALPAD_CALIB		DUALPAD_TLOAD	
W																
Reset	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	SLICE3_CALIB		SLICE3_TLOAD		SLICE2_CALIB		SLICE2_TLOAD		SLICE1_CALIB		SLICE1_TLOAD		SLICE0_CALIB		SLICE0_TLOAD	
Reset	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0

HW_PINCTRL_EMI_ODT_CTRL field descriptions

Field	Description
31–28 RSRVD1	Empty Description.

HW_PINCTRL_EMI_ODT_CTRL field descriptions (continued)

Field	Description

27–26 ADDRESS_ CALIB	ODT Calibration for EMI ADDRESS:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

25–24 ADDRESS_ TLOAD	Termination Resistince for On-Die-Termination on chip for EMI ADDRESS: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

23–22 CONTROL_ CALIB	ODT Calibration for EMI CONTROL signals:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

21–20 CONTROL_ TLOAD	Termination Resistince for On-Die-Termination on chip for EMI CONTROL: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

19–18 DUALPAD_ CALIB	ODT Calibration for EMI DUALPAD:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;

HW_PINCTRL_EMI_ODT_CTRL field descriptions (continued)

Field	Description

	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

17–16 DUALPAD_ TLOAD	Termination Resistince for On-Die-Termination on chip for EMI DUALPAD: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

15–14
SLICE3_CALIB
	ODT Calibration for EMI SLICE3:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

13–12 SLICE3_TLOAD	Termination Resistince for On-Die-Termination on chip for EMI SLICE3: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

11–10
SLICE2_CALIB
	ODT Calibration for EMI SLICE2:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

HW_PINCTRL_EMI_ODT_CTRL field descriptions (continued)

Field	Description
9–8 SLICE2_TLOAD	Termination Resistince for On-Die-Termination on chip for EMI SLICE2: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

7–6
SLICE1_CALIB
	ODT Calibration for EMI SLICE1:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

5–4 SLICE1_TLOAD	Termination Resistince for On-Die-Termination on chip for EMI SLICE1: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm 2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

3–2
SLICE0_CALIB
	ODT Calibration for EMI SLICE0:
	With On Die Termination set (through the emi controller):
	0= Disabled;
	1= 1 Res Adj;
	2= 2 Res Adj;
	3= 3 Res Adj;
	With ODT disabled:
	Disabled;

1–0 SLICE0_TLOAD	Termination Resistince for On-Die-Termination on chip for EMI SLICE0: With On Die Termination set (through the emi controller): 0= Disabled 1= 75 ohm

HW_PINCTRL_EMI_ODT_CTRL field descriptions (continued)

Field	Description
	2= 150 ohm 3= 50 ohm With ODT disabled: Disabled;

9.4.84 PINCTRL EMI Slice DS Control (HW_PINCTRL_EMI_DS_CTRL)

The PINCTRL EMI On Die Termination (DS) Control Register Controls the load and calibration of each EMI slice.

HW_PINCTRL_EMI_DS_CTRL: 0x1b80

HW_PINCTRL_EMI_DS_CTRL_SET: 0x1b84

HW_PINCTRL_EMI_DS_CTRL_CLR: 0x1b88

HW_PINCTRL_EMI_DS_CTRL_TOG: 0x1b8C

This register controls Pad Drive Strength for the EMI SLices 0-3.

Address: HW_PINCTRL_EMI_DS_CTRL – 8001_8000h base + 1B80h offset = 8001_9B80h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1														DDR_MODE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0		ADDRESS_MA		CONTROL_MA		DUALPAD_MA		SLICE3_MA		SLICE2_MA		SLICE1_MA		SLICE0_MA	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PINCTRL_EMI_DS_CTRL field descriptions

Field	Description
31–18 RSRVD1	Empty Description.
17–16 DDR_MODE	set working mode for EMI pads, default is DDR2 mode: bit 1 controls DDR2_EN

HW_PINCTRL_EMI_DS_CTRL field descriptions (continued)

Field	Description
	bit 0 controls LVDDR2_ENB 00 **mDDR** — mDDR(LPDDR1) mode 01 **DISABLED** — low power suspend mode, in which EMI pads are disabled 10 **LVDDR2** — LVDDR2 mode 11 **DDR2** — DDR2 mode
15–14 RSRVD0	Empty Description.

13–12
ADDRESS_MA
	Pin output drive strength selection for the EMI address signals:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

11–10
CONTROL_MA
	Pin output drive strength selection for the EMI control signals:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

9–8
DUALPAD_MA
	Pin output drive strength selection for the dual pads (CLK/CLKN and DQS/DQSN):
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

7–6
SLICE3_MA
	Pin output drive strength selection for SLICE 3:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

5–4
SLICE2_MA
	Pin output drive strength selection for SLICE 2:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

3–2
SLICE1_MA
	Pin output drive strength selection for SLICE 1:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;

HW_PINCTRL_EMI_DS_CTRL field descriptions (continued)

Field	Description
	11= Reserved;

1–0
SLICE0_MA
	Pin output drive strength selection for SLICE 0:
	00= 5 mA;
	01= 10 mA;
	10= 20 mA;
	11= Reserved;

Chapter 10
Clock Generation and Control (CLKCTRL)

10.1 Clock Generation and Control (CLKCTRL) Overview

The clock control module, or CLKCTRL, generates the clock domains for all components in the i.MX28 system. The crystal clock or PLL clock are the two fundamental sources used to produce all the clock domains. For lower performance and reduced power consumption, the crystal clock is selected. The PLL is selected for higher performance requirements but requires increased power consumption. In most cases, when the PLL is used as the source, a phase fractional divider (PFD) can be programmed to reduce the PLL clock frequency by up to a factor of 2.

The PLL and PFD clocks are used as reference clock sources to drive digital clock dividers in the clock control module. These reference clocks, or ref_<clock>, drive the digital clock dividers in CLKCTRL. The digital clock dividers have three modes of operation, integer divide mode, fractional divide mode, and gated clock mode. The details of these three modes will be described to understand which mode should be selected to achieve the desired frequency.

All programming control for system clocks are contained in the CLKCTRL module. All clock domains have a programmable clock frequency to meet application requirements. Also, all analog clock control programming is done indirectly through the CLKCTRL module. This contains the complexity of overall system clock selection to a single device. Also, the hardware used to generate all clock domains is replicated. Following is a description of all clock domains in the i.MX28 system.

10.2 Clock Structure

The reference clocks are used in CLKCTRL as fundamental clock sources to produce clock domains throughout the system. A reference clock can be either the crystal clock, 480 MHz PLL, or PFD output from the analog module. The selected reference clock is used by a digital clock divider to produce the desired clock domain. The table below summarizes all available reference clocks used within the CLKCTRL and all clock domains used in the system. The diagram that follows depicts all clock domains and how they are connected within the CLKCTRL module. This should provide a reference for how clocks are generated within the i.MX28 system.

10.2.1 Table of System Clocks

The following table summarizes the clocks produced by the clock control module.

Table 10-1. System Clocks

NAME	REFERENCE	DIVIDE/FREQ	DESCRIPTION
Reference Clocks.			
ref_xtal	xtal_24m/ ring_24m	1	This is the muxed select between the internal ring oscillator and the external crystal.
ref_cpu	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the CPU clock divider.
ref_emi	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the EMI clock divider.
ref_io0	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the SSP0/SSP1 clock dividers.
ref_io1	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the SSP2/SSP3 clock divider.
ref_pix	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the LCDIF clock divider.
ref_hsadc	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the HSADC clock divider.
ref_gpmi	PLL0	9 phase	The 9 phase fractional divider output used as the reference for the GPMI clock divider.
ref_pll	PLL0	1	This is the raw PLL output used as the reference for the SAIF0 and SAIF1 clock divider.
ref_enet_pll	PLL2	1	This is the raw PLL output used as the reference for the ENET clock divider.
Divided clock domains referenced from PLL or Xtal clock.			
clk_p	ref_xtal /ref_cpu	10/6 bits	ARM core clock.
clk_h	clk_p	5 bits	AHB/APBH clock domain. clk_h is a gated branch of the clk_p domain.
clk_h_flexcan0_ipg	clk_h	1	Flexcan0 Message Buffer Management (MBM) clock whose clock gating is controlled by hw_clkctrl_flexcan register.

NAME	REFERENCE	DIVIDE/FREQ	DESCRIPTION
clk_h_flexcan1_ipg	clk_h	1	Flexcan1 Message Buffer Management (MBM) clock whose clock gating is controlled by hw_clkctrl_flexcan register.
clk_h_flexcan0	clk_h	1	Flexcan0 bus clock which is gated off when no access to flexcan0 module.
clk_h_flexcan1	clk_h	1	Flexcan1 bus clock which is gated off when no access to flexcan1 module.
clk_h_enet_swi	clk_h	1	Ethernet Switch bus clock whose clock gating is controlled by hw_clkctrl_enet register.
clk_h_mac0	clk_h	1	Ethernet MAC0 bus clock whose clock gating is controlled by hw_clkctrl_enet register.
clk_h_mac1	clk_h	1	Ethernet MAC1 bus clock whose clock gating is controlled by hw_clkctrl_enet register.
clk_ocrom	clk_h	1	OCROM bus clock whose clock gating is controlled by OCROM controller H/W.
clk_etm	ref_xtal /ref_cpu	6/6 bits	ARM etm clock.
clk_emi	ref_xtal /ref_emi /ref_cpu	4/6 bits	External DDR interface clock.
clk_ssp0	ref_xtal/ref_io0	9 bits	SSP0 interface clock.
clk_ssp1	ref_xtal/ref_io0	9 bits	SSP1 interface clock.
clk_ssp2	ref_xtal/ref_io1	9 bits	SSP2 interface clock.
clk_ssp3	ref_xtal/ref_io1	9 bits	SSP3 interface clock.
clk_gpmi	ref_xtal /ref_gpmi	8 bits	General purpose memory interface clock domain.
clk_spdif/clk_pcmspdif	ref_xtal /ref_pll		Clk_spdif is an intermediate clock that drives the clk_pcmspdif fractional clock divider.
clk_saif0	ref_xtal/ref_pll	DDA	Serial Audio Interface clock domain. Its reference is the PLL clock output which drives a DDA (Digital Differential Analyzer) fractional divider.
clk_saif1	ref_xtal/ref_pll	DDA	Serial Audio Interface clock domain. Its reference is the PLL clock output which drives a DDA (Digital Differential Analyzer) fractional divider.
clk_dis_lcdif	ref_xtal/ref_pix	13 bits	LCD interface clock.
clk_hsadc	ref_hsadc	9/18/36/ 72	High-Speed ADC clock.

NAME	REFERENCE	DI-VIDE/FREQ	DESCRIPTION
clk_enet_time	ref_pll /ref_xtal /ref_net_pll	6 bits	Ethernet 1588 timer clock
Divided clock domains referenced from Xtal clock.			
clk_x	ref_xtal	10 bits	APBX clock domain.
clk_uart	ref_xtal	2 bits	UART clock domain.
Fixed clock domains.			
clk_xtal24m	ref_xtal	24Mhz	Used for the PWM and analog 24 MHz clock domains.
clk_32k	ref_xtal	32khz	Fixed 32 KHz clock domain. The reference is the 24 MHz crystal and divides by 768 to produce 32 KHz.
clk_flex-can0_nogate	ref_xtal	24MHz	Flexcan0 CAN Protocol Interface(CPI) clock without gating.
clk_flex-can1_nogate	ref_xtal	24MHz	Flexcan1 CAN Protocol Interface(CPI) clock without gating.
clk_flexcan0	ref_xtal	24MHz	Flexcan0 CAN Protocol Interface (CPI) clock with gating which is controlled by FLEXCAN H/W.
clk_flexcan1	ref_xtal	24MHz	Flexcan1 CAN Protocol Interface (CPI) clock with gating which is controlled by FLEXCAN H/W.
clk_lradc2k	ref_xtal	2khz	Fixed 2 KHz clock domain.

10.2.2 Logical Diagram of Clock Domains

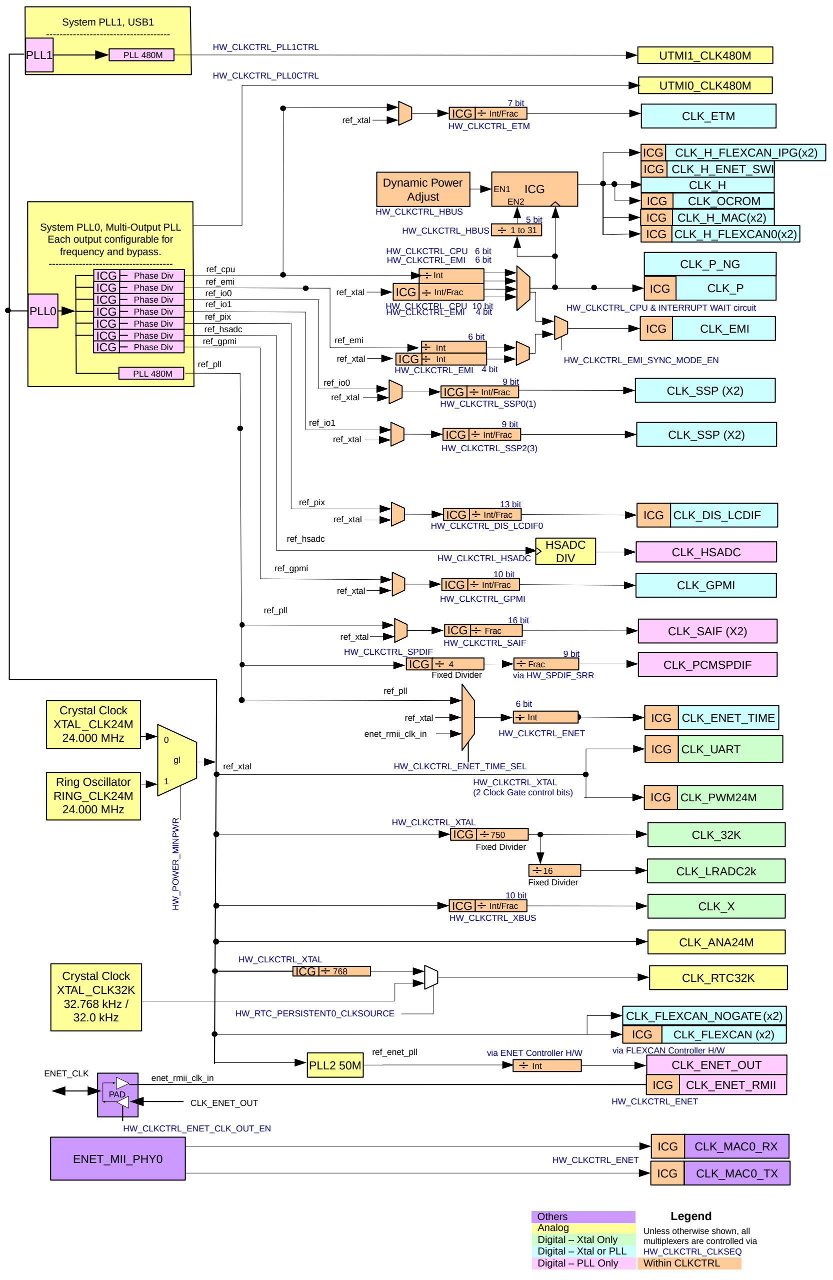


Figure 10-1. Logical Diagram of Clock Domains

10.2.3 Clock Domain Description

All major functional clock domains/branches have trunk level clock gating for power management. The intent is to gate clock domains off when modules of certain applications are not necessary. This clock gating is instantiated using an ICG element from the standard cell library. Software will have to enable the clock domain that drives on chip devices where trunk level clock gating is implemented.

The location of ICG elements to gate clock domains is not systematic. Since most of the clock structures throughout the system are unique, the location of ICGs for clock tree power reduction differs from one domain to the other. The location of these ICGs to gate off clock domains is apparent in the clock structure diagram.

All clock domains are asynchronous unless noted otherwise.

10.2.3.1 CLK_P, CLK_H

The clk_p domain is used to drive the integrated ARM9 core. The reference for clk_p can be either ref_xtal or ref_cpu. The reference ref_cpu drives a 6-bit clock divider to provide a maximum divide down of the reference clock by 2^6. The reference ref_xtal drives a 10-bit clock divider to provide a maximum divide down of the selected reference clock by 2^10. All of the ARM core and SoC components on the clk_h branch are considered to be on the clk_p domain. The clk_h is actually a branch of the clk_p domain. So, clk_h runs synchronous to clk_p.

The clk_h domain can be programmed to any divided ratio with respect to the clk_p domain depending on performance and power requirements. A dynamic clock frequency management controller monitors the system performance requirements and scales the clk_h frequency to meet the performance needs. When the CPU or support components require data transfer to/from the system memory, the frequency manager scales the clk_h domain to meet the system performance requirements. Also, when the system is quiesced, the clk_h frequency is reduced to save power.

The clk_h has a 5-bit divider that divides the clk_p domain to produce the clk_h domain. The frequency for clk_h can be clk_p/32 <= clk_h <= clk_p. Two divide modes exist for the clk_h branch:

- Integer divide: In this mode, the value programmed in the hw_clkctrl_hbus.div field represents an integer divide value.
- Fractional divide: In this mode, the value programmed in the hw_clkctrl_hbus.div field represents a binary fraction. When the accumulation of the current count and the programmed divide value carry out of the most significant bit, a clk_h pulse is generated.

For example, to achieve a 8:3 clk_p:clk_h clock ratio, set the div field to 0.01100 which represents (0*1/2) + (1*1/4) + (1*1/8) + (0*1/16) + (0*1/32). Note, fractional divide can not be used when clk_emi is synchronous with clk_h.

The clk_h branch can be further divided by the dynamic clock frequency adjustment logic, (hw_clkctrl_emi_sync_mode_en = 0 only). When all the system clk_h components are not busy and their respective busy signals are inactive, the clk_h branch is further divided down by the value in the hw_clkctrl_hbus register. The frequency reduction of the clk_h branch saves overall power consumption. Note, the dynamic clock frequency adjustment logic should not be enabled when clk_emi is synchronous with clk_h.

10.2.3.2 CLK_EMI

The external memory interface domain is called clk_emi. This clock can be asynchronous to clk_h to achieve the highest possible clock rate for the EMI interface, or synchronously to minimize the incurred latency for CPU access to external DRAM. This option is provided to tradeoff the optimization of performace for systems that are dependent on memory access latency or throughput.

When the hw_clkctrl_emi_sync_mode_en bit is set to 1, clk_h is synchronous and edge aligned with the emi clock and clk_p. The emi clock dividers will set the frequency of clk_h and clk_emi domains when synchronous mode is selected. In synchronous mode, the dynamic clock frequency adjust logic should be disabled. This is required since DRAM devices cannot operate correctly with changing clock frequencies.

10.2.3.3 System Clocks

All reference clock domains used in the CLKCTRL are driven by replicated instances of the PFD pre dividers in the analog module. These PFD reference clocks drive replicated instances of a single digital clock divider design to create all system clocks. The following sections describe the features of the digital clock dividers and how these drives can be used to create clocks throughout the system. The CLKCTRL structural diagram should be used with the digital clock divider description to understand how clocks are generated in the i.MX28 system.

10.3 CLKCTRL Digital Clock Divider

The digital clock divider that is used to drive all the functional clock domains has three modes of operation. These include:

- Integer divide mode

- Fractional divide mode
- Gated clock divide mode

These modes are described in the following three sections.

10.3.1 Integer Clock Divide Mode

Each divider has the capability to divide an input reference frequency by a fixed integer value. This is the most common mode that will be used to select a particular clock frequency. For a desired clock frequency, first try to select a PFD reference clock frequency AND an integer clock divide value to achieve the desired clock domain frequency. This mode is selected when the respective frac_en field in the clock control register is logic 0. The divide value will be in the range of 1 to 2^N. When programming the DIV field to 1, the reference clock for the domain is passed and the clock domain assumes the same frequency as the reference domain. When a value of 2 is programmed, the clock domain frequency will be half the reference clock frequency. The maximum divide value depends on the number of bits each digital clock divider implements. This is different for each digital clock divider. The number of bits implemented for each divider is indicated by each DIV field that controls each clock domain. Divide by zero is NOT a valid programming value for the DIV field of any clock control PIO register.

10.3.2 Fractional Clock Divide Mode

This mode is used to divide a reference clock in the range of 2 < div < 2^N. The fractional clock divider in the CLKCTRL module implements a fractional counter to approximate a divided clock with respect to the selected reference frequency. The accuracy of the output clock is dependent on the extent of the bits used to implement the fractional counter. The reference clock frequency and the fractional divide value must both be selected to achieve the desired output frequency.

This mode is enabled when setting the FRAC_EN field of the respective clock domain control register to logic 1 AND the most significant bit of the DIV field is logic 0. Do NOT use this mode to divide the reference clock domain by an integer value (such as 4, 8, and so on). Use the integer divide mode to achieve the best results to divide by an integer.

Note

It is important to note that the nearest rising or falling edge of the input reference clock frequency is used to approximate the rising edge of the output clock domain. So, the output clock frequency will jitter based on the input reference clock frequency and the programmed fractional divide value.

10.3.2.1 Fractional Clock Divide Example, Divide by 3.5

As an example, if the desired divide value is 3.5, the digital approximation of 1/3.5 is 0.01001001 using a 8 bit fractional approximation. The most significant bit of the div field in this case is logic 0, so the fractional divide mode is selected. The following sequence indicates the first eight values of the fractional clock divider. The accumulated count is simply the current value incremented by the value programmed in the div field on each cycle.

1. 0.01001001
2. 0.10010010
3. 0.11011011
4. 1.00100100 (carry out of MSB initiates an output clock edge)
5. 0.01101101
6. 0.10110110
7. 0.11111111
8. 1.01001000 (output edge initiated)

When the carry out of the fractional count is one, a rising edge output pulse is initiated and the remainder of the accumulator is preserved. The sub-fractional accumulated value is considered to determine if the output edge should occur on the falling edge of the reference clock or the rising edge of the reference clock to minimize the output clock jitter.

10.3.2.1.1 Fractional ClockDivide Example, Divide by 3/8

This example uses a 3-bit fractional accumulator to divide the reference clock input by 3/8. There are 3 output clock edges produced for every eight input reference clock edges. An output edge is generated on every cycle that the fractional accumulator carries out of the most significant bit. Notice when the fractional component is .01, the output edge is shifted and generated off the falling edge of the input reference clock. This is done to produce the best output duty cycle that can be achieved based on the input reference clock frequency.

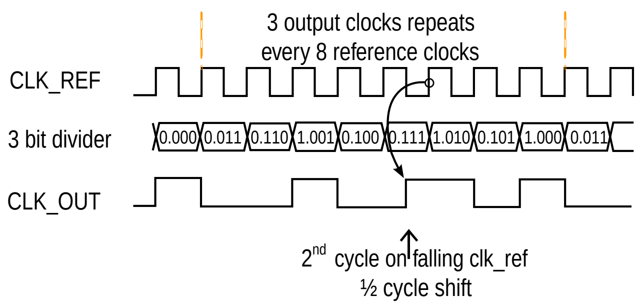


Figure 10-2. Fractional Clock divide; 3/8 example

10.3.3 Gated Clock Divide Mode

This mode is selected when the reference clock frequency is divided by a range of 1 < div < 2. To select this mode, program the FRAC_EN field to logic 1 and progarm the DIV field with the most significant bit set to logic 1. In this case, the reference clock is enabled/disabled on a cycle by cycle basis to pass to the output clock domain. Essentially, the reference clock is gated on or off depending on the carry out bit of the fractional count accumulator. This option is useful to divide the 24 MHz clock to a range between 12 to 24 MHz. The effective period is equal to the reference period since the output clock is a gated version of the reference clock. For example, a divide value of 4/3 will allow three consecutive pulses of the reference clock to propagate and will then gate off a single reference clock cycle. The edge to edge timing is effectively equal to the reference clock.

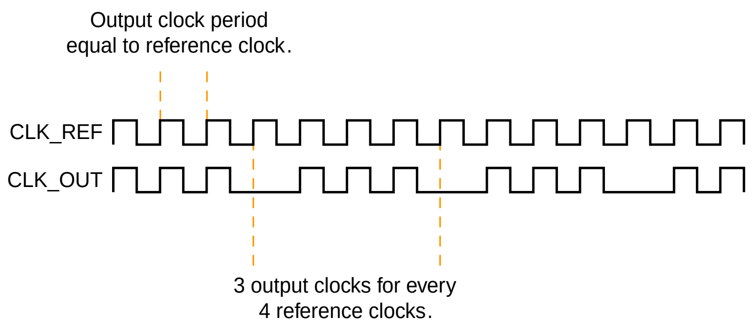


Figure 10-3. Divide Range 1 < div < 2

10.4 Clock Frequency Management

Clock frequency selection for some domains can be a function of multiple reference clock sources and divide parameters that are set in the CLKCTRL PIO control registers. The most extreme case is using a programmable fractional PLL clock divider, a multiplexer that selects either the xtal clock or fractional PLL clock as a source to drive the CLKCTRL divider, and the divide value for the CLKCTRL divider itself. When programming a selected frequency, the sequence of events to achieve a given frequency must maintain the integrity of the system as a whole. During a clock system context switch, intermediate clock frequencies for the selected domains cannot be faster than the sub system or the I/O interface which is designed to support.

It is expected that the sequence of events when a clock domain is tuned to a desired frequency be managed by a software using a hardware status polling mechanism. Each parameter has an associated enable bit so that all the divide parameters can be programmed in advance of the parameters taking effect. A single register, hw_clkctrl_clkseq, contains all the enable bits that cause the divide parameters to take effect. The enable bits can be set, the busy bits can be polled for each parameter, and thus the enable/busy sequencing through software control can manage the tuning of clock frequencies throughout the system.

10.5 Analog Clock Control

Analog clock control is performed indirectly through PIO accessable registers in the CLKCTRL module. The analog circuits that are controlled through CLKCTRL PIO access are the PLL and all instances of the phase fractional dividers, or PFDs.

10.6 CPU and EMI Clock Programming

A defined protocol is necessary for selecting clock frequencies and root sources for driving the clk_p and clk_emi domains. These two clock structures are unique in that each implement a separate divider, one referenced by xtal clock and the other referenced by a PLL/PFD structure. The roots of these clocks must be programmed in order of the sources furthest from the trunk first. Elements in the clock roots should subsequently be configured along the root path up to the desired clock trunk. The programming sequence to go from a clock that is referenced from the xtal clock to the PLL is outlined below. This is the case when the device is in low power operation and there exists the need for higher clock rates to meet the demands of a more compute-intensive application.

1. The crystal is the current source for the CPU or the EMI clock domain.
2. Enable the PLL.
3. Wait for PLL lock.
4. Program and enable the PFD with the desired configuration.
5. Clear the PFD clock gate to establish the desired reference clock frequency.
6. Program the CLKCTRL clock divider register (EMI or CPU) that uses the PLL/PFD as its reference clock.

7. Switch the bypass to *off* (select PLL, not crystal).

The requirement is that the roots of the clock are configured and stable before the elements higher up in the tree are programmed. This will allow the roots to stabilize before selected as a valid source to drive a clock trunk/tree. If this sequence is not honored, unpredictable frequencies can occur which may violate the maximum operating frequency of components on the respective clock trees. Be sure to gate off the clock paths directly downstream from the PLL before powering off the PLL.

When clk_emi is operating in synchronous mode, the following requirements must be maintained:

- The clk_p divide value is less than or equal to the clk_emi divide value.
- The clk_emi divide value must be divisable by the clk_p divide value. An example of possible clk_p:clk_emi divide values would be, but not limited, to 1:1, 1:2, 1:3, 2:2, 2:4, 2:6, 3:3, 3:6, 3:9.

10.7 Chip Reset

Two PIO accessible soft reset bits exists to establish the initial state of the device. These bits are called HW_CLKCTRL_RESET_CHIP and HW_CLKCTRL_RESET_DIG. Setting these bits will result in a chip wide reset cycle. When setting the DIG software reset bit, the digital logic is reset with the exception of the power module and the DCDC converter control logic. The CHIP software reset bit also initiates the full reset cycle and the power and the DCDC converter logic are also reset. These two soft reset bits themselves reset during a soft reset sequence.

Ethernet module is a special case. To avoid network traffic blocking, Ethernet Switch need to maintain working as long as possible. So when the Ethernet module is in Switch mode, it can be configured to only POR resettable by setting both HW_CLKCTRL_ENET_RESET_BY_SW_CHIP and HW_CLKCTRL_ENET_RESET_BY_SW to 0. If Ethernet module is not in Switch mode, it will be reset by both DIG software reset bit and CHIP reset bit regardless of the value of HW_CLKCTRL_ENET_RESET_BY_SW_CHIP and HW_CLKCTRL_ENET_RESET_BY_SW.

The following figure shows the functionality of these two reset bits.

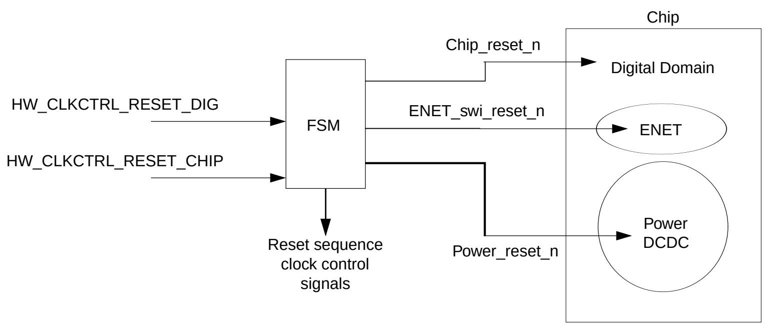


Figure 10-4. Reset Logic Functional Diagram

10.8 Programmable Registers

CLKCTRL Hardware Register Format Summary

HW_CLKCTRL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8004_0000	System PLL0, System/USB0 PLL Control Register 0 (HW_CLKCTRL_PLL0CTRL0)	32	R/W	0000_0000h	10.8.1/856
8004_0010	System PLL0, System/USB0 PLL Control Register 1 (HW_CLKCTRL_PLL0CTRL1)	32	R/W	0000_0000h	10.8.2/858
8004_0020	System PLL1, USB1 PLL Control Register 0 (HW_CLKCTRL_PLL1CTRL0)	32	R/W	8000_0000h	10.8.3/859
8004_0030	System PLL1, USB1 PLL Control Register 1 (HW_CLKCTRL_PLL1CTRL1)	32	R/W	0000_0000h	10.8.4/861
8004_0040	System PLL2, Ethernet PLL Control Register 0 (HW_CLKCTRL_PLL2CTRL0)	32	R/W	8000_0000h	10.8.5/862
8004_0050	CPU Clock Control Register (HW_CLKCTRL_CPU)	32	R/W	0001_0001h	10.8.6/863
8004_0060	AHB, APBH Bus Clock Control Register (HW_CLKCTRL_HBUS)	32	R/W	0000_0001h	10.8.7/865
8004_0070	APBX Clock Control Register (HW_CLKCTRL_XBUS)	32	R/W	0000_0100h	10.8.8/867
8004_0080	XTAL Clock Control Register (HW_CLKCTRL_XTAL)	32	R/W	6000_0001h	10.8.9/868
8004_0090	Synchronous Serial Port0 Clock Control Register (HW_CLKCTRL_SSP0)	32	R/W	8000_0001h	10.8.10/869
8004_00A0	Synchronous Serial Port1 Clock Control Register (HW_CLKCTRL_SSP1)	32	R/W	8000_0001h	10.8.11/870
8004_00B0	Synchronous Serial Port2 Clock Control Register (HW_CLKCTRL_SSP2)	32	R/W	8000_0001h	10.8.12/871
8004_00C0	Synchronous Serial Port3 Clock Control Register (HW_CLKCTRL_SSP3)	32	R/W	8000_0001h	10.8.13/873
8004_00D0	General-Purpose Media Interface Clock Control Register (HW_CLKCTRL_GPMI)	32	R/W	8000_0001h	10.8.14/874

HW_CLKCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8004_00E0	SPDIF Clock Control Register (HW_CLKCTRL_SPDIF)	32	R/W	8000_0000h	10.8.15/875
8004_00F0	EMI Clock Control Register (HW_CLKCTRL_EMI)	32	R/W	8000_0101h	10.8.16/876
8004_0100	SAIF0 Clock Control Register (HW_CLKCTRL_SAIF0)	32	R/W	8000_0001h	10.8.17/877
8004_0110	SAIF1 Clock Control Register (HW_CLKCTRL_SAIF1)	32	R/W	8000_0001h	10.8.18/878
8004_0120	CLK_DIS_LCDIF Clock Control Register (HW_CLKCTRL_DIS_LCDIF)	32	R/W	8000_0001h	10.8.19/880
8004_0130	ETM Clock Control Register (HW_CLKCTRL_ETM)	32	R/W	8000_0001h	10.8.20/881
8004_0140	ENET Clock Control Register (HW_CLKCTRL_ENET)	32	R/W	E010_0000h	10.8.21/882
8004_0150	HSADC Clock Control Register (HW_CLKCTRL_HSADC)	32	R/W	0000_0000h	10.8.22/883
8004_0160	FLEXCAN Clock Control Register (HW_CLKCTRL_FLEXCAN)	32	R/W	7800_0000h	10.8.23/884
8004_01B0	Fractional Clock Control Register 0 (HW_CLKCTRL_FRAC0)	32	R/W	9292_9292h	10.8.24/885
8004_01C0	Fractional Clock Control Register 1 (HW_CLKCTRL_FRAC1)	32	R/W	0092_9292h	10.8.25/887
8004_01D0	Clock Frequency Sequence Control Register (HW_CLKCTRL_CLKSEQ)	32	R/W	0044_83FFh	10.8.26/889
8004_01E0	System Reset Control Register (HW_CLKCTRL_RESET)	32	R/W	0000_0012h	10.8.27/891
8004_01F0	ClkCtrl Status (HW_CLKCTRL_STATUS)	32	R	0000_0000h	10.8.28/892
8004_0200	ClkCtrl Version (HW_CLKCTRL_VERSION)	32	R	0C02_0000h	10.8.29/893

10.8.1 System PLL0, System/USB0 PLL Control Register 0 (HW_CLKCTRL_PLL0CTRL0)

HW_CLKCTRL_PLL0CTRL0: 0x000

HW_CLKCTRL_PLL0CTRL0_SET: 0x004

HW_CLKCTRL_PLL0CTRL0_CLR: 0x008

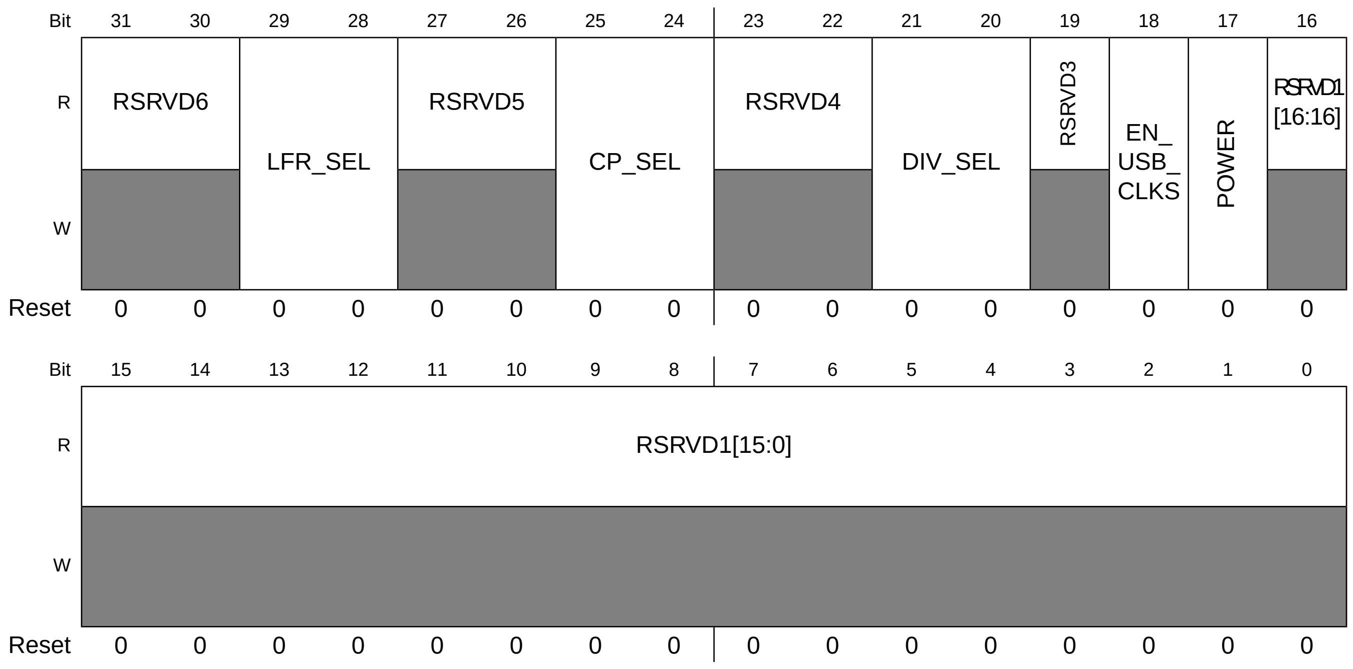
HW_CLKCTRL_PLL0CTRL0_TOG: 0x00C

The PLL0 Control Register 0 programs the 480 MHz PLL0 and the USB0-clock enables.

EXAMPLE

```
HW_CLKCTRL_PLL0CTRL0_WR(BF_CLKCTRL_PLL0CTRL0_POWER(1));  // enable PLL and wait 10 us to let
PLL0 lock before using it
```

Address: HW_CLKCTRL_PLL0CTRL0 – 8004_0000h base + 0h offset = 8004_0000h



HW_CLKCTRL_PLL0CTRL0 field descriptions

Field	Description
31–30 RSRVD6	Always set to zero (0).
29–28 LFR_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts loop filter resistor. 0x0 **DEFAULT** — Default loop filter resistor 0x1 **TIMES_2** — Doubles the loop filter resistor 0x2 **TIMES_05** — Halves the loop filter resistor 0x3 **UNDEFINED** — Undefined
27–26 RSRVD5	Always set to zero (0).
25–24 CP_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts charge pump current 0x0 **DEFAULT** — Default charge pump current 0x1 **TIMES_2** — Doubles charge pump current 0x2 **TIMES_05** — Halves the charge pump current 0x3 **UNDEFINED** — Undefined
23–22 RSRVD4	Always set to zero (0).
21–20 DIV_SEL	TEST MODE FOR FREESCALE USE ONLY. This field is currently NOT supported. 0x0 **DEFAULT** — PLL0 frequency is 480 Mhz 0x1 **LOWER** — Lower the PLL0 fequency from 480MHz to 384Mhz 0x2 **LOWEST** — Lower the PLL0 fequency from 480MHz to 288MHz 0x3 **UNDEFINED** — Undefined
19 RSRVD3	Always set to zero (0).

HW_CLKCTRL_PLL0CTRL0 field descriptions (continued)

Field	Description
18 EN_USB_CLKS	0: 8-phase PLL outputs for USB0 PHY are powered down. If set to 1, 8-phase PLL outputs for USB0 PHY are powered up. Additionally, the utmi clock gate must be deasserted in the UTMI0 phy to enable USB0 operation. If HW_USBPHY_CTRL.ENAUTOSET_USBCLKS of USB0 is set, this bit will be set automatically when USB0 remote wakeup event happens.
17 POWER	PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL0 on before using the PLL0 as a clock source. This is the time the PLL0 takes to lock to 480 MHz. If HW_USB_PHY_CTRL_ENAUTO_PWRON_PLL of UTM0 is set, this bit will be set to one (1'b1) automatically when either USB0 or USB1 remote wakeup event happens. Note: The POWER bit must be set to on to ungate the reference xtal to all of the PLLs.
16–0 RSRVD1	Always set to zero (0).

10.8.2 System PLL0, System/USB0 PLL Control Register 1 (HW_CLKCTRL_PLL0CTRL1)

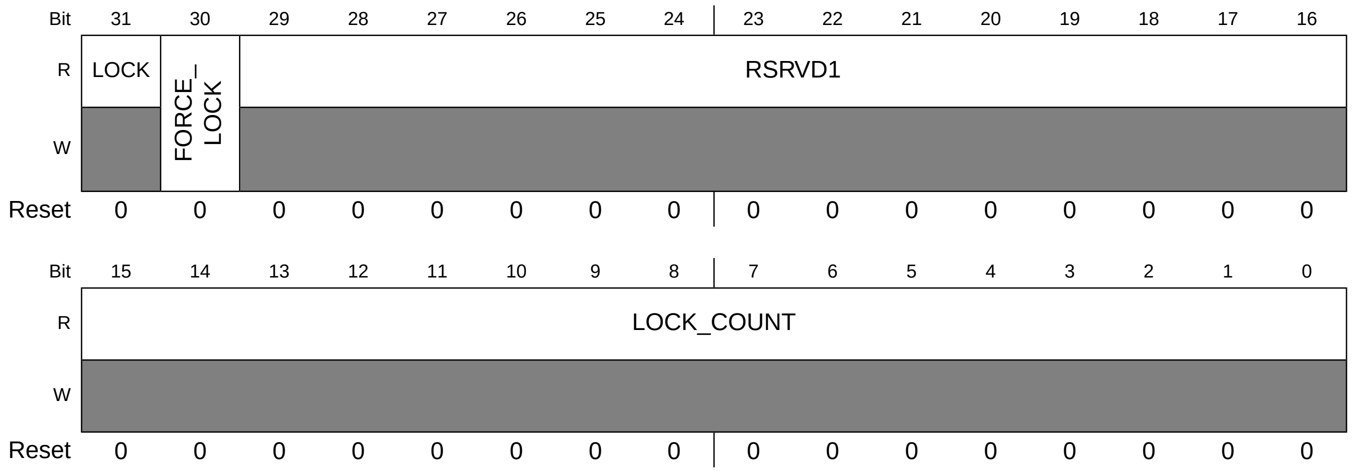
PLL0 Lock Control Register

The lock count is driven off of xtal. So after the PLL0 is powered on, the PLL0 Lock should be asserted after 50 us.

EXAMPLE

```
HW_CLKCTRL_PLL0CTRL1_WR(BF_CLKCTRL_PLL0CTRL1_FORCE_LOCK(1));  // force pll lock sequence
                             HW_CLKCTRL_PLL0CTRL1_WR(BF_CLKCTRL_PLL0CTRL1_FORCE_LOCK(0));  // clear
 force pll lock
```

Address: HW_CLKCTRL_PLL0CTRL1 – 8004_0000h base + 10h offset = 8004_0010h



HW_CLKCTRL_PLL0CTRL1 field descriptions

Field	Description
31 LOCK	PLL0 Lock bit. 1=PLL0 Locked. 0=PLL0 Unlocked.
30 FORCE_LOCK	Force the PLL0 Lock sequence to start. 1=Enable Force Lock. This bit is not self clearing.
29–16 RSRVD1	Reserved - Always set to zero (0).
15–0 LOCK_COUNT	Status of the PLL0 lock count. The PLL0 lock bit will assert when the count reaches 0x4B0. The lock count is driven off of xtal, so it should be asserted after 50 us.

10.8.3 System PLL1, USB1 PLL Control Register 0 (HW_CLKCTRL_PLL1CTRL0)

HW_CLKCTRL_PLL1CTRL0: 0x020

HW_CLKCTRL_PLL1CTRL0_SET: 0x024

HW_CLKCTRL_PLL1CTRL0_CLR: 0x028

HW_CLKCTRL_PLL1CTRL0_TOG: 0x02C

The PLL1 Control Register 0 programs the 480 MHz PLL1 and the USB1-clock enables.

EXAMPLE

```
HW_CLKCTRL_PLL1CTRL2_WR(BF_CLKCTRL_PLL1CTRL0_POWER(1));  // enable PLL and wait 10 us to let
PLL1 lock before using it
```

Address: HW_CLKCTRL_PLL1CTRL0 – 8004_0000h base + 20h offset = 8004_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATEEMI	RSRVD6	LFR_SEL		RSRVD5		CP_SEL		RSRVD4		DIV_SEL		RSRVD3	EN_ USB_ CLKS	POWER	RSRVD1 [16:16]
W	CLKGATEEMI	▨	LFR_SEL		▨	▨	CP_SEL		▨	▨	DIV_SEL		▨	EN_ USB_ CLKS	POWER	▨
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_PLL1CTRL0 field descriptions

Field	Description
31 CLKGATEEMI	TEST MODE FOR FREESCALE USE ONLY.
30 RSRVD6	Always set to zero (0).
29–28 LFR_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts loop filter resistor. 0x0 **DEFAULT** — Default loop filter resistor 0x1 **TIMES_2** — Doubles the loop filter resistor 0x2 **TIMES_05** — Halves the loop filter resistor 0x3 **UNDEFINED** — Undefined
27–26 RSRVD5	Always set to zero (0).
25–24 CP_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts charge pump current 0x0 **DEFAULT** — Default charge pump current 0x1 **TIMES_2** — Doubles charge pump current 0x2 **TIMES_05** — Halves the charge pump current 0x3 **UNDEFINED** — Undefined
23–22 RSRVD4	Always set to zero (0).
21–20 DIV_SEL	TEST MODE FOR FREESCALE USE ONLY. This field is currently NOT supported. 0x0 **DEFAULT** — PLL1 frequency is 480 Mhz 0x1 **LOWER** — Lower the PLL1 fequency from 480MHz to 384Mhz 0x2 **LOWEST** — Lower the PLL1 fequency from 480MHz to 288MHz 0x3 **UNDEFINED** — Undefined
19 RSRVD3	Always set to zero (0).
18 EN_USB_CLKS	0: 8-phase PLL outputs for USB1 PHY are powered down. If set to 1, 8-phase PLL outputs for USB1 PHY are powered up. Additionally, the utmi clock gate must be deasserted in the UTMI1 phy to enable USB1 operation. If HW_USBPHY_CTRL.ENAUTOSET_USBCLKS of USB1 is set, this bit will be set automatically when USB0 remote wakeup event happens.
17 POWER	PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL1 on before using the PLL1 as a clock source. This is the time the PLL1 takes to lock to 480 MHz. If HW_USB_PHY_CTRL_ENAUTO_PWRON_PLL of UTM1 is set, this bit will be set to one (1'b1) automatically when USB1 remote wakeup event happens. Note: The HW_CLKCTRL_PLL0CTRL0_POWER bit must be set to on to ungate the reference xtal to all of the PLLs.

HW_CLKCTRL_PLL1CTRL0 field descriptions (continued)

Field	Description
16–0 RSRVD1	Always set to zero (0).

10.8.4 System PLL1, USB1 PLL Control Register 1 (HW_CLKCTRL_PLL1CTRL1)

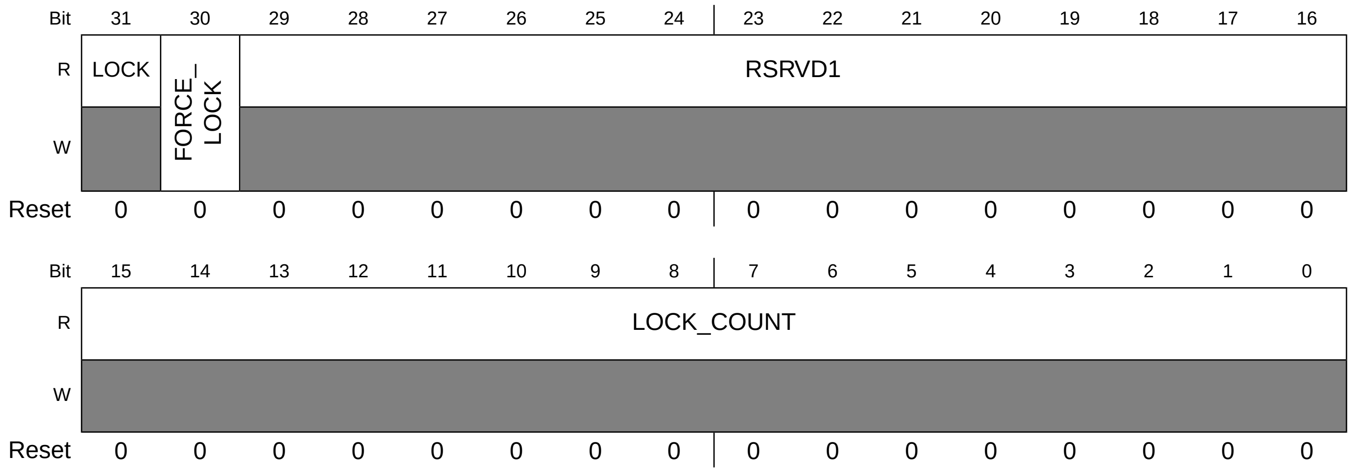
PLL1 Lock Control Register

The lock count is driven off of xtal. So after the PLL1 is powered on, the PLL1 Lock should be asserted after 50 us.

EXAMPLE

```
HW_CLKCTRL_PLL1CTRL1_WR(BF_CLKCTRL_PLL1CTRL1_FORCE_LOCK(1));  // force pll lock sequence
                              HW_CLKCTRL_PLL1CTRL1_WR(BF_CLKCTRL_PLL1CTRL1_FORCE_LOCK(0));  // clear
 force pll lock
```

Address: HW_CLKCTRL_PLL1CTRL1 – 8004_0000h base + 30h offset = 8004_0030h



HW_CLKCTRL_PLL1CTRL1 field descriptions

Field	Description
31 LOCK	PLL1 Lock bit. 1=PLL1 Locked. 0=PLL1 Unlocked.
30 FORCE_LOCK	Force the PLL1 Lock sequence to start. 1=Enable Force Lock. This bit is not self clearing.
29–16 RSRVD1	Reserved - Always set to zero (0).
15–0 LOCK_COUNT	Status of the PLL1 lock count. The PLL1 lock bit will assert when the count reaches 0x4B0. The lock count is driven off of xtal, so it should be asserted after 50 us.

10.8.5 System PLL2, Ethernet PLL Control Register 0 (HW_CLKCTRL_PLL2CTRL0)

HW_CLKCTRL_PLL2CTRL0: 0x040

HW_CLKCTRL_PLL2CTRL0_SET: 0x044

HW_CLKCTRL_PLL2CTRL0_CLR: 0x048

HW_CLKCTRL_PLL2CTRL0_TOG: 0x04C

The System PLL2 Control Register 0 programs the Ethernet PLL.

EXAMPLE

```
HW_CLKCTRL_PLL2CTRL0_WR(BF_CLKCTRL_PLL2CTRL0_POWER(1));  // enable PLL and wait 10 us to let
PLL lock before using it
```

Address: HW_CLKCTRL_PLL2CTRL0 – 8004_0000h base + 40h offset = 8004_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD3	LFR_SEL		RSRVD2	HOLD_RING_OFF_B	CP_SEL		POWER	RSRVD1[22:16]						
W		▨			▨					▨						
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:0]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_PLL2CTRL0 field descriptions

Field	Description
31 CLKGATE	PLL Clock Gate. If set to 1, the ENET PLL clock is off (power savings). 0: ENET PLL clock is enabled.
30 RSRVD3	Always set to zero (0).
29–28 LFR_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts loop filter resistor.

HW_CLKCTRL_PLL2CTRL0 field descriptions (continued)

Field	Description
27 RSRVD2	Always set to zero (0).
26 HOLD_RING_ OFF_B	TEST MODE FOR FREESCALE USE ONLY. Adjust VCO phase.
25–24 CP_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts charge pump current.
23 POWER	PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL on before ungating the PLL and using it as a clock source. This is the time the PLL takes to lock.
22–0 RSRVD1	Always set to zero (0).

10.8.6 CPU Clock Control Register (HW_CLKCTRL_CPU)

The CPUCLK Clock Control Register provides controls for generating the ARM CPUCLK.

HW_CLKCTRL_CPU: 0x050

HW_CLKCTRL_CPU_SET: 0x054

HW_CLKCTRL_CPU_CLR: 0x058

HW_CLKCTRL_CPU_TOG: 0x05c

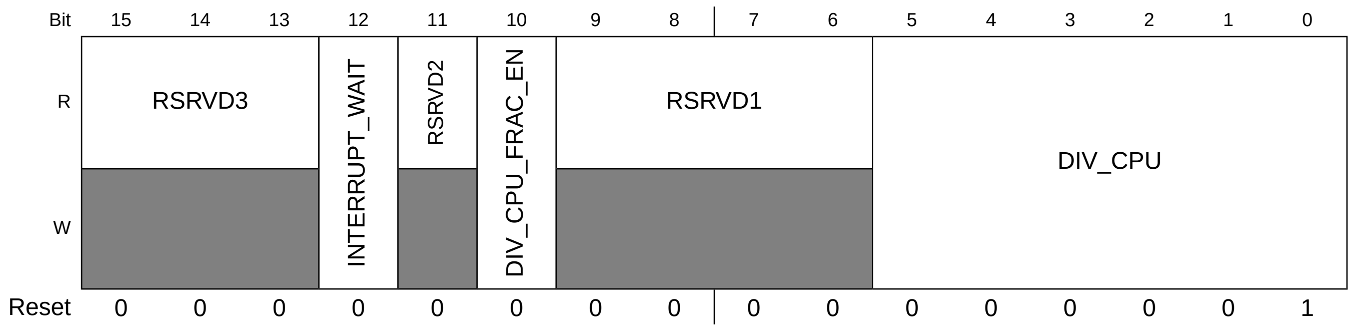
Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_CPU_WR(BF_CLKCTRL_DIV_CPU(12)); // 480 MHz / 12 = 40 MHz
```

Address: HW_CLKCTRL_CPU – 8004_0000h base + 50h offset = 8004_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD5		BUSY_REF_XTAL	BUSY_REF_CPU	RSRVD4	DIV_XTAL_FRAC_EN	DIV_XTAL									
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



HW_CLKCTRL_CPU field descriptions

Field	Description
31–30 RSRVD5	Always set to zero (0).
29 BUSY_REF_ XTAL	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28 BUSY_REF_CPU	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
27 RSRVD4	Always set to zero (0).
26 DIV_XTAL_ FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
25–16 DIV_XTAL	This field controls the divider connected to the crystal reference clock that drives the CLK_P domain when bypass is selected. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
15–13 RSRVD3	Always set to zero (0).
12 INTERRUPT_ WAIT	Gate off CLK_P while waiting for an interrupt.
11 RSRVD2	Always set to zero (0).
10 DIV_CPU_ FRAC_EN	Reserved - Always set to zero (0).
9–6 RSRVD1	Always set to zero (0).
5–0 DIV_CPU	This field controls the divider connected to the ref_cpu reference clock that drives the CLK_P domain when bypass is NOT selected. For changes to this field to take effect, the ref_cpu reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.7 AHB, APBH Bus Clock Control Register (HW_CLKCTRL_HBUS)

HW_CLKCTRL_HBUS: 0x060

HW_CLKCTRL_HBUS_SET: 0x064

HW_CLKCTRL_HBUS_CLR: 0x068

HW_CLKCTRL_HBUS_TOG: 0x06c

This register controls the clock divider that generates the CLK_H, the clock used by the AHB and APBH buses, when HW_CLKCTRL_EMI_SYNC_MODE_EN = 0. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_HBUS_WR(BF_CLKCTRL_HBUS_DIV(2)); // set CLK_H to half the ARM clock (CLK_P) frequency
```

Address: HW_CLKCTRL_HBUS – 8004_0000h base + 60h offset = 8004_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ASM_ BUSY	DCP_AS_ENABLE	PXP_AS_ENABLE	RSRVD2	ASM_EMIPORT_AS_ ENABLE	APBHDMA_AS_ ENABLE	APBXDMA_AS_ ENABLE	TRAFFIC_JAM_AS_ ENABLE	TRAFFIC_AS_ ENABLE	CPU_DATA_AS_ ENABLE	CPU_INSTR_AS_ ENABLE	ASM_ENABLE	AUTO_CLEAR_DIV_ ENABLE	SLOW_DIV		
W	▨			▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1										DIV_FRAC_EN	DIV				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_HBUS field descriptions

Field	Description
31 ASM_BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.

HW_CLKCTRL_HBUS field descriptions (continued)

Field	Description
30 DCP_AS_ ENABLE	Enable auto-slow mode based on DCP activity. 0 = Run at the programmed CLK_H frequency.
29 PXP_AS_ ENABLE	Enable auto-slow mode based on PXP activity. 0 = Run at the programmed CLK_H frequency.
28 RSRVD2	Reserved
27 ASM_EMIPORT_ AS_ENABLE	Enable auto-slow mode based on EMI axi0 port activity. 0 = Run at the programmed CLK_HS frequency.
26 APBHDMA_AS_ ENABLE	Enable auto-slow mode based on APBH DMA activity. 0 = Run at the programmed CLK_H frequency.
25 APBXDMA_AS_ ENABLE	Enable auto-slow mode based on APBX DMA activity. 0 = Run at the programmed CLK_H frequency.
24 TRAFFIC_JAM_ AS_ENABLE	Enable auto-slow mode when less than three masters are trying to use the AHB. More than three active masters will engage the default mode. 0 = Run at the programmed CLK_H frequency.
23 TRAFFIC_AS_ ENABLE	Enable auto-slow mode based on AHB master activity. 0 = Run at the programmed CLK_H frequency.
22 CPU_DATA_AS_ ENABLE	Enable auto-slow mode based on with CPU Data access to AHB. 0 = Run at the programmed CLK_H frequency.
21 CPU_INSTR_ AS_ENABLE	Enable auto-slow mode based on with CPU Instruction access to AHB. 0 = Run at the programmed CLK_H frequency.
20 ASM_ENABLE	Enable CLK_H auto-slow mode. When this is set, then CLK_H will run at the slow rate until one of the fast mode events has occurred.
19 AUTO_CLEAR_ DIV_ENABLE	If this bit is set to one (1'b1), HW_CLKCTRL_HBUS_DIV is cleared to 1 automatically without S/W interaction when wakeup interrupt event happens. This feature is to accelerate the waking up from Wait-For-Interrupt Mode. Note: This bit is not self-cleared. This feature is supposed to used when the clk_h is reduced to very low frequency, for example, 24KHz.
18–16 SLOW_DIV	Slow mode divide ratio. Sets the ratio of CLK_H fast rate to the slow rate. 0x0 **BY1** — Slow mode divide ratio = 1 0x1 **BY2** — Slow mode divide ratio = 2 0x2 **BY4** — Slow mode divide ratio = 4 0x3 **BY8** — Slow mode divide ratio = 8 0x4 **BY16** — Slow mode divide ratio = 16 0x5 **BY32** — Slow mode divide ratio = 32
15–6 RSRVD1	Reserved

HW_CLKCTRL_HBUS field descriptions (continued)

Field	Description
5 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
4–0 DIV	CLK_P-to-CLK_H divide ratio. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.8 APBX Clock Control Register (HW_CLKCTRL_XBUS)

This register controls the clock divider that generates the CLK_X, the clock used by the APBX bus. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_XBUS_WR(BF_CLKCTRL_XBUS_DIV(4)); // set apbx xbus clock to 1/4 the 24.0MHz crystal
 clock frequency
```

Address: HW_CLKCTRL_XBUS – 8004_0000h base + 70h offset = 8004_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUSY	RSRVD1[30:16]														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:12]				AUTO_CLEAR_DIV_ENABLE	DIV_FRAC_EN	DIV									
W					AUTO_CLEAR_DIV_ENABLE	DIV_FRAC_EN	DIV									
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

HW_CLKCTRL_XBUS field descriptions

Field	Description
31 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
30–12 RSRVD1	Always set to zero (0).
11 AUTO_CLEAR_ DIV_ENABLE	If this bit is set to one (1'b1), HW_CLKCTRL_XBUS_DIV is cleared to 1 automaticlly without S/W interaction when wakeup interrupt event happens. This feature is to accelerate the waking up from Wait-For-Interrupt Mode. Note: This bit is not self-cleared. This feature is supposed to used when the clk_x is reduced to very low frequency, for example, 24KHz.

HW_CLKCTRL_XBUS field descriptions (continued)

Field	Description
10 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
9–0 DIV	This field controls the CLK_X divide ratio. CLK_X is sourced from the 24-MHz XTAL through this divider. Do NOT divide by 0.

10.8.9 XTAL Clock Control Register (HW_CLKCTRL_XTAL)

The XTAL control register provides gating control for clocks sourced from the 24-MHz XTAL clock domain.

HW_CLKCTRL_XTAL: 0x080

HW_CLKCTRL_XTAL_SET: 0x084

HW_CLKCTRL_XTAL_CLR: 0x088

HW_CLKCTRL_XTAL_TOG: 0x08C

This register controls various fixed-rate divider clocks working off the 24.0-MHz crystal clock.

EXAMPLE

```
HW_CLKCTRL_XTAL_WR(BF_CLKCTRL_XTAL_UART_CLK_GATE(0)|BF_CLKCTRL_XTAL_DRI_CLK24M_GATE(1));
```

Address: HW_CLKCTRL_XTAL – 8004_0000h base + 80h offset = 8004_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UART_CLK_GATE	RSRVD3	PWM_CLK24M_GATE	RSRVD2		TIMROT_CLK32K_GATE	RSRVD1[25:16]									
W				▨	▨		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:2]														DIV_UART	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_XTAL field descriptions

Field	Description
31 UART_CLK_ GATE	If set to 1, fixed 24-MHz clock for the UART, CLK_UART, is gated off.
30 RSRVD3	Always set to zero (0).
29 PWM_CLK24M_ GATE	If set to 1, fixed 24-MHz clock for the PWM, CLK_PWM24M, is gated off.
28–27 RSRVD2	Always set to zero (0).
26 TIMROT_ CLK32K_GATE	If set to 1, fixed 32-kHz clock for the TIMROT block, CLK_32K, is gated off.
25–2 RSRVD1	Always set to zero (0).
1–0 DIV_UART	Reserved - Always set to one (1)

10.8.10 Synchronous Serial Port0 Clock Control Register (HW_CLKCTRL_SSP0)

This register controls the clock divider that generates the clock for the synchronous serial port (SSP0), CLK_SSP0. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SSP0_WR(BF_CLKCTRL_SSP0_DIV(40));
```

Address: HW_CLKCTRL_SSP0 – 8004_0000h base + 90h offset = 8004_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD2	BUSY	RSRVD1[28:16]												
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:10]						DIV_FRAC_EN	DIV								
W	▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_SSP0 field descriptions

Field	Description
31 CLKGATE	CLK_SSP0 Gate. If set to 1, CLK_SSP0 is gated off. 0: CLK_SSP0 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–10 RSRVD1	Always set to zero (0).
9 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
8–0 DIV	The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

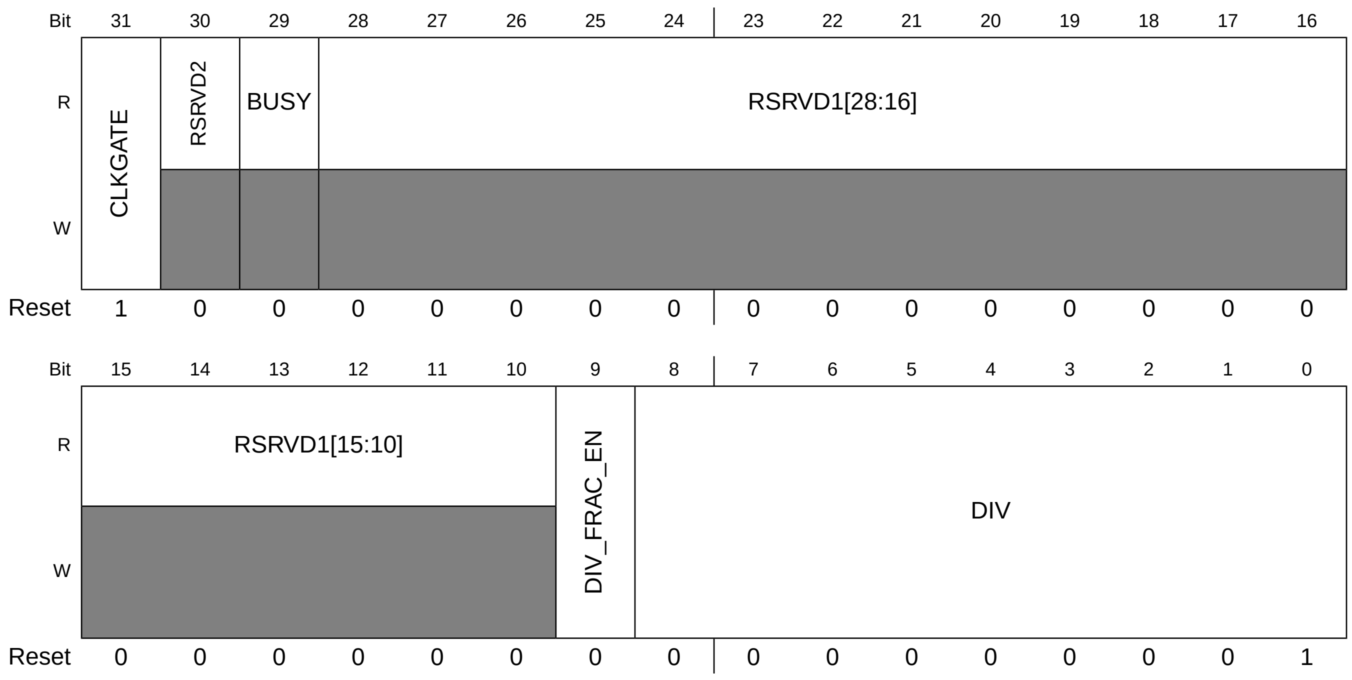
10.8.11 Synchronous Serial Port1 Clock Control Register (HW_CLKCTRL_SSP1)

This register controls the clock divider that generates the clock for the synchronous serial port (SSP1), CLK_SSP1. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SSP1_WR(BF_CLKCTRL_SSP1_DIV(40));
```

Address: HW_CLKCTRL_SSP1 – 8004_0000h base + A0h offset = 8004_00A0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
CLKGATE
RSRVD2
BUSY
RSRVD1[28:16]
Reset 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1[15:10]
DIV_FRAC_EN
DIV
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

HW_CLKCTRL_SSP1 field descriptions

Field	Description
31 CLKGATE	CLK_SSP1 Gate. If set to 1, CLK_SSP1 is gated off. 0: CLK_SSP1 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–10 RSRVD1	Always set to zero (0).
9 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
8–0 DIV	The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.12 Synchronous Serial Port2 Clock Control Register (HW_CLKCTRL_SSP2)

This register controls the clock divider that generates the clock for the synchronous serial port (SSP2), CLK_SSP2. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SSP2_WR(BF_CLKCTRL_SSP2_DIV(40));
```

Address: HW_CLKCTRL_SSP2 – 8004_0000h base + B0h offset = 8004_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD2	BUSY	RSRVD1[28:16]												
W		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:10]						DIV_FRAC_EN	DIV								
W	▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_SSP2 field descriptions

Field	Description
31 CLKGATE	CLK_SSP2 Gate. If set to 1, CLK_SSP2 is gated off. 0: CLK_SSP2 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–10 RSRVD1	Always set to zero (0).
9 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
8–0 DIV	The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

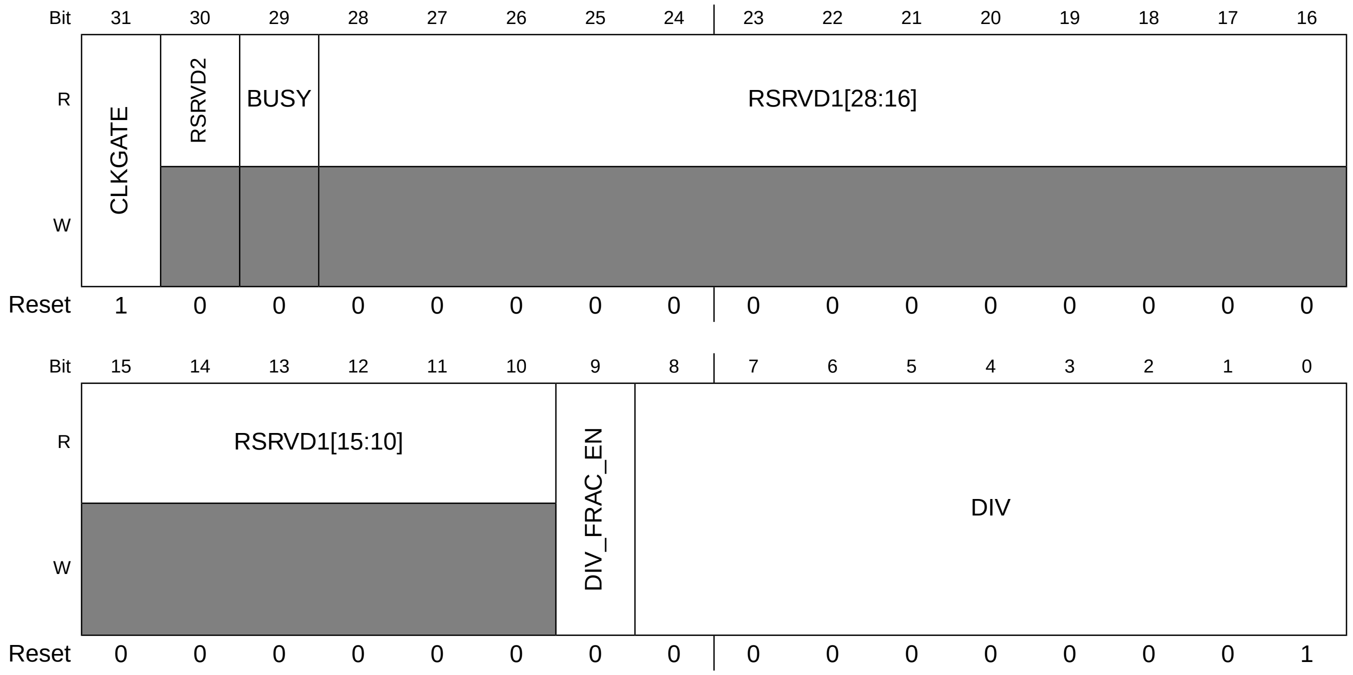
10.8.13 Synchronous Serial Port3 Clock Control Register (HW_CLKCTRL_SSP3)

This register controls the clock divider that generates the clock for the synchronous serial port (SSP3), CLK_SSP3. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SSP3_WR(BF_CLKCTRL_SSP3_DIV(40));
```

Address: HW_CLKCTRL_SSP3 – 8004_0000h base + C0h offset = 8004_00C0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
CLKGATE
RSRVD2
BUSY
RSRVD1[28:16]
Reset 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RSRVD1[15:10]
DIV_FRAC_EN
DIV
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

HW_CLKCTRL_SSP3 field descriptions

Field	Description
31 CLKGATE	CLK_SSP3 Gate. If set to 1, CLK_SSP3 is gated off. 0: CLK_SSP3 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–10 RSRVD1	Always set to zero (0).
9 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.

HW_CLKCTRL_SSP3 field descriptions (continued)

Field	Description
8–0 DIV	The synchronous serial port clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.14 General-Purpose Media Interface Clock Control Register (HW_CLKCTRL_GPMI)

This register controls the divider that generates the General-Purpose Media Interface (GPMI) clock, CLK_GPMI. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_GPMI_WR(BF_CLKCTRL_GPMI_DIV(40));
```

Address: HW_CLKCTRL_GPMI – 8004_0000h base + D0h offset = 8004_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD2	BUSY	RSRVD1[28:16]												
W		▨	▨	▨												
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:11]					DIV_FRAC_EN	DIV									
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_GPMI field descriptions

Field	Description
31 CLKGATE	CLK_GPMI Gate. If set to 1, CLK_GPMI is gated off. 0: CLK_GPMI is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.

HW_CLKCTRL_GPMI field descriptions (continued)

Field	Description
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–11 RSRVD1	Always set to zero (0).
10 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
9–0 DIV	The GPMI clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_io) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.15 SPDIF Clock Control Register (HW_CLKCTRL_SPDIF)

This register controls the clock gate on the SPDIF clock, CLK_PCMSPDIF.

EXAMPLE

```
HW_CLKCTRL_SPDIF_WR(BF_CLKCTRL_SPDIF_CLKGATE(1));
```

Address: HW_CLKCTRL_SPDIF – 8004_0000h base + E0h offset = 8004_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD[30:16]														
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_SPDIF field descriptions

Field	Description
31 CLKGATE	CLK_PCMSPDIF Gate. If set to 1, CLK_PCMSPDIF is gated off. 0: CLK_PCMSPDIF is not gated. When this bit is modified, or when it is high, the SPDIF rate change field should not change its value. The SPDIF rate change field can change ONLY when this clock gate bit field is low.

HW_CLKCTRL_SPDIF field descriptions (continued)

Field	Description
30–0 RSRVD	Always set to zero (0).

10.8.16 EMI Clock Control Register (HW_CLKCTRL_EMI)

This register controls the clock dividers that generate the External Memory Interface (EMI) clock. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_EMI_WR(BF_CLKCTRL_EMI_DIV_XTAL(1));
```

Address: HW_CLKCTRL_EMI – 8004_0000h base + F0h offset = 8004_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	SYNC_MODE_EN	BUSY_REF_XTAL	BUSY_REF_EMI	BUSY_REF_CPU	BUSY_SYNC_MODE	RSRVD3								BUSY_DCC_RESYNC	DCC_RESYNC_ENABLE
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2				DIV_XTAL				RSRVD1		DIV_EMI					
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

HW_CLKCTRL_EMI field descriptions

Field	Description
31 CLKGATE	CLK_EMI crystal divider Gate. If set to 1, the EMI_CLK divider that is sourced by the crystal reference clock, ref_xtal, is gated off. 0: CLK_EMI crystal divider is not gated
30 SYNC_MODE_ EN	If set to 1, EMI_CLK is synchronous with the APBH clock. If set to 0, EMI_CLK is aysnchronous. In synchronous operation, the EMI clock dividers control both EMI_CLK and APBH clock. Both xtal and ref_cpu must be active to switch between asynchronous/synchronous operation.

HW_CLKCTRL_EMI field descriptions (continued)

Field	Description
29 BUSY_REF_ XTAL	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains. This bit is valid when HW_CLKCTRL_EMI_SYNC_MODE_EN = 0.
28 BUSY_REF_EMI	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
27 BUSY_REF_CPU	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains. This bit is valid when HW_CLKCTRL_EMI_SYNC_MODE_EN = 1.
26 BUSY_SYNC_ MODE	This read-only bit field returns a one when there is a change in HW_CLKCTRL_EMI_SYNCE_MODE_EN or when there is a change in HW_CLKCTRL_CLKSEQ_BYPASS_CPU and HW_CLKCTRL_EMI_SYNCE_MODE_EN is set. When this bit returns a one, do not change the CPU or EMI divider values.
25–18 RSRVD3	Always set to zero (0).
17 BUSY_DCC_ RESYNC	Reserved.
16 DCC_RESYNC_ ENABLE	Reserved.
15–12 RSRVD2	Always set to zero (0).
11–8 DIV_XTAL	This field controls the divider connected to the crystal reference clock, ref_xtal, that drives the CLK_EMI domain when bypass IS selected. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
7–6 RSRVD1	Always set to zero (0).
5–0 DIV_EMI	This field controls the divider connected to the ref_emi reference clock that drives the CLK_EMI domain when bypass IS NOT selected. For changes to this field to take effect, the ref_emi reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

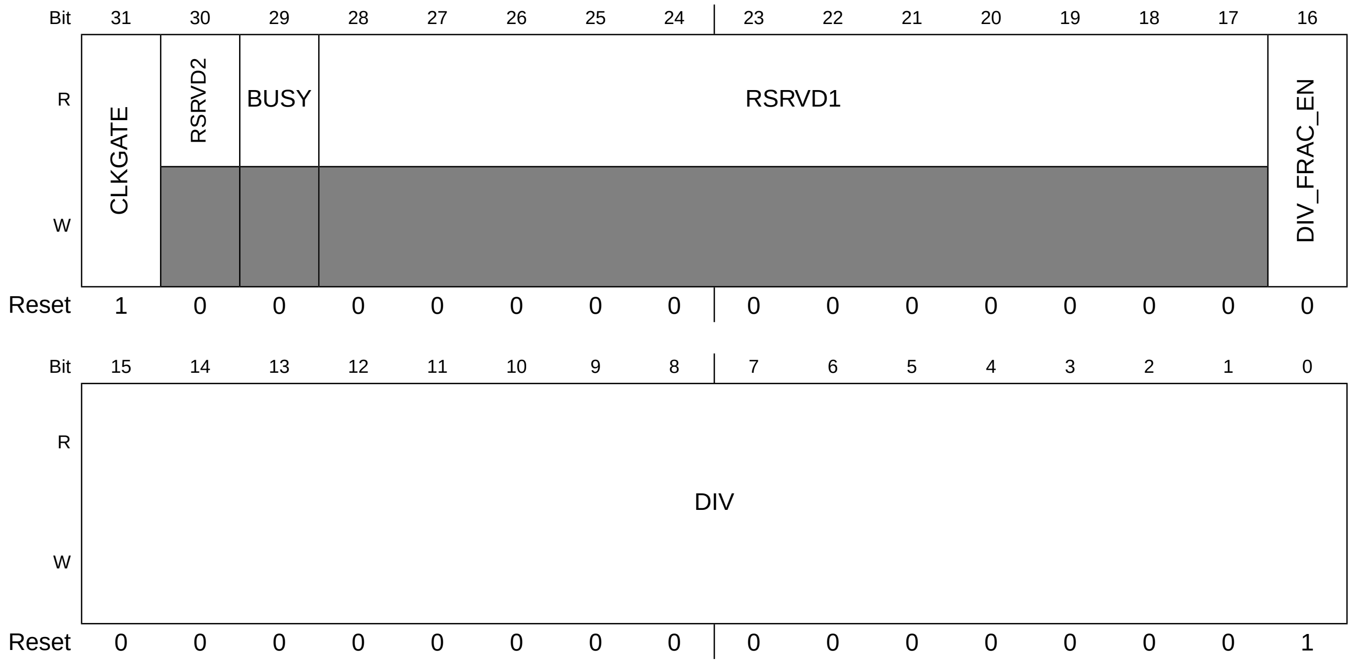
10.8.17 SAIF0 Clock Control Register (HW_CLKCTRL_SAIF0)

This register controls the divider that generates the Serial Audio Interface (SAIF0) clock. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SAIF0_WR(BF_CLKCTRL_SAIF0_DIV(40));
```

Address: HW_CLKCTRL_SAIF0 – 8004_0000h base + 100h offset = 8004_0100h



HW_CLKCTRL_SAIF0 field descriptions

Field	Description
31 CLKGATE	CLK_SAIF0 Gate. If set to 1, CLK_SAIF0 is gated off. 0: CLK_SAIF0 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–17 RSRVD1	Always set to zero (0).
16 DIV_FRAC_EN	Always set to one (1) for functional operation. - Notice this is not the reset value.
15–0 DIV	The SAIF0 clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_pll) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0. ref_pll should be selected for functional operation.

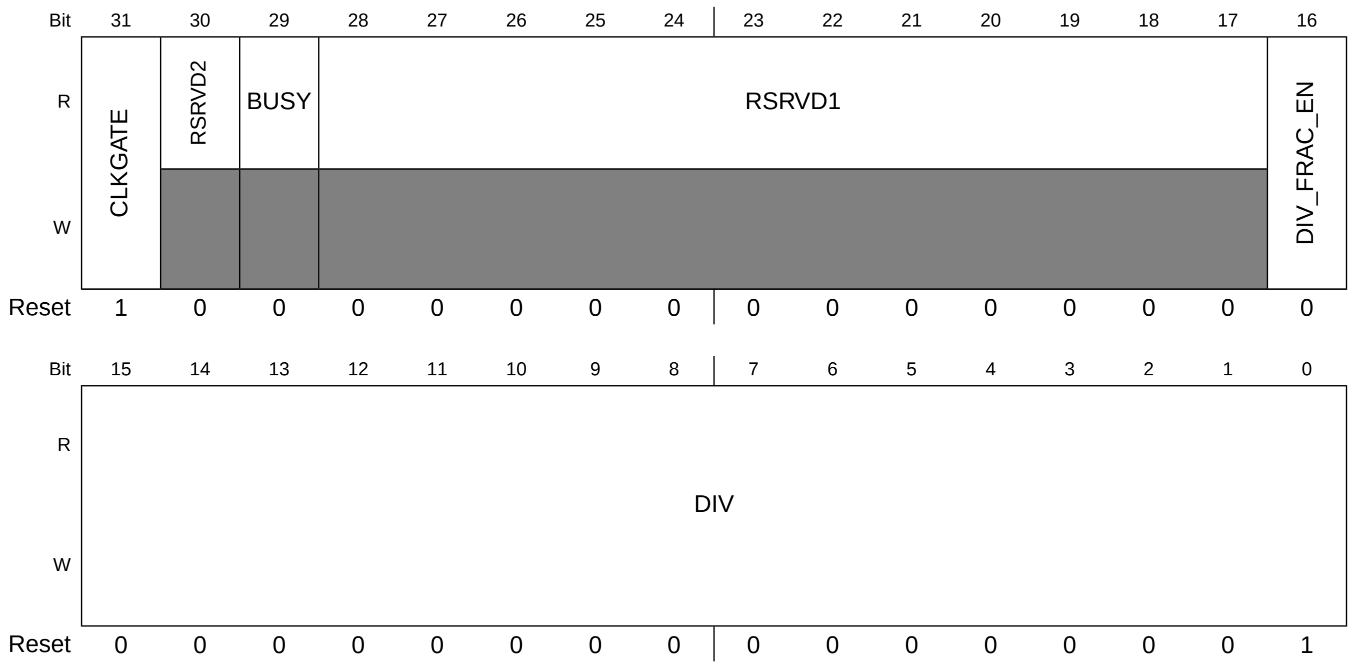
10.8.18 SAIF1 Clock Control Register (HW_CLKCTRL_SAIF1)

This register controls the divider that generates the Serial Audio Interface (SAIF1) clock. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_SAIF1_WR(BF_CLKCTRL_SAIF1_DIV(40));
```

Address: HW_CLKCTRL_SAIF1 – 8004_0000h base + 110h offset = 8004_0110h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
CLKGATE
RSRVD2
BUSY
RSRVD1
DIV_FRAC_EN
Reset 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
DIV
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

HW_CLKCTRL_SAIF1 field descriptions

Field	Description
31 CLKGATE	CLK_SAIF1 Gate. If set to 1, CLK_SAIF1 is gated off. 0: CLK_SAIF1 is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–17 RSRVD1	Always set to zero (0).
16 DIV_FRAC_EN	Always set to one (1) for functional operation - Notice this is not the reset value.
15–0 DIV	The SAIF1 clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_pll) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0. ref_pll should be selected for functional operation.

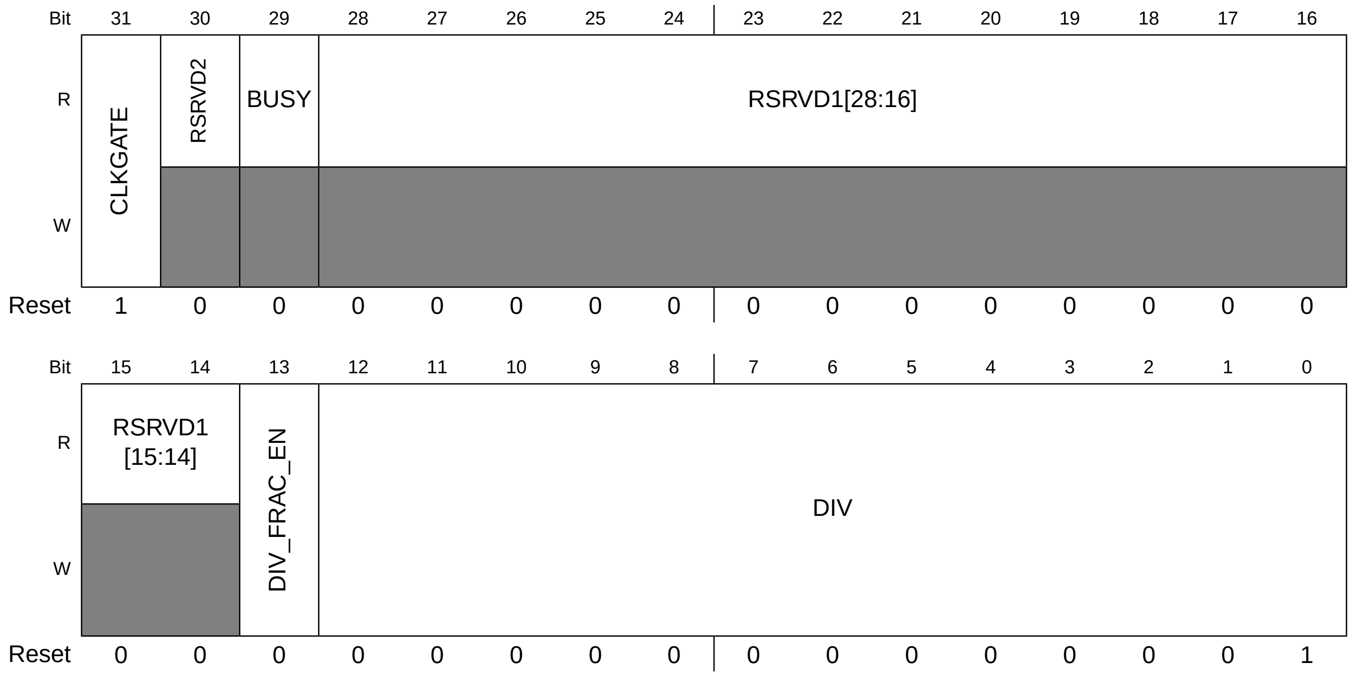
10.8.19 CLK_DIS_LCDIF Clock Control Register (HW_CLKCTRL_DIS_LCDIF)

This register controls the divider that generates the CLK_DIS_LCDIF clock. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_DIS_LCDIF_WR(BF_CLKCTRL_DIS_LCDIF_DIV(40));
```

Address: HW_CLKCTRL_DIS_LCDIF – 8004_0000h base + 120h offset = 8004_0120h



HW_CLKCTRL_DIS_LCDIF field descriptions

Field	Description
31 CLKGATE	CLK_DIS_LCDIF Gate. If set to 1, CLK_DIS_LCDIF is gated off. 0: CLK_DIS_LCDIF is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–14 RSRVD1	Always set to zero (0).
13 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.

HW_CLKCTRL_DIS_LCDIF field descriptions (continued)

Field	Description
12–0 DIV	The DIS_LCDIF clock frequency is determined by dividing the reference clock, ref_xtal or ref_pix, by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0. Do not divide by more than 255.

10.8.20 ETM Clock Control Register (HW_CLKCTRL_ETM)

This register controls the clock divider that generates the clock for the ETM, CLK_ETM. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_ETM_WR(BF_CLKCTRL_ETM_DIV(4));
```

Address: HW_CLKCTRL_ETM – 8004_0000h base + 130h offset = 8004_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATE	RSRVD2	BUSY	RSRVD1[28:16]												
W		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:8]								DIV_FRAC_EN	DIV						
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_CLKCTRL_ETM field descriptions

Field	Description
31 CLKGATE	CLK_ETM Gate. If set to 1, CLK_ETM is gated off. 0: CLK_ETM is not gated. When this bit is modified, or when it is high, the DIV field should not change its value. The DIV field can change ONLY when this clock gate bit field is low.
30 RSRVD2	Always set to zero (0).

HW_CLKCTRL_ETM field descriptions (continued)

Field	Description
29 BUSY	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
28–8 RSRVD1	Always set to zero (0).
7 DIV_FRAC_EN	1 = Enable fractional divide. 0 = Enable integer divide.
6–0 DIV	The ETM clock frequency is determined by dividing the selected reference clock (ref_xtal or ref_cpu) by the value in this bit field. This field can be programmed with a new value only when CLKGATE = 0. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.

10.8.21 ENET Clock Control Register (HW_CLKCTRL_ENET)

This register provides control for ETHERNET clock generation

EXAMPLE

```
HW_CLKCTRL_ENET_WR(BF_CLKCTRL_ENET_SLEEP(1));
```

Address: HW_CLKCTRL_ENET – 8004_0000h base + 140h offset = 8004_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SLEEP	DISABLE	STATUS	RSRVD1	BUSY_TIME	DIV_TIME						TIME_SEL		CLK_OUT_EN	RESET_BY_SW_CHIP	RESET_BY_SW
W			▨	▨	▨											
Reset	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_ENET field descriptions

Field	Description
31 SLEEP	CLOCK Gate. If set to 1, put Ethernet block in sleep mode. CLK_H_MAC0(1), CLK_H_MAC0(1)_S, and CLK_ENET0(1)_TX are gated off. Ethernet can be wakeup remotely in sleep mode 0: Resume Ethernet block to normal operation. CLK_H_MAC0(1), CLK_H_MAC0(1)_S, and CLK_ENET_TX are not gated. .
30 DISABLE	This bit is used to gate off all Ethernet clocks when Ethernet is disabled in some use case. If set to 1, gate off all of the Ethernet clocks. Ethernet can not be wakeup remotely when Ethernet is disabled. 0: Do not gate off Ethernet's clocks.
29 STATUS	This read-only bit indicates the status of Ehternet module. 1'b1: Ethernet is in SLEEP or DISABLE mode. 1'b0: Ethernet is in NORMAL mode.
28 RSRVD1	Always set to zero (0).
27 BUSY_TIME	This read-only bit field returns a one when the clock divider is busy transfering a new divider value across clock domains.
26–21 DIV_TIME	This field controls the divider that drives the CLK_ENET_TIME (1588 timer) domain. For changes to this field to take effect, the reference clock must be running. NOTE: The divider is set to divide by 1 at power-on reset. Do NOT divide by 0.
20–19 TIME_SEL	This field selects clock that drives the Ethernet 1588 timer between the xtal, ref_pll and enet_rmii_clk_in sources. 0x0 **XTAL** — select xtal source 0x1 **PLL** — select ref_pll source 0x2 **RMII_CLK** — select enet_rmii_clk_in source which is from PAD 0x3 **UNDEFINED** — Undefined
18 CLK_OUT_EN	This bit controls the ENET_CLK PAD direction. 1: Enable the output; 0: Disable the output. NOTE: This bit must be configured before ENET PLL is enabled.
17 RESET_BY_SW_CHIP	Setting this bit to a logic one will enable the function that ENET SWITCH can be reset by SW chip reset (HW_CLKCTRL_RESET.CHIP, or watchdog_reset when HW_CLKCTRL_RESET.WDOG_POR_DISABLE is set to 1'b1). This bit's reset value reflects OTP fuse ENET_SWITCH_RESET_BY_SW_CHIP.
16 RESET_BY_SW	Setting this bit to a logic one will enable the function that ENET SWITCH can be reset by all software reset (Either HW_CLKCTRL_RESET.CHIP or HW_CLKCTRL_RESET.DIG). This bit's reset value reflects OTP fuse ENET_SWITCH_RESET_BY_SW.
15–0 RSRVD0	Always set to zero (0).

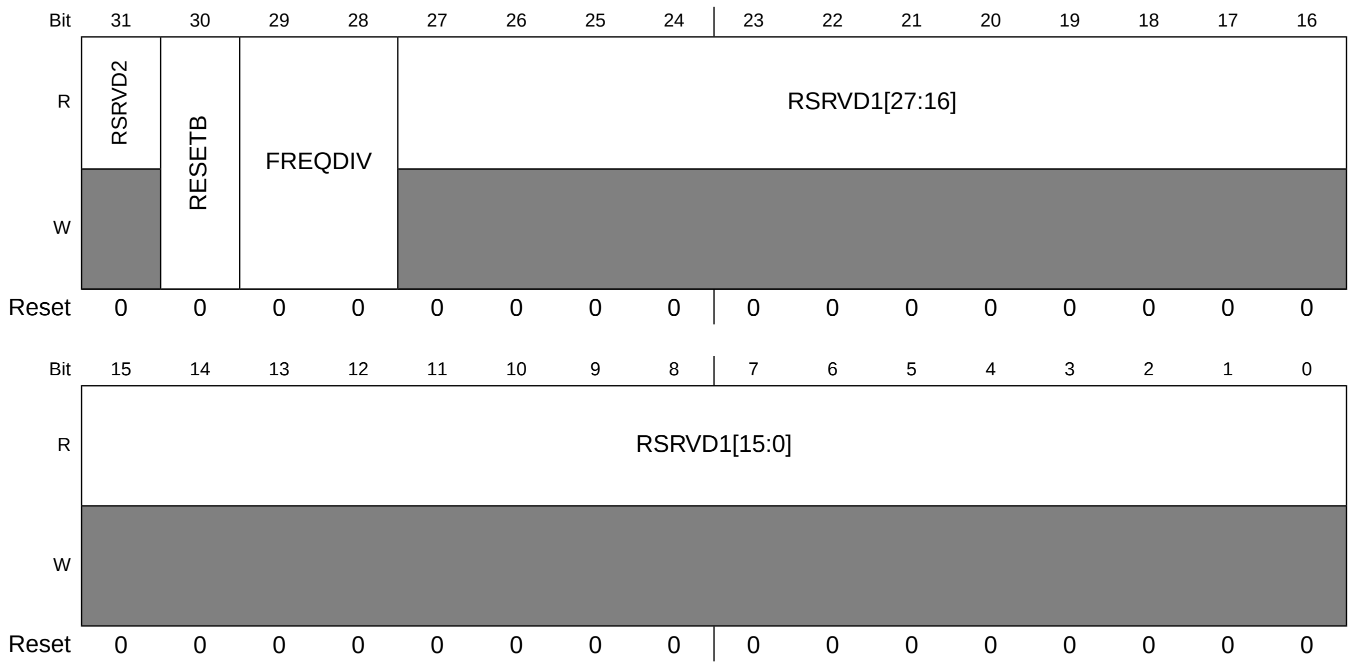
10.8.22 HSADC Clock Control Register (HW_CLKCTRL_HSADC)

This register controls the clock dividers that generate the High Speed ADC (HSADC) clock. Note: Do not write register space when busy bit(s) are high.

EXAMPLE

```
HW_CLKCTRL_HSADC_WR(BF_CLKCTRL_HSADC_CLKGATE(1));
```

Address: HW_CLKCTRL_HSADC – 8004_0000h base + 150h offset = 8004_0150h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD2
RESETB
FREQDIV
RSRVD1[27:16]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1[15:0]
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_CLKCTRL_HSADC field descriptions

Field	Description
31 RSRVD2	Always set to zero (0).
30 RESETB	0: Reset the HSADC Divider. 1: Divider normal work.
29–28 FREQDIV	This field selects HSADC Divider's divide factor. 00: Divide by 9; 01: Divide by 18; 10: Divide by 36; 11: Divide by 72
27–0 RSRVD1	Always set to zero (0).

10.8.23 FLEXCAN Clock Control Register (HW_CLKCTRL_FLEXCAN)

The FLEXCAN control register provides control for Flexcan0 and Flexcan1 clock generation.

EXAMPLE

```
HW_CLKCTRL_FLEXCAN_WR(BF_CLKCTRL_FLEXCAN_STOP_CAN0(1));
```

Address: HW_CLKCTRL_FLEXCAN – 8004_0000h base + 160h offset = 8004_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2	STOP_CAN0	CAN0_STATUS	STOP_CAN1	CAN1_STATUS	RSRVD1[26:16]										
W																
Reset	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_FLEXCAN field descriptions

Field	Description
31 RSRVD2	Always set to zero (0).
30 STOP_CAN0	If this bit is set to one (1'b1), the FLEXCAN0 will be stopped and all the clocks to FLEXCAN0 are gated off.
29 CAN0_STATUS	This read-only bit indicates FLEXCAN0 status. 1'b1: FLEXCAN0 is in STOP mode. 1'b0: FLEXCAN0 is in NORMAL Mode.
28 STOP_CAN1	If this bit is set to one (1'b1), the FLEXCAN1 will be stopped and all the clocks to FLEXCAN1 are gated off.
27 CAN1_STATUS	This read-only bit indicates FLEXCAN1 status. 1'b1: FLEXCAN1 is in STOP mode. 1'b0: FLEXCAN1 is in NORMAL Mode.
26–0 RSRVD1	Always set to zero (0).

10.8.24 Fractional Clock Control Register 0 (HW_CLKCTRL_FRAC0)

The FRAC0 control register provides control for PFD clock generation.

HW_CLKCTRL_FRAC0: 0x1B0

HW_CLKCTRL_FRAC0_SET: 0x1B4

HW_CLKCTRL_FRAC0_CLR: 0x1B8

HW_CLKCTRL_FRAC0_TOG: 0x1BC

This register controls the 9-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers. NOTE: This register can only be addressed by byte instructions. Addressing word or half-word are not allowed.

EXAMPLE

```
*((u8 *)(HW_CLKCTRL_FRAC0_ADDR + 1)) = 30;
```

Address: HW_CLKCTRL_FRAC0 – 8004_0000h base + 1B0h offset = 8004_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CLKGATEIO0	IO0_STABLE	IO0FRAC						CLKGATEIO1	IO1_STABLE	IO1FRAC					
W		▨								▨						
Reset	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CLKGATEEMI	EMI_STABLE	EMIFRAC						CLKGATECPU	CPU_STABLE	CPUFRAC					
W		▨								▨						
Reset	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1	0

HW_CLKCTRL_FRAC0 field descriptions

Field	Description
31 CLKGATEIO0	IO0 Clock Gate. If set to 1, the IO0 fractional divider clock (reference PLL0 ref_io0) is off (power savings). 0: IO0 fractional divider clock is enabled.
30 IO0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 IO0FRAC	This field controls the IO0 clocks fractional divider. The resulting frequency shall be 480 * (18/IO0FRAC) where IO0FRAC = 18-35.
23 CLKGATEIO1	IO1 Clock Gate. If set to 1, the IO1 fractional divider clock (reference PLL0 ref_io1) is off (power savings). 0: IO1 fractional divider clock is enabled.
22 IO1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.

HW_CLKCTRL_FRAC0 field descriptions (continued)

Field	Description
21–16 IO1FRAC	This field controls the IO1 clocks fractional divider. The resulting frequency shall be 480 * (18/IO1FRAC) where IO1FRAC = 18-35.
15 CLKGATEEMI	EMI Clock Gate. If set to 1, the EMI fractional divider clock (reference PLL0 ref_emi) is off (power savings). 0: EMI fractional divider clock is enabled.
14 EMI_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. This value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 EMIFRAC	This field controls the EMI clock fractional divider. The resulting frequency shall be 480 * (18/EMIFRAC) where EMIFRAC = 18-35.
7 CLKGATECPU	CPU Clock Gate. If set to 1, the CPU fractional divider clock (reference PLL0 ref_cpu) is off (power savings). 0: CPU fractional divider clock is enabled.
6 CPU_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divide should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5–0 CPUFRAC	This field controls the CPU clock fractional divider. The resulting frequency shall be 480 * (18/CPUFRAC) where CPUFRAC = 18-35.

10.8.25 Fractional Clock Control Register 1 (HW_CLKCTRL_FRAC1)

The FRAC1 control register provides control for PFD clock generation.

HW_CLKCTRL_FRAC1: 0x1C0

HW_CLKCTRL_FRAC1_SET: 0x1C4

HW_CLKCTRL_FRAC1_CLR: 0x1C8

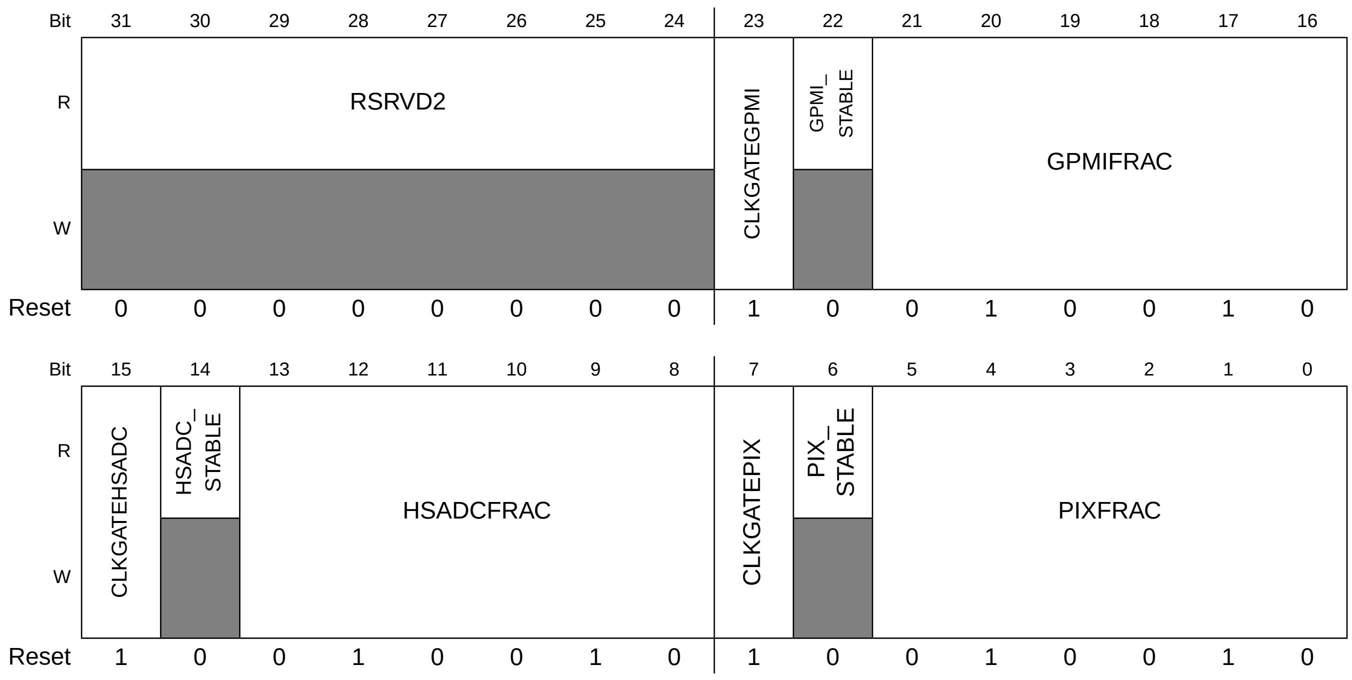
HW_CLKCTRL_FRAC1_TOG: 0x1CC

This register controls the 9-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers. NOTE: This register can only be addressed by byte instructions. Addressing word or half-word are not allowed.

EXAMPLE

```
*((u8 *)(HW_CLKCTRL_FRAC1_ADDR + 1)) = 30;
```

Address: HW_CLKCTRL_FRAC1 – 8004_0000h base + 1C0h offset = 8004_01C0h



HW_CLKCTRL_FRAC1 field descriptions

Field	Description
31–24 RSRVD2	Always set to zero (0).
23 CLKGATEGPMI	GPMI Clock Gate. If set to 1, the GPMI fractional divider clock (reference PLL0 ref_gpmi) is off (power savings). 0: GPMI fractional divider clock is enabled.
22 GPMI_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 GPMIFRAC	This field controls the GPMI clock fractional divider. The resulting frequency shall be 480 * (18/GPMIFRAC) where GPMIFRAC = 18-35.
15 CLKGATEHSADC	HSADC Clock Gate. If set to 1, the HSADC fractional divider clock (reference PLL0 ref_adc) is off (power savings). 0: HSADC fractional divider clock is enabled.
14 HSADC_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 HSADCFRAC	This field controls the HSADC clock fractional divider. The resulting frequency shall be 480 * (18/HSADCFRAC) where HSADCFRAC = 18-35.
7 CLKGATEPIX	PIX Clock Gate. If set to 1, the PIX fractional divider clock (reference PLL0 ref_pix) is off (power savings). 0: PIX fractional divider clock is enabled.

HW_CLKCTRL_FRAC1 field descriptions (continued)

Field	Description
6 PIX_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
5–0 PIXFRAC	This field controls the PIX clock fractional divider. The resulting frequency shall be 480 * (18/PIXFRAC) where PIXFRAC = 18-35.

10.8.26 Clock Frequency Sequence Control Register (HW_CLKCTRL_CLKSEQ)

The CLKSEQ control register provides control for switching between XTAL and PLL clock generation.

HW_CLKCTRL_CLKSEQ: 0x1D0

HW_CLKCTRL_CLKSEQ_SET: 0x1D4

HW_CLKCTRL_CLKSEQ_CLR: 0x1D8

HW_CLKCTRL_CLKSEQ_TOG: 0x1DC

This register controls the selection of clock sources (ref_xtal or ref_*) for various clock dividers.

EXAMPLE

```
HW_CLKCTRL_CLKSEQ_WR(BF_CLKCTRL_CLKSEQ_BYPASS_SAIF0(1));
```

Address: HW_CLKCTRL_CLKSEQ – 8004_0000h base + 1D0h offset = 8004_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0													BYPASS_CPU	RSRVD1 [17:16]	
W																
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1 [15:15]	BYPASS_ DIS_LCDIF	RSRVD2					BYPASS_ ETM	BYPASS_ EMI	BYPASS_ SSP3	BYPASS_ SSP2	BYPASS_ SSP1	BYPASS_ SSP0	BYPASS_ GPMI	BYPASS_ SAIF1	BYPASS_ SAIF0
W	▨		▨	▨	▨	▨	▨									
Reset	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

HW_CLKCTRL_CLKSEQ field descriptions

Field	Description
31–19 RSRVD0	Always set to zero (0).
18 BYPASS_CPU	CPU bypsss select. 1 = Select ref_xtal path to generate the CPU clock domain. 0 = Select ref_cpu path to generate the CPU clock domain. PLL0 and 9-phase fractional divider must already be configured when changing from bypass mode.
17–15 RSRVD1	Always set to zero (0).
14 BYPASS_DIS_ LCDIF	LCDIF bypsss select. 1 = Select ref_xtal path to generate the LCDIF clock domain. 0 = Select ref_pix path to generate the LCDIF clock domain. PLL0 and 9-phase fractional divider must already be configured when changing from bypass mode. 0x1 **BYPASS** — select xtal source, bypass mode 0x0 **PFD** — select PFD, ref_pix
13–9 RSRVD2	Always set to zero (0).
8 BYPASS_ETM	ETM bypass select. 1 = Select ref_xtal path to generate the ETM clock domain. 0 = Select ref_cpu path to generate the ETM clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
7 BYPASS_EMI	EMI bypass select. 1 = Select ref_xtal path to generate the EMI clock domain. 0 = Select ref_emi path to generate the EMI clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
6 BYPASS_SSP3	SSP3 bypass select. 1 = Select ref_xtal path to generate the SSP3 clock domain. 0 = Select ref_io1 path to generate the SSP3 clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
5 BYPASS_SSP2	SSP2 bypass select. 1 = Select ref_xtal path to generate the SSP2 clock domain. 0 = Select ref_io1 path to generate the SSP2 clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
4 BYPASS_SSP1	SSP1 bypass select. 1 = Select ref_xtal path to generate the SSP1 clock domain. 0 = Select ref_io0 path to generate the SSP1 clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
3 BYPASS_SSP0	SSP0 bypass select. 1 = Select ref_xtal path to generate the SSP0 clock domain. 0 = Select ref_io0 path to generate the SSP0 clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.
2 BYPASS_GPMI	GPMI bypass select. 1 = Select ref_xtal path to generate the GPMI clock domain. 0 = Select ref_gpmi path to generate the GPMI clock domain. PLL0 and 9-phase fractional divider must already be configured when this bit is cleared.

HW_CLKCTRL_CLKSEQ field descriptions (continued)

Field	Description
1 BYPASS_SAIF1	Always set to zero (0) for functional operation. Bypass is only used for POR and SAIF1 divider configuration.
0 BYPASS_SAIF0	Always set to zero (0) for functional operation Bypass is only used for POR and SAIF0 divider configuration.

10.8.27 System Reset Control Register (HW_CLKCTRL_RESET)

The RESET control register provides control for software reset and enable/disable reset control in analog.

EXAMPLE

```
HW_CLKCTRL_RESET_WR(BF_CLKCTRL_RESET_ALL(1));
```

Address: HW_CLKCTRL_RESET – 8004_0000h base + 1E0h offset = 8004_01E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD[15:6]										WDOG_POR_DISABLE	EXTERNAL_RESET_ENABLE	THERMAL_RESET_ENABLE	THERMAL_RESET_DEFAULT	CHIP	DIG
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

HW_CLKCTRL_RESET field descriptions

Field	Description
31–6 RSRVD	Always set to zero (0).
5 WDOG_POR_ DISABLE	Analog Reset Control. Setting this bit to a logic one will disable the function that watchdog perform a POR. Then watchdog will function same with software reset HW_CLKCTRL_RESET.CHIP. Setting this bit to a logic zero, watchdog_reset will trigger a POR. This bit's reset value reflects OTP fuse WDOG_PERFORM_POR.
4 EXTERNAL_ RESET_ENABLE	Analog Reset Control. Setting this bit to a logic one will enable the external pin reset control logic.

HW_CLKCTRL_RESET field descriptions (continued)

Field	Description
3 THERMAL_ RESET_ENABLE	Analog Reset Control. Setting this bit to a logic one will enable the thermal reset control logic.
2 THERMAL_ RESET_ DEFAULT	Reserved.
1 CHIP	Software reset. Setting this bit to a logic one will reset the ENTIRE chip, no exceptions. This bit will also be reset after the full chip reset cycle completes.
0 DIG	Software reset. Setting this bit to a logic one will reset the digital sections of the chip. The DCDC and power module will not be reset. This bit will also be reset after the reset cycle completes.

10.8.28 ClkCtrl Status (HW_CLKCTRL_STATUS)

The STATUS control register provides read only status of the CPU frequecy limits.

EXAMPLE

```
HW_CLKCTRL_STATUS_RD(BF_CLKCTRL_STATUS_CPU_LIMIT());
```

Address: HW_CLKCTRL_STATUS – 8004_0000h base + 1F0h offset = 8004_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CPU_LIMIT		RSRVD[29:16]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_STATUS field descriptions

Field	Description
31–30 CPU_LIMIT	CPU Limiting. 00: full cpu frequency, 01: limit cpu frequency to 455 MHz, 10: limit cpu frequency to 411 MHz, 11: limit cpu frequency to 360 MHz
29–0 RSRVD	Always set to zero (0).

10.8.29 ClkCtrl Version (HW_CLKCTRL_VERSION)

This register indicates the RTL version in use.

EXAMPLE

```
HW_CLKCTRL_VERSION_RD(BF_CLKCTRL_VERSION_MAJOR());
```

Address: HW_CLKCTRL_VERSION – 8004_0000h base + 200h offset = 8004_0200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CLKCTRL_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 11
Power Supply

11.1 Overview

This chapter describes the power supply subsystem provided on the i.MX28. It includes sections on the DC-DC converter, linear regulators, PSWITCH pin functions, battery monitor and charger, silicon speed sensor, thermal-overload protection circuit, bandgap reference and programmable registers.

The i.MX28 integrates a comprehensive power supply subsystem, including the following features:

- One integrated DC-DC converter that supports Li-Ion batteries.
- Four linear regulators directly power the supply rails from 5 V.
- Linear battery charger for Li-Ion cells.
- Battery voltage and brownout detection monitoring for VDDD, VDDA, VDDIO, VDDMEM, VDD4P2 and 5V supplies.
- Integrated current limiter from 5 V power source.
- Reset controller.
- System monitors for temperature and speed.
- Generates USB-Host 5V from Li-Ion battery (using PWM).
- Support for on-the-fly transitioning between 5V and battery power.
- VDD4P2, a nominal 4.2 V supply, is available when the i.MX28 is connected to a 5 V source and allows the DCDC to run from a 5 V source with a depleted battery-integrated current limiter from 5 V power source.
- The 4.2 V regulated output also allows for programmable current limits:
    a. Battery Charge current + DCDC input current < the 5 V current limit.

b. DCDC input current (which ultimately provides current to the on-chip and off-chip loads) as the priority and battery charge current will be automatically reduced if the 5 V current limit is reached.

The i.MX28 power supply is designed to offer maximum flexibility and performance, while minimizing external component requirements. Figure 11-1 shows a functional block diagram of the power supply components including switching converters, five regulators, battery charge support, as well as battery monitoring, supply brownout detection, and silicon process/temperature sensors. This figure can be used to understand which register and status bits relate to which subsystems, but it is not intended to be a complete architecture description.

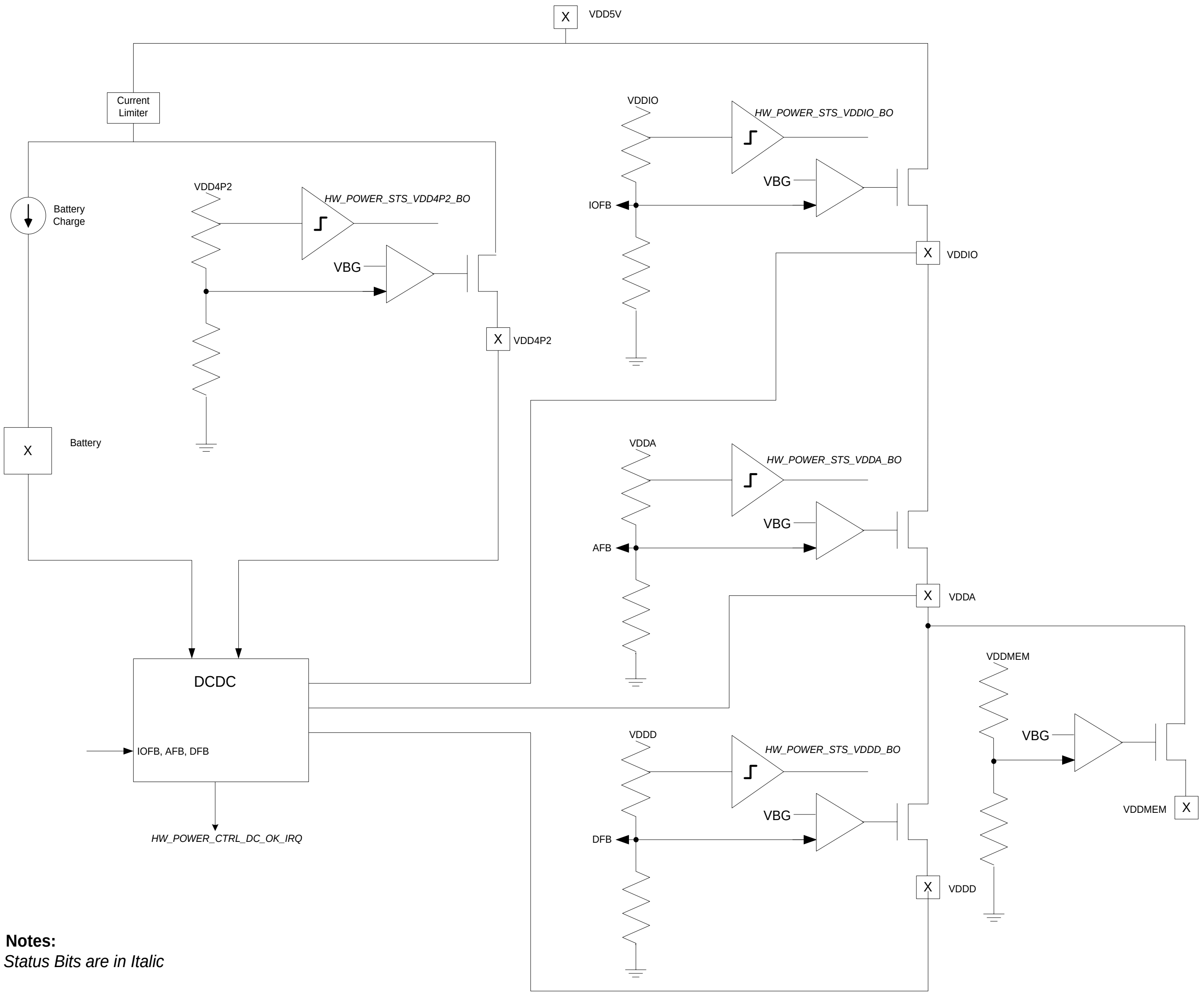


Figure 11-1. Power Supply Block Diagram

11.2 DC-DC Converters

The DC-DC converters efficiently scale battery voltage (or a regulated 4.2 V derived from a 5 V source) to the required supply voltages. The DC-DC converters include several advanced features:

- Single inductor architecture
- Programmable output voltages
- Programmable brownout detection thresholds
- Pulse frequency modulation (PFM) mode for low-current load operation

11.2.1 DC-DC Operation

The i.MX28 DC-DC converter enables a low-power system and features programmable output voltages and control modes. Most products adjust VDDD dynamically to provide the minimum voltage required for proper system operation. VDDIO and VDDA are typically set once during system initialization and not changed during operation.

11.2.1.1 Brownout/Error Detection

The power subsystem has several mechanisms active by default that safely return the device to off state if any one of the following errors or brownouts occur:

- The crystal oscillator frequency is detected below a certain threshold—This threshold is process-and voltage-sensitive, but will always be between 100 KHz and 2 MHz. This feature can be disabled in DISABLE_XTALOK field in HW_RTC_PERSISTENT0 register.
- The battery voltage falls below the battery brownout level (field BRWNOUT_LVL in HW_POWER_BATTMONITOR)—This feature is disabled by clearing PWDN_BATTBRNOUT in the same register.
- 5 V is detected, then removed—This feature is disabled by clearing HW_POWER_5VCTRL_PWDN_5VBRNOUT.

All three mechanisms are active by default to ensure that the device always has a valid transition to a known state in case the power source is unexpectedly removed before the software has completed the system configuration. Software will typically disable the functionality of PWDN_5VBRNOUT and PWDN_BATTBRNOUT after system configuration is complete, as shown in Figure 11-2. System configuration generally includes setting up brownout detection thresholds on the supply voltages, battery, and so on to obtain the desired system operation as the battery or power source is depleted or removed.

Typically, each output target voltage is set to some voltage margin above the minimum operating level through the TRG field in HW_POWER_VDDDCTRL, HW_POWER_VDDACTRL, HW_POWER_VDDIOCTRL, and HW_POWER_VDDMEMCTRL registers. The brownout detection threshold is also set through the BO_OFFSET field in same three registers. The BO_OFFSET field determines how far the brownout voltage is below the output target voltage for each supply and might typically be set 75–100 mV below the target voltage. If the voltage drops to the brownout detector's level, then it optionally triggers a CPU Fast Interrupt (FIQ). The CPU can then alleviate the problem and/or shut down the system elegantly. See i.MX28 Datasheet for suggested voltage settings for the supply and brownout targets for different operating frequencies.

To eliminate false detection, the brownout circuit filters transient noise above 1 MHz. Any system with an i.MX28 should include at least 33 µF of decoupling capacitance on VDDIO, VDDA and VDDD power rails. The capacitors should be arranged to filter supply noise in 1 MHz and higher frequencies.

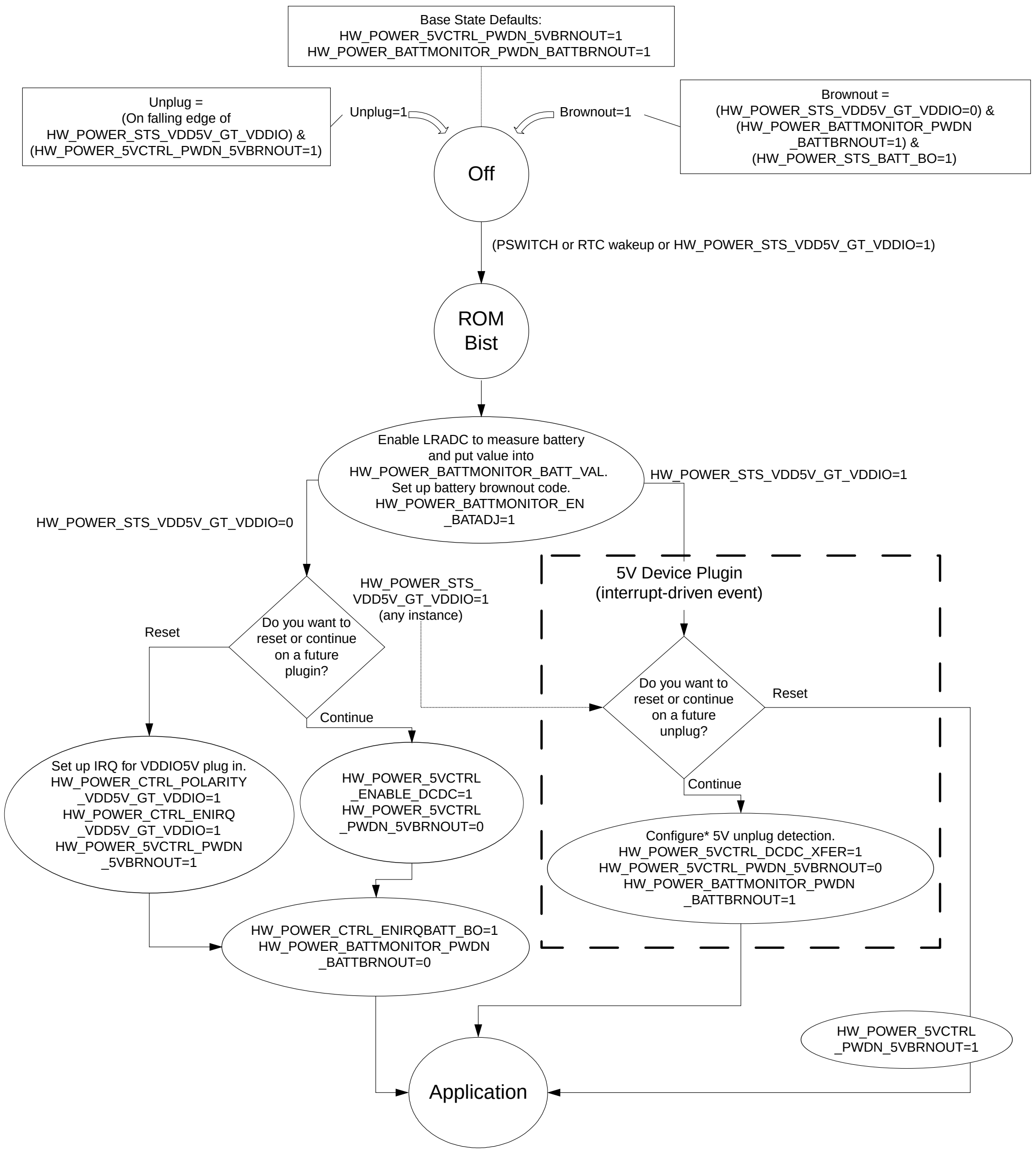


Figure 11-2. Brownout Detection Flowchart

11.2.1.2 DC-DC Extended Battery Life Features

The DC-DC converter has several other power-reducing programmable modes that are useful for maximizing battery life:

- **Li-Ion Buck/Boost**—The Li-Ion battery configuration supports buck/boost operation, which means that a VDDIO voltage can be supported which is higher than the input Li-Ion battery voltage. This is important to maximize the battery life in all applications, but is crucial in hard drives that have large transient current requirements.

- **Transient Loading Optimizations**—Several new incremental improvements have been made to the control architecture of the switching converters. At this time, Freescale recommends setting the following bits through software in HW_POWER_LOOPCTRL to obtain maximum efficiency and minimum supply ripple: TOGGLE_DIF, EN_CM_HYST and EN_RCSCALE=0x1. Also, set HW_POWER_BATTMONITOR_EN_BATADJ. The complete settings to optimize transient loading will be dependent on the application, component selection and board layout.
- **Pulse Frequency Modulation (PFM)**—PFM, also known as pulse-skipping mode, is used to reduce power consumed by the DC-DC converter when the voltage outputs are lightly-loaded at a cost of higher transient noise. The DC-DC converter can be separately placed in PFM mode through the EN_DC_PFM bit in HW_POWER_MINPWR.
- **DC-DC Switching Frequency**—The standard DC-DC switching frequency is 1.5 MHz, which provides a good mix of efficiency and power output. The frequency can be reduced to 750 KHz to reduce operating current in some light load situations through DC_HALFCLK in HW_POWER_MINPWR. The DC-DC converter can also be programmed to a frequency that is based on PLL using the SEL_PLLCLK and FREQSEL fields in the HW_POWER_MISC register.
- **DC-DC Converter Power Down**—If the system is to operate from linear regulators or an external power supply, then the internal DC-DC converters can be powered down through DC_STOPCLK in HW_POWER_MINPWR. This bit is not intended to power down the whole system. Use the HW_POWER_RESET bits to power the system off.

The DC-DC converter can be programmed to run off of the battery or the 4.2 V regulated voltage supply. It can be set up to opportunistically draw power from either source. This allows the system to draw up to the USB current limit from the 5 V supply and take the balance of the required power from the battery.

11.3 Linear Regulators

The i.MX28 integrates four linear regulators that can be used to directly power the supply rails when 5 V is present. All of these regulators have an output impedance of approximately one-quarter ohm. This means that the measured output voltage will be slightly dependent on the current consumed on each supply. Architecturally, one regulator generates VDDIO from the VDD5V pin, one generates VDDA from VDDIO, one generates VDDMEM from VDDA, and the other generates VDDD from the VDDA supply. Therefore, all of the current is supplied by the VDD5V->VDDIO regulator.

In addition, the i.MX28 integrates a 4.2 V regulator which enables the DC-DC converter to run off the power supplied by 5 V. This allows the DC-DC converter operation when 5 V is present and the battery is exhausted. In addition, the path can provide better power consumption as compared to the other linear regulator generated supplies. A current limiter is implemented in a series with the 4.2 V regulator and the battery supply. The intent is to limit the Ibattery plus I4P2 to be less than Ilimit. The 4.2 V regulator has priority on the current and will dynamically reduce the battery charge current in order to preserve the Ilimit and the regulated 4.2 V.

It should also be noted that the VDD5V voltage can dynamically change as the product is plugged into a USB port or other 5-V supply. The i.MX28 is programmable to provide a variety of behaviors when the VDD5V supply becomes valid or invalid, as well as to support operation through USB or external power.

11.3.1 USB Compliance Features

Upon connection of 5 V to the powered-down device, the linear regulators will automatically power up the device. To meet USB inrush specifications, linear regulators have a current limit which is <100 mA, nominally 50 mA, and is active by default. This current limit is disabled in hardware after power-up, but can be re-enabled through the HW_POWER_5VCTRL_ENABLE_ILIMIT by software. System designers must understand that the current inrush limit during 5 V power-up places restrictions on application current consumption until it is disabled. Specifically, after connection to 5V, if the system draws more current than the current limit allows, the startup sequence does not complete and the ROM code does not execute.

Further, USB Host implies that B-devices must draw very little current from 5 V. This requirement can be met by setting HW_POWER_5VCTRL_ILIMIT_EQ_ZERO when the Host application is active. The comparators required for Host capability can be enabled by HW_POWER_5VCTRL_PWRUP_VBUS_CMPS. It is also possible to change the threshold of the Vbus valid comparator using HW_POWER_5VCTRL_VBUSVALID_TRSH.

If very low power operation is required, as in USB suspend, then the circuits required to elegantly switch to the DC-DC converter may have to be powered off. In those cases, the system must fully power down after VDD5V becomes invalid. It can auto restart with the DC-DC converter, if HW_RTC_PERSISTENT0_AUTO_RESTART is set.

11.3.2 5V to Battery Power Interaction

The i.MX28 supports several different options related to the interaction of the switching converters with the linear regulators. The two primary options are a reset on 5V insertion/removal or a handoff to the DC-DC converters that is invisible to the end-user of the application. Figure 11-3 includes these two options as the two system architecture decision boxes.

11.3.2.1 Battery Power to 5-V Power

By default, the DC-DC converter turns off when VDD5V becomes valid and the system does not reset. If the system is operating from the DC-DC converter and is using more current than the linear regulators can supply, then the VDDD, VDDA, and VDDIO rails will droop when 5V is attached and the system may brownout and shut down. To avoid this issue, set the ENABLE_DCDC bit and set the LINREG_OFFSET fields to 0b2 in anticipation of VDD5V becoming present. The ENABLE_DCDC bit will cause the DC-DC converter to remain on even after 5V is connected and, thus, guarantee a stable supply voltage until the system is configured for removal of 5V. The LINREG_OFFSET fields = 0b2 cause the linear regulators to regulate to a lower target voltage than the switching converter and prevent unwanted interaction between the two power supplies. After the system is configured for removal of 5V, ENABLE_DCDC can be set low and ENABLE_ILIMIT set low in register HW_POWER_5VCTRL to allow the linear regulators to supply the system power, if desired.

11.3.2.2 5-V Power to Battery Power

Configuring the system for a 5-V-to-battery power handoff requires setup code to monitor the battery voltage as well as detect the removal of 5V.

Monitoring the battery voltage is performed by LRADC. Typically, this involves programming the LRADC registers to periodically monitor the battery voltage as described in LRADC Overview. The measured battery voltage should be written into the HW_POWER_BATTMONITOR register field BATT_VAL using the AUTOMATIC field in the HW_LRADC_CONVERSION register. Also, configuring battery brownout should be performed so that the system behaves as desired when 5V is no longer present and the battery is low.

The recommended method to detect removal of 5V requires setting VBUSVALID_5VDETECT and programming the detection threshold VBUSVALID_TRSH to 0x1 in HW_POWER_5VCTRL. Next, in order to minimize linear regulator and DC-DC converter interaction, it is necessary to set the LINREG_OFFSET = 0b2 in the

HW_POWER_VDDIOCTRL, HW_POWER_VDDACTRL, and HW_POWER_VDDDCTRL registers. Finally, set DCDC_XFER and clear PWDN_5VBRNOUT in the HW_POWER_5VCTRL register. This sequence is important because it is safe to disable the powerdown-on-unplug functionality of the device only after the system is completely ready for a transition to battery power.

11.3.2.3 5-V Power and Battery Power

Battery charge can also be enabled to provide additional power efficiency when connected to 5V. If the DC-DC switching converter is also enabled, the buck switching converters will efficiently convert the battery voltage to the desired VDDA, VDDD, and VDDIO voltages (instead of using the less efficient internal linear regulators).

11.3.3 Power-Up Sequence

The DC-DC converter controls the power-up and reset of the i.MX28. The power-up sequence begins when the battery is connected to the BATT pin of the device (or a 5V source is connected to the VDD5V pin). Either the BATT pin or VDD5V provides power to the DC-DC startup circuitry, the crystal oscillator, and the real-time clock. This means that the crystal oscillator can be running, if desired, whenever a battery is connected to BATT pin. This feature allows the real-time clock to operate when the chip is in the off state. The crystal oscillator/RTC is the only power drain on the battery in this state and consumes only a very small amount of power. During this time, the VDDIO, VDDD and VDDA supplies are held at ground. This is the off state that continues until the system power up begins.

Power-up can be started with one of several events:

- PSWITCH pin >= minimum MID level PSWITCH for 100 ms
- VDD5V power pin > 4.25 V for 100 ms
- Real-time clock alarm wakeup

When a power-up event has occurred, if VDD5V is valid, then the on-chip linear regulators charge the VDDD, VDDA and VDDIO rails to their default voltages. If VDD5V is not valid, then the DC-DC supplies the VDDD, VDDA, and VDDIO rails. When the voltage rails have reached their target values, the digital logic reset is deasserted and the CPU begins executing code. If the power supplies do not reach the target values by the time PSWITCH is deasserted or 5V is removed, the system returns to the off state.

The power-up time is dependent on the VDDD/VDDA/VDDIO load and battery or VDD5V voltage, but should be less than 100 ms. The VDDD//VDDAVDDIO load should be minimal during power up to ensure proper startup of the DC-DC converter.

There is an integrated 5-KΩ resistor that can be switched in between the VDDXTAL pin and PSWITCH. If enabled using HW_RTC_PERSISTENT0_AUTO_RESTART, then the device immediately begins the power-up sequence after power-down.

11.3.4 Power-Down Sequence

Power-down is also controlled by the DC-DC converters. When the DC-DC converter detects a power-down event, it returns the device to the off state described above. The power-down sequence is started when one of these events occur:

- HW_POWER_RESET_PWD bit set while the register is unlocked.
- Error condition occurs, as described in Brownout/Error Detection.
- The watchdog timer expires while enabled.
- Fast falling edge (<10 ns) on PSWITCH pin.

The HW_POWER_RESET_PWD_OFF bit disables all the power-down paths except for the watchdog timer when it is set.

The lower 16 bits of the HW_POWER_RESET register can only be written, if the value 0x3E77 is placed in the unlock field.

An external capacitor on the PSWITCH can be used to prevent an unwanted power down due to falling edges. This can also be disabled in HW_POWER_RESET_PWD_OFF register.

11.3.4.1 Powered-Down State

While the chip is powered down, the VDDIO, VDDD and VDDA rail are pulled down to ground. The VDDIO rail is shorted to ground. The crystal oscillator and the RTC can continue to operate by drawing power from the BATT pin. See RTC Overview for more information about operating a crystal and the RTC in the powered-down state.

11.3.5 Reset Sequence

Figure 11-3 shows a flowchart for the power-up, power-down and reset sequences. A reset event can be triggered by unlocking the HW_POWER_RESET register and setting the HW_CLKCTRL_RESET_DIG bit. This reset affects the digital logic only, although the digital logic also includes most of the registers that control the analog portions of the chip. The persistent bits within the RTC block and the power module control bits are not reset using this method. To reset the analog as well as the digital logic, set the HW_CLKCTRL_RESET_CHIP bit. The DC-DC converter and/or linear regulators continue to maintain the power supply rails during the reset.

11.4 PSWITCH Pin Functions

The PSWITCH pin has several functions whose operation is determined by the i.MX28-based product's hardware and software design.

11.4.1 Power On

When the PSWITCH pin voltage is higher than approximately the minimum MID level (as specified in Table 11-1) for >100 ms, the DC-DC converter begins its startup routine. This is the primary method of starting the system through PSWITCH. All products based on the i.MX28 must have a mechanism of bringing PSWITCH high to power up through an always-present supply (for example, battery or VDDXTAL).

Table 11-1. PSWITCH Input Characteristics

Parameter	HW_PWR_STS_PSWITCH	Min	Max	Units
PSWITCH LOW LEVEL [1]	0x00	0.00	0.30	V
PSWITCH MID LEVEL & STARTUP [12]	0x01	0.65	1.50	V
PSWITCH HIGH LEVEL [13]	0x11	(1.1* VDDX-TAL)+ 0.58	2.45	V

1. Consult the reference schematics for recommended PSWITCH button circuitry.
2. A MID LEVEL PSWITCH state can be generated by connecting the VDDXTAL output of the SOC to PSWITCH through a switch.
3. PSWITCH acts like a high impedance input (>300 kΩ) when the voltage applied to it is less than 1.5 V. However, above 1.5 V it becomes lower impedance. To simplify design, it is recommended that a 10 kΩ resistor to VDDIO be applied to PSWITCH to set the HIGH LEVEL state (the PSWITCH input can tolerate voltages greater than 2.45 V as long as there is a 10 kΩ resistor in series to limit the current).

11.4.2 Power Down

If the PSWITCH pin voltage has a falling edge faster than 15 ns, then this sends a power-down request to the DC-DC converter. The fast-falling-edge power-down may be blocked by the HW_POWER_RESET_PWD_OFF function. The fast-falling edge can also be prevented by placing a RC filter on the PSWITCH pin. Most i.MX28-based systems do not use the PSWITCH fast-falling-edge power-down and include the RC filter to prevent it from occurring accidentally.

11.4.3 Software Functions/Recovery Mode

When the PSWITCH pin voltage is pulled up to the minimum MID level (as specified in Table 11-1) the lower HW_POWER_STS_PSWITCH bit is set. Software can poll this bit and perform a function as desired by the product designer. Example functions include a play/pause/power-down button, delay for startup, and so on.

When the PSWITCH pin is connected to VDDIO through a current limiting resistor, the upper HW_POWER_STS_PSWITCH bit is also set. If this bit is set for more than five seconds during ROM boot, the system executes the Freescale USB Firmware Recovery function, as described in Boot Modes. If the product designer does not wish to use Freescale USB Firmware Recovery, the product can be designed to not assert a voltage higher than the maximum MID level (as specified in Table 11-1) on the PSWITCH pin.

Refer to the Freescale i.MX28 reference schematics for example configurations of the PSWITCH pin.

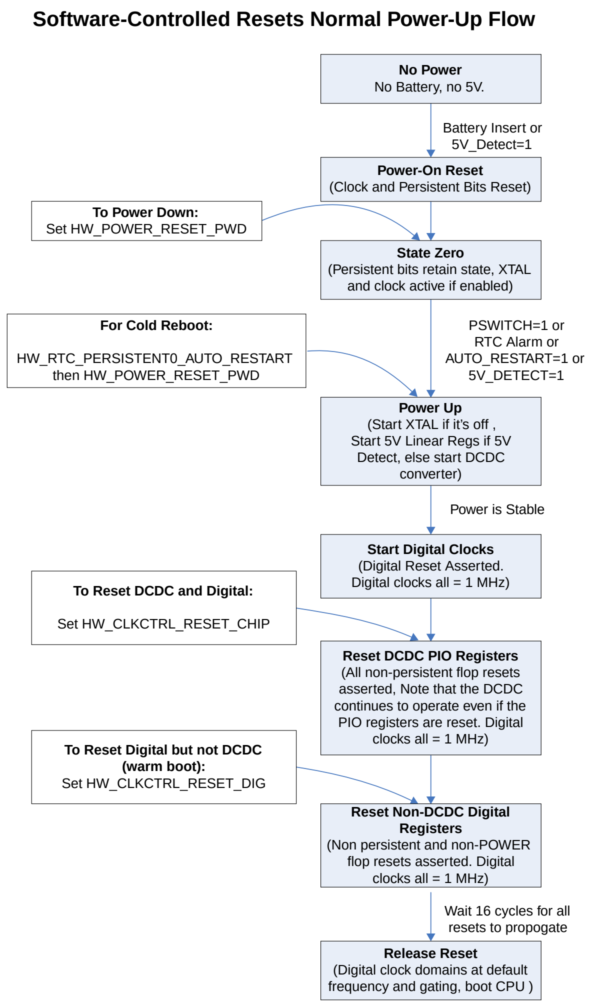

Software-Controlled Resets Normal Power-Up Flow
No Power
No Battery, no 5V.
Battery Insert or 5V_Detect=1
Power-On Reset
(Clock and Persistent Bits Reset)
To Power Down:
Set HW_POWER_RESET_PWD
State Zero
(Persistent bits retain state, XTAL and clock active if enabled)
For Cold Reboot:
HW_RTC_PERSISTENT0_AUTO_RESTART then HW_POWER_RESET_PWD
PSWITCH=1 or RTC Alarm or AUTO_RESTART=1 or 5V_DETECT=1
Power Up
(Start XTAL if it's off , Start 5V Linear Regs if 5V Detect, else start DCDC converter)
Power is Stable
Start Digital Clocks
(Digital Reset Asserted. Digital clocks all = 1 MHz)
To Reset DCDC and Digital:
Set HW_CLKCTRL_RESET_CHIP
Reset DCDC PIO Registers
(All non-persistent flop resets asserted, Note that the DCDC continues to operate even if the PIO registers are reset. Digital clocks all = 1 MHz)
To Reset Digital but not DCDC (warm boot):
Set HW_CLKCTRL_RESET_DIG
Reset Non-DCDC Digital Registers
(Non persistent and non-POWER flop resets asserted. Digital clocks all = 1 MHz)
Wait 16 cycles for all resets to propogate
Release Reset
(Digital clock domains at default frequency and gating, boot CPU)

Figure 11-3. Power-Up, Power-Down, and Reset Flow Chart

11.5 Battery Monitor

The power control system includes a battery monitor. The battery monitor has two functions: battery brownout detection and battery voltage feedback to the DC-DC converter.

If the battery voltage drops below the programmable brownout, then a fast interrupt (FIQ) can be generated for the CPU. Software typically uses the LRADC to monitor the battery voltage and shut down elegantly while there is a minimal operating margin. But, if an unexpected event (such as a battery removal) occurs, then the system needs to be placed immediately in the off state to ensure that it can restart properly. The brownout is controlled in the HW_POWER_BATTMONITOR register. The IRQ must also be enabled in the interrupt collector.

To enable optimum performance over the battery range, the DC-DC converter needs to be provided with the battery voltage, which is measured by the battery pin LRADC. Normally, LRADC channel 7 is dedicated to periodically measuring the battery voltage with a period in the millisecond range for most applications. The voltage is automatically placed into the BATT_VAL field of the HW_POWER_BATTMONITOR register through the HW_LRADC_CONVERSION register. If necessary, software can turn off the automatic battery voltage update and set the BATT_VAL field manually.

11.6 Battery Charger

The battery charger is essentially a linear regulator that has current and voltage limits.

Charge current is software-programmable within the HW_POWER_CHARGE register. Li-Ion batteries can be charged at the lower of 1C, 785 mA, or the VDD5V current limit. USB charging is typically limited to 500 mA or less to meet compliance requirements. Also, battery charge current will be automatically reduced if the current demands from VDD4P2 and the battery charger exceed the CHARGE_4P2_ILIMIT. Full battery charge current will be restored once the VDD4P2 + battery charge current falls below the CHARGE_4P2_ILIMIT value.

Typical charge times for a Li-Ion battery are 1.5 to 3 hours with >70% of the charge delivered in the first hour.

The battery charge voltage limit is 4.2 V for the Li-Ion batteries.

The Li-Ion charge is typically stopped after a certain time limit OR when the charging current drops below 10% of the charge current setting. The HW_POWER_CHARGE register includes controls for the maximum charge current and for the stop charge current. While the charger is delivering current greater than the stop charge limit, the HW_POWER_STS_CHRGSTS bit will be high. This bit should be polled (a low rate of 1 second or greater is fine) during charge. When the bit goes low, the charging is complete. It would be good practice to check that this bit is low for two consecutive checks, as the DC-DC switching might cause a spurious low result. Once this bit goes low, the charger can either be stopped immediately or stopped after a top-off time limit. Although the charger will avoid exceeding the charge voltage limit on the battery, it is NOT recommended to leave the charger active indefinitely. It should be turned off when the charge is complete.

One can programatically monitor the battery voltage using the LRADC. The charger has its own (very robust) voltage limiting that operates independently of the LRADC. But monitoring the battery voltage during the charge might be helpful for reporting the charge progress.

The battery charger is capable of generating a large amount of heat within the i.MX28, especially at currents above 400 mA. The dissipated power can be estimated as: (5V – battery_volt) * current. At max current (785 mA) and a 3-V battery, the charger can dissipate 1.57 W, raising the die temp as much as 80 C°. To ensure that the system operates correctly, the die temperature sensor should be monitored every 100 ms. If the die temperature exceeds 115 C° (the max value for the chip temp sensor), then the battery charge current must be reduced. The LRADC can also be used to monitor the battery temperature or chip temperature. There is an integrated current source for the external temperature sensor that can be configured and enabled through HW_LRADC_CTRL2 register.

11.7 11.5.1 Battery Detection

The battery is detected by checking the battery brownout status or measuring the battery pin voltage. If either indicates the battery voltage is below an application-defined low-voltage threshold, the battery is not attached. When the battery voltage is measured in the expected operational range, the FASTSETTLING bitfield in the HW_POWER_REFCTRL register should be set to enable an internal load on the battery. If no battery is connected, the internal load will discharge the capacitor on the battery pin to bring the voltage below the application-defined low-voltage threshold. If a battery is connected, the voltage should remain in the expected operational range. After battery detection is complete, the FASTSETTLING bitfield should be cleared to disable the internal load.

The internal load is automatically disabled when a battery brownout is detected. In this case, the FASTSETTLING bitfield must be cycled to re-enabled the load. This is important for situations where the load is enabled at a battery voltage just above the battery brownout threshold and then the battery voltage drops below the threshold causing the brownout status to change. Software must account for this situation by re-enabling the load when the battery voltage is below the brownout threshold. The length of time for the battery capacitor to discharge is variable between designs and is dependent on the amount of capacitance on the board.

11.8 Silicon Speed Sensor

The i.MX28 integrates three silicon speed sensors to measure the performance characteristics of an individual die at its ambient temperature and process parametrics. One is inside ARM9 to measure ARM9 performance, one is in core logic which including system bus and peripherals to measure core logic performance and another one is inside DC-DC block. A silicon speed sensor consists of a ring oscillator and a frequency counter. The ring oscillator runs on the VDDD power rail. Therefore, its frequency tracks the silicon performance as it changes in response to changes in operating voltage and temperature. The crystal oscillator is directly used as the precision time base for measuring the frequency of a ring oscillator. The ring oscillator is normally disabled. There is a 8-bit/16-bit counter connected to the ring oscillator that performs the frequency measurement. See HW_POWER_SPEED register.

Thus, the counter holds the number of cycles the ring oscillator was able to generate during one crystal clock period. The natural frequency of the ring oscillator strongly tracks the silicon process parametrics, that is, faster silicon processes yield ring oscillators that run faster and thereby yield larger count values. The natural frequency tracks junction temperature effects on silicon speed as well.

The information given by the speed sensors can be used with the silicon temperature and process parameters, which can also be monitored by system software. Freescale can provide a power management application note and firmware that takes full advantage of the on-chip monitoring functions to enable minimum-voltage operation.

11.9 Thermal-Overload Protection Circuit

The i.MX28 integrates a thermal-overload protection circuit to shut down the device when the on-die temperature exceeds a programmed target value. It consists of a bipolar based voltage generator whose characteristics track the on-die temperature. This temperature-dependent voltage is compared to a fixed voltage level that does not vary with the temperature to trigger a shut down of the device. The thermal-overload protection circuit is disabled by default, but can be enabled after the system is up. Its default thermal trip-point is set to 115degC and can be programmed up to 150degC in 5degC steps. See HW_POWER_THERMAL register.

11.10 Bandgap Reference Generator

The i.MX28 integrates a bandgap generator which creates supply and temperature stable voltage and current references for the on-chip blocks. Several bits are provided to adjust the reference voltages and currents. See HW_POWER_REFCTRL register.

11.11 Interrupts

The power system supports nine CPU interrupt events that are programmable within the HW_POWER_CTRL register. The interrupts are listed in Table 11-2.

Table 11-2. Power System Interrupts

HW_POWER_CTRL INTERRUPT BIT	Description
VDDA_BO_IRQ	VDDA Brownout

HW_POWER_CTRL INTERRUPT BIT	Description
VDDD_BO_IRQ	VDDD Brownout
VDDIO_BO_IRQ	VDDIO Brownout
BATT_BO_IRQ	Battery Brownout
VDD5V_GT_VDDIO_IRQ	VDD5V > (VDDIO + 0.6)
DC_OK_IRQ	Voltage Supplies Ok after target voltage change
VBUSVALID_IRQ	VDD5V > Vbusvalid Threshold
LINREG_OK_IRQ	Debug use only
PSWITCH_IRQ	PSWITCH Status

The VDDA_BO_IRQ, VDDD_BO_IRQ, VDDIO_BO_IRQ, and BATT_BO_IRQ each have their own interrupt line back to the interrupt collector. However, the remaining five interrupts—VDD5V_GT_VDDIO_IRQ, DC_OK_IRQ, VBUSVALID_IRQ, LINREG_OK_IRQ and PSWITCH_IRQ—all share a single interrupt line. In this case, software must read the interrupt status bits to discover which event caused the interrupt.

11.12 Programmable Registers

iMX28 POWER Hardware Register Format Summary

HW_POWER memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8004_4000	Power Control Register (HW_POWER_CTRL)	32	R/W	0001_0024h	11.12.1/912
8004_4010	DC-DC 5V Control Register (HW_POWER_5VCTRL)	32	R/W	2010_0490h	11.12.2/915
8004_4020	DC-DC Minimum Power and Miscellaneous Control Register (HW_POWER_MINPWR)	32	R/W	0000_0000h	11.12.3/917
8004_4030	Battery Charge Control Register (HW_POWER_CHARGE)	32	R/W	0001_0000h	11.12.4/919
8004_4040	VDDD Supply Targets and Brownouts Control Register (HW_POWER_VDDDCTRL)	32	R/W	0032_0710h	11.12.5/921
8004_4050	VDDA Supply Targets and Brownouts Control Register (HW_POWER_VDDACTRL)	32	R/W	0000_270Ch	11.12.6/923
8004_4060	VDDIO Supply Targets and Brownouts Control Register (HW_POWER_VDDIOCTRL)	32	R/W	0000_2406h	11.12.7/924
8004_4070	VDDMEM Supply Targets Control Register (HW_POWER_VDDMEMCTRL)	32	R/W	0000_0230h	11.12.8/926

HW_POWER memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8004_4080	DC-DC Converter 4.2V Control Register (HW_POWER_DCDC4P2)	32	R/W	0000_0018h	11.12.9/927
8004_4090	DC-DC Miscellaneous Register (HW_POWER_MISC)	32	R/W	0000_0000h	11.12.10/929
8004_40A0	DC-DC Duty Cycle Limits Control Register (HW_POWER_DCLIMITS)	32	R/W	0000_0C5Fh	11.12.11/930
8004_40B0	Converter Loop Behavior Control Register (HW_POWER_LOOPCTRL)	32	R/W	0000_0021h	11.12.12/931
8004_40C0	Power Subsystem Status Register (HW_POWER_STS)	32	R	0000_0200h	11.12.13/933
8004_40D0	Transistor Speed Control and Status Register (HW_POWER_SPEED)	32	R/W	0000_0000h	11.12.14/936
8004_40E0	Battery Level Monitor Register (HW_POWER_BATTMONITOR)	32	R/W	0000_0A00h	11.12.15/937
8004_4100	Power Module Reset Register (HW_POWER_RESET)	32	R/W	0000_0000h	11.12.16/938
8004_4110	Power Module Debug Register (HW_POWER_DEBUG)	32	R/W	0000_0000h	11.12.17/939
8004_4120	Power Module Thermal Reset Register (HW_POWER_THERMAL)	32	R/W	0000_0080h	11.12.18/940
8004_4130	Power Module USB1 Manual Controls Register (HW_POWER_USB1CTRL)	32	R/W	0000_000Eh	11.12.19/941
8004_4140	Power Module Special Register (HW_POWER_SPECIAL)	32	R/W	0000_0000h	11.12.20/942
8004_4150	Power Module Version Register (HW_POWER_VERSION)	32	R	0400_0000h	11.12.21/943
8004_4160	Analog Clock Control Register (HW_POWER_ANACLKCTRL)	32	R/W	8400_0400h	11.12.22/943
8004_4170	POWER Reference Control Register (HW_POWER_REFCTRL)	32	R/W	0000_0000h	11.12.23/945

11.12.1 Power Control Register (HW_POWER_CTRL)

The Power Control Register contains control bits specific to the digital section.

HW_POWER_CTRL: 0x000

HW_POWER_CTRL_SET: 0x004

HW_POWER_CTRL_CLR: 0x008

HW_POWER_CTRL_TOG: 0x00C

Address: HW_POWER_CTRL – 8004_4000h base + 0h offset = 8004_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2				PSWITCH_MID_TRAN	RSRVD1		DCDC4P2_BO_IRQ	ENIRQ_DCDC4P2_BO	VDD5V_DROOP_IRQ	ENIRQ_VDD5V_DROOP	PSWITCH_IRQ	PSWITCH_IRQ_SRC	POLARITY_PSWITCH	ENIRQ_PSWITCH	POLARITY_DC_OK
W	▨	▨	▨	▨		▨	▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	DC_OK_IRQ	ENIRQ_DC_OK	BATT_BO_IRQ	ENIRQBATT_BO	VDDIO_BO_IRQ	ENIRQ_VDDIO_BO	VDDA_BO_IRQ	ENIRQ_VDDA_BO	VDDD_BO_IRQ	ENIRQ_VDDD_BO	POLARITY_VBUSVALID	VBUSVALID_IRQ	ENIRQ_VBUS_VALID	POLARITY_VDD5V_GT_VDDIO	VDD5V_GT_VDDIO_IRQ	ENIRQ_VDD5V_GT_VDDIO
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0

HW_POWER_CTRL field descriptions

Field	Description
31–28 RSRVD2	Reserved.
27 PSWITCH_MID_TRAN	When this bit is set, it selects the from-mid-transition interrupt functionality for PSWITCH. When set, in conjuction with ENIRQ_PSWITCH, it will set PSWITCH_IRQ when either a 01 to 11 or 01 to 00 transition is detected on the PSWITCH comparators. When this bit is set, PSWITCH_IRQ_SRC and POLARITY_PSWITCH have no effect.
26–25 RSRVD1	Reserved.
24 DCDC4P2_BO_IRQ	Interrupt Status for 4P2_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
23 ENIRQ_DCDC4P2_BO	Enable interrupt for 4P2_BO.
22 VDD5V_DROOP_IRQ	Interrupt Status for VDD5V_DROOP. Reset this bit by writing a one to the SCT clear address space and not by a general write.
21 ENIRQ_VDD5V_DROOP	Enable interrupt for VDD5V_DROOP.
20 PSWITCH_IRQ	Interrupt status for PSWITCH signals. Interrupt polarity is set using POLARITY_PSWITCH. Comparator bit is selected through PSWITCH_IRQ_SRC. Reset this bit by writing a one to the SCT clear address space and not by a general write.
19 PSWITCH_IRQ_SRC	Set to 1 to use HW_POWER_STS_PSWITCH bit 1 as source, 0 for HW_POWER_STS_PSWITCH bit 0

HW_POWER_CTRL field descriptions (continued)

Field	Description
18 POLARITY_ PSWITCH	Set to 1 to interupt when the interrupt source is high, 0 for low
17 ENIRQ_ PSWITCH	Interrupt status for PSWITCH signal. Interrupt polarity is set using POLARITY_PWITCH and source selected by PSWITCH_SRC.
16 POLARITY_DC_ OK	Debug use only. Set to 1 to check for linear regulators ok. Set to 0 to check for 5V disconnected.
15 DC_OK_IRQ	Interrupt Status for DC_OK. Reset this bit by writing a one to the SCT clear address space and not by a general write.
14 ENIRQ_DC_OK	Enable interrupt for DC_OK.
13 BATT_BO_IRQ	Interrupt Status for BATT_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
12 ENIRQBATT_BO	Enable interrupt for battery brownout.
11 VDDIO_BO_IRQ	Interrupt Status for VDDIO_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
10 ENIRQ_VDDIO_ BO	Enable interrupt for VDDIO brownout.
9 VDDA_BO_IRQ	Interrupt Status for VDDA_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 ENIRQ_VDDA_ BO	Enable interrupt for VDDIO brownout.
7 VDDD_BO_IRQ	Interrupt Status for VDDD_BO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
6 ENIRQ_VDDD_ BO	Enable interrupt for VDDD brownout.
5 POLARITY_ VBUSVALID	Set to 1 to check for 5V connected using VBUSVALID status bit. Set to 0 to check for 5V disconnected.
4 VBUSVALID_IRQ	Interrupt status for VBUSVALID signal. Interrupt polarity is set using POLARITY_VBUSVALID. Reset this bit by writing a one to the SCT clear address space and not by a general write.
3 ENIRQ_VBUS_ VALID	Enable interrupt for 5V detect using VBUSVALID.

HW_POWER_CTRL field descriptions (continued)

Field	Description
2 POLARITY_ VDD5V_GT_ VDDIO	Set to 1 to check for 5V connected. Set to 0 to check for 5V disconnected.
1 VDD5V_GT_ VDDIO_IRQ	Interrupt status for VDD5V_GT_VDDIO signal. Interrupt polarity is set using POLARITY_VDD5V_GT_VDDIO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
0 ENIRQ_VDD5V_ GT_VDDIO	Enable interrupt for 5V detect.

11.12.2 DC-DC 5V Control Register (HW_POWER_5VCTRL)

This register contains the configuration options of the power management subsystem that are available when external 5V is applied.

HW_POWER_5VCTRL: 0x010

HW_POWER_5VCTRL_SET: 0x014

HW_POWER_5VCTRL_CLR: 0x018

HW_POWER_5VCTRL_TOG: 0x01C

Address: HW_POWER_5VCTRL – 8004_4000h base + 10h offset = 8004_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD6		VBUSDROOP_TRSH		RSRVD5	HEADROOM_ADJ			RSRVD4		PWD_CHARGE_4P2		RSRVD3		CHARGE_4P2_ILIMIT[17:16]	
W	▨	▨			▨				▨	▨			▨	▨		
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CHARGE_4P2_ILIMIT[15:12]				RSRVD2	VBUSVALID_TRSH			PWDN_5VBRNOUT	ENABLE_LINREG_ILIMIT	DCDC_XFER	VBUSVALID_5VDETECT	VBUSVALID_TO_B	ILIMIT_EQ_ZERO	PWRUP_VBUS_CMPS	ENABLE_DCDC
W					▨											
Reset	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0

HW_POWER_5VCTRL field descriptions

Field	Description
31–30 RSRVD6	Reserved.
29–28 VBUSDROOP_ TRSH	Set the threshold for the VBUSDROOP comparator. This comparator should typically be programmed at least 200mV above VBUSVALID, and can be used to terminate battery charge when the voltage on VDD5V falls below the programmed threshold. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 0b00: 4.3V 0b01: 4.4V 0b10: 4.5V(default) 0b11: 4.7V
27 RSRVD5	Reserved.
26–24 HEADROOM_ ADJ	Adjustment to optimize the performance of the battery charge and 4.2V regulation circuit at low 5v voltages.
23–22 RSRVD4	Reserved.
21–20 PWD_CHARGE_ 4P2	Controls the power down of both the battery charger and 4.2V regulation circuit. Default is powered down.
19–18 RSRVD3	Reserved.
17–12 CHARGE_4P2_ ILIMIT	Limits the combined current from 5V that the battery charger and DCDC_4P2 circuit consume. Current represented by each bits is as follows: (400 mA, 200 mA, 100 mA, 50 mA, 20 mA, 10 mA) = (bit 5,bit 4, bit 3, bit 2, bit 1, bit 0). The DCDC_4P2 circuit is given priority as the sum of the currents exceeds the limit.
11 RSRVD2	Reserved.
10–8 VBUSVALID_ TRSH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 0b000: 2.9V 0b001: 4.0V 0b010: 4.1V 0b011: 4.2V 0b100: 4.3V(default) 0b101: 4.4V 0b110: 4.5V 0b111: 4.6V
7 PWDN_ 5VBRNOUT	The purpose of this bit is to power down the device if 5V is removed before the system is completely initialized. Clear this bit to disable automatic hardware powerdown AFTER the system is configured for 5v removal. This bit should not be set if DCDC_XFER is set.

HW_POWER_5VCTRL field descriptions (continued)

Field	Description
6 ENABLE_ LINREG_ILIMIT	Enable the current limit in the linear regulators. The current limit is active during powerup from 5v and automatically disables before the ROM executes. This limit is needed to meet the USB in rush current specification of 100mA + 50uC. Note this linear regulator current limit is not related to the CHARGE_4P2_ILIMIT.
5 DCDC_XFER	Enable automatic transition to switching DC-DC converter when VDD5V is removed. The LRADC must be operational and the BATT_VAL field must be written with the battery voltage using 8-mV step-size. It is also important to set the EN_BATADJ field.
4 VBUSVALID_ 5VDETECT	Power up and use VBUSVALID comparator as detection circuit for 5V in the switching converter. The VBUSVALID comparator provides a more accurate and adjustable threshold to determine the presence of 5V in the system, and is the recommended method of detecting 5V.
3 VBUSVALID_ TO_B	This bit muxes the Bvalid comparator to the VBUSVALID comparator and is used for test purposes only.
2 ILIMIT_EQ_ ZERO	The amount of current the device will consume from the 5V rail is minimized. The VDDIO linear regulator current limit is set to zero mA. Also, the source of current for the crystal oscillator and RTC is switched to the battery. Note that this functionality does not affect battery charge.
1 PWRUP_VBUS_ CMPS	Powers up comparators for 5v
0 ENABLE_DCDC	Enables the switching DC-DC converter when 5V is present. It is recommended to set ILIMIT_EQ_ZERO, ENABLE_ILIMIT, and all LINREG_OFFSET bits when enabling this functionality.

11.12.3 DC-DC Minimum Power and Miscellaneous Control Register (HW_POWER_MINPWR)

This register controls options to drop the power used by the switching DC-DC converter. These bits should only be modified with guidance from Freescale.

HW_POWER_MINPWR: 0x020

HW_POWER_MINPWR_SET: 0x024

HW_POWER_MINPWR_CLR: 0x028

HW_POWER_MINPWR_TOG: 0x02C

Address: HW_POWER_MINPWR – 8004_4000h base + 20h offset = 8004_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1 [15:15]	LOWPWR_4P2	RSVD1	PWD_BO	USE_VDDXTAL_ VBG	PWD_ANA_ CMPS	ENABLE_ OSC	SELECT_ OSC	VBG_ OFF	DOUBLE_ FETS	HALF_FETS	LESSANA_I	PWD_ XTAL24	DC_ STOPCLK	EN_ DC_ PFM	DC_ HALFCLK
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_MINPWR field descriptions

Field	Description
31–15 RSRVD1	Reserved.
14 LOWPWR_4P2	Enable low power regulation of DCDC_4P2 limited to 2.5mA.
13 RSVD1	Program this field to 0x0.
12 PWD_BO	Powers down supply brownout comparators. This should only be used when monitoring supply brownouts is not needed.
11 USE_VDDXTAL_ VBG	Change the reference used in the dcdc converter to a low power, less accurate reference. This should only be used when achieving minimum system power is more important than supply voltage accuracy.
10 PWD_ANA_ CMPS	Powers down analog comparators used in the power module, including the VDD and VDDA brownout comparators, low power reference. This bit should only be set to reach absolute minimum system power when running from the linear regulators and supply accuracy and VDDD/VDDA brownout detection is not important.
9 ENABLE_OSC	Enables the internal oscillator. This oscillator is less accurate , but lower power than the 24 MHz xtal.
8 SELECT_OSC	Switch internal 24 MHz clock reference to the less accurate internal oscillator. This bit should be set only when accuracy of the 24 MHz clock is not important. The value of this bit should only be changed when ENABLE_OSC=1 and PWD_XTAL24=0.
7 VBG_OFF	Powers down the bandgap reference. This should only be used in 5v-powered applications when absolute minimum power is more important than supply voltage accuracy. USB_I_SUSPEND must be set before this bit is set.
6 DOUBLE_FETS	Approximately doubles the size of power transistors in DC-DC converter. This maybe be useful in high power conditions.
5 HALF_FETS	Disable half the power transistors in DC-DC converter. This maybe be useful in low-power conditions when the increased resistance of the power FETs is acceptable.

HW_POWER_MINPWR field descriptions (continued)

Field	Description
4 LESSANA_I	Reduce DC-DC analog bias current 20%. This bit is intended to reduce power in low-performance operating modes, such as USB suspend.
3 PWD_XTAL24	Powers down the 24 MHz oscillator, even when the system is still operating. This can be used with ENABLE_OSC and SELECT_OSC to reduce current in usb suspend and other low power modes where the accuracy of the clock is not important.
2 DC_STOPCLK	Stop the clock to internal logic of switching DC-DC converter. This bit will take effect only after the switching FETs are off, due to battery configuration, PFM mode, or internal linear regulator operation.
1 EN_DC_PFM	Forces DC-DC to operate in a Pulse Frequency Modulation mode. Intended to allow minimum system power in very low-power configurations when increased ripple on the supplies is acceptable. Also, HYST_SIGN in HW_POWER_LOOPCTRL should be set high when using PFM mode.
0 DC_HALFCLK	Slow down DC-DC clock from 1.5 MHz to 750 kHz. This maybe be useful to improve efficiency at light loads or improve EMI radiation, although peak-to-peak voltage on the supplies will increase.

11.12.4 Battery Charge Control Register (HW_POWER_CHARGE)

This register cotrols the battery charge features for both NiMH slow charge and Li-Ion charge.

HW_POWER_CHARGE: 0x030

HW_POWER_CHARGE_SET: 0x034

HW_POWER_CHARGE_CLR: 0x038

HW_POWER_CHARGE_TOG: 0x03C

Address: HW_POWER_CHARGE – 8004_4000h base + 30h offset = 8004_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD6					ADJ_VOLT			RSRVD5	ENABLE_LOAD	RSRVD4	ENABLE_FAULT_DETECT	CHRG_STS_OFF	LIION_4P1	RSRVD3	PWD_BATTCHRG
W	▨	▨	▨	▨	▨				▨		▨				▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2		ENABLE_CHARGER_USB1	ENABLE_CHARGER_USB0	STOP_ILIMIT				RSRVD1		BATTCHRG_I					
W	▨	▨							▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_CHARGE field descriptions

Field	Description
31–27 RSRVD6	Reserved.
26–24 ADJ_VOLT	Adjustments to the final LiIon final voltage. These bits should not be set unless recommended by Freescale. 0b000: no change 0b001: -0.25% 0b010: +0.50% 0b011: -0.75% 0b100: +0.25% 0b101: -0.50% 0b110: +0.75% 0b111: -1.00%
23 RSRVD5	Reserved.
22 ENABLE_LOAD	Enable 100ohm load on the regulated 4.2V output. This bit should not be set unless recommended by Freescale.
21 RSRVD4	Reserved.
20 ENABLE_ FAULT_DETECT	Enable fault detection in the battery charger. When enabled, this bit will power down the battery charger when the VDD5V voltage falls below the battery voltage. The fault can be cleared by cycling PWD_CHARGE_4P2. The fault detection is visible through HW_POWER_STS_VDD5V_FAULT.
19 CHRG_STS_OFF	Setting this bit disables the CHRGSTS status bit. Disabling CHRGSTS should only be done when the switching converter is enabled during battery charge if noise from the switching converter causes CHRGSTS to toggle excessively.
18 LIION_4P1	This bit has no effect.
17 RSRVD3	Reserved.
16 PWD_ BATTCHRG	Power-down the battery charge circuitry. This should only be set low when 5V is present. This bit can be set (charger turned off) automatically by the threshold units of the LRADC if programmed to do so.
15–14 RSRVD2	Reserved.

HW_POWER_CHARGE field descriptions (continued)

Field	Description
13 ENABLE_ CHARGER_ USB1	Enable 125k pullup on USB_DP and 375k on USB_DN to provide USB_CHARGER functionality for USB1. This functionality is a new USB spec and should not be enabled unless recommended by Freescale.
12 ENABLE_ CHARGER_ USB0	Enable 125k pullup on USB_DP and 375k on USB_DN to provide USB_CHARGER functionality for USB0. This functionality is a new USB spec and should not be enabled unless recommended by Freescale.
11–8 STOP_ILIMIT	Current threshold at which the Li-Ion battery charger signals to stop charging. The current represented by each bit is as follows: (100 mA, 50 mA, 20 mA, 10 mA) = (bit 3, bit 2, bit 1, bit 0) It is recommended to set this value to 10% of the charge current.
7–6 RSRVD1	Reserved.
5–0 BATTCHRG_I	Magnitude of the battery charge current, the current represented by each bit is as follows: (400 mA, 200 mA, 100 mA, 50 mA, 20 mA, 10 mA) = (bit 5,bit 4, bit 3, bit 2, bit 1, bit 0)

11.12.5 VDDD Supply Targets and Brownouts Control Register (HW_POWER_VDDDCTRL)

This register controls the voltage targets and brownout targets for the VDDD supply generated from the switching DC-DC converter and integrated linear regulators. The brownout comparators default enabled.

Address: HW_POWER_VDDDCTRL – 8004_4000h base + 40h offset = 8004_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ADJTN				RSRVD4				PWDN_ BRNOUT	DISABLE_ STEPPING	ENABLE_ LINREG	DISABLE_FET	RSRVD3		LINREG_ OFFSET	
W					▨	▨	▨	▨					▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2					BO_OFFSET			RSRVD1			TRG				
W	▨	▨	▨	▨	▨				▨	▨	▨					
Reset	0	0	0	0	0	1	1	1	0	0	0	1	0	0	0	0

HW_POWER_VDDDCTRL field descriptions

Field	Description
31–28 ADJTN	Two's complement number that can be used to adjust the duty cycle of VDDD. This is intended for test purposes only

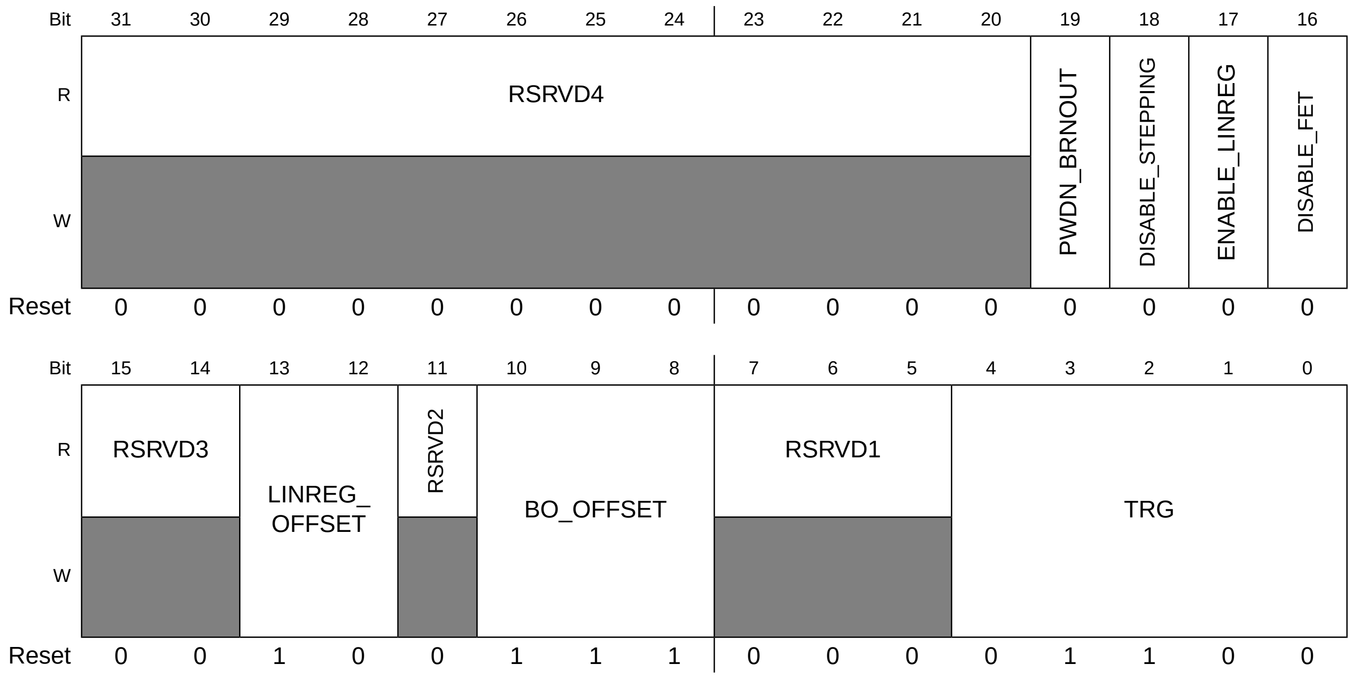
HW_POWER_VDDDCTRL field descriptions (continued)

Field	Description
27–24 RSRVD4	Reserved.
23 PWDN_BRNOUT	Powers down the device after the DC-DC converter completes startup if a VDDD brownout occurs. This function is only active when 5V is not present. Additionally, software should clear this bit and disable this function after the system is configured for a VDDD brownout and the VDDD brownout interrupt is enabled.
22 DISABLE_ STEPPING	Disables the default behavior of the voltage stepping algorithm when the TRG field is updated. By default, the control loop steps sequentially through 25mV steps on both target and brownout voltage to minimize transients during TRG adjustments. When this bit is set, the entire target voltage and brownout adjustment takes place at one time, which will speed transitions, but will result in increased supply transients as well as possible brownouts when the new (TRG - BO_OFFSET) >= old TRG. Thus, it is recommended to change TRG by less than the (BO_OFFSET-2) value when using DISABLE_STEPPING. This bit should be set high when powering VDDD from the integrated linear regulators.
21 ENABLE_ LINREG	Enables the VDDD linear regulator converter when the switching converter is active. By default, the linear regulator is not active when the switching converter is active.
20 DISABLE_FET	Disable the VDDD switching converter output.
19–18 RSRVD3	Reserved.
17–16 LINREG_ OFFSET	Number of 25mV steps between linear regulator output voltage and switching converter target. 00b = 0 steps, recommended when powering VDDD only from linear regulator and ENABLE_DCDC=DCDC_XFER=0. It is also recommended to set DISABLE_STEPPING when powering VDDD from the linear regulators. 01b = 1 step above. 10b = 1 step below (default), important when powering VDDD from DC-DC converter and linear regulator simultaneously. 11b = 2 steps below.
15–11 RSRVD2	Reserved.
10–8 BO_OFFSET	Brownout voltage offset in 25mV steps below the TRG value. Note that the hardware only supports brownout voltages between 0.8V and 1.475V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
7–5 RSRVD1	Reserved.
4–0 TRG	Voltage level of the VDDD supply. The step size of this field is 25 mV. 0x00 = 0.8 V, 0x1F = 1.575 V, and the reset value = 1.2 V. This field should not be set above the VDDD operating maximum voltage as specified in the Datasheet. It is also recommended to set DISABLE_STEPPING when powering VDDD from the integrated linear regulators. Setting DISABLE_STEPPING does set additional restrictions on TRG adjusments.

11.12.6 VDDA Supply Targets and Brownouts Control Register (HW_POWER_VDDACTRL)

This register controls the voltage targets and brownout targets for the VDDA supply generated from the switching DC-DC converter and integrated linear regulators. The brownout comparators default enabled.

Address: HW_POWER_VDDACTRL – 8004_4000h base + 50h offset = 8004_4050h



HW_POWER_VDDACTRL field descriptions

Field	Description
31–20 RSRVD4	Reserved.
19 PWDN_BRNOUT	Powers down the device after the DC-DC converter completes startup if a VDDA brownout occurs. This function is only active when 5V is not present. Additionally, software should clear this bit and disable this function after the system is configured for a VDDA brownout and the VDDA brownout interrupt is enabled.
18 DISABLE_ STEPPING	Disables the default behavior of the voltage stepping algorithm when the TRG field is updated. By default, the control loop steps sequentially through 25mV steps on both target and brownout voltage to minimize transients during TRG adjustments. When this bit is set, the entire target voltage and brownout adjustment takes place at one time, which will speed transitions, but will result in increased supply transients as well as possible brownouts when the new (TRG - BO_OFFSET) >= old TRG. Thus, it is recommended to change TRG by less than the (BO_OFFSET-2) value when using DISABLE_STEPPING. This bit should be set high when powering VDDA from the integrated linear regulators.
17 ENABLE_ LINREG	Enables the VDDA linear regulator converter when the switching converter is active. By default, the linear regulator is not active when the switching converter is active.

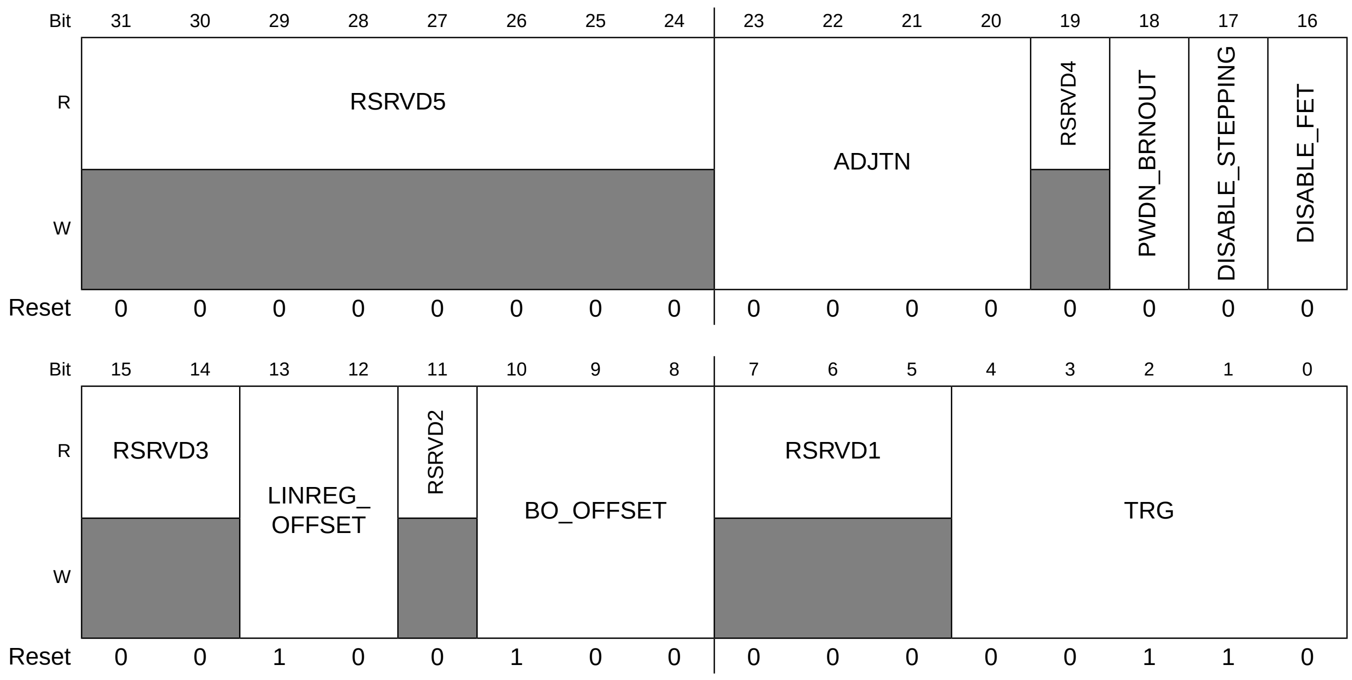
HW_POWER_VDDACTRL field descriptions (continued)

Field	Description
16 DISABLE_FET	Disable the VDDA switching converter output. The switching converter is enabled by default when the battery is present or the ENABLE_DCDC is set.
15–14 RSRVD3	Reserved.
13–12 LINREG_ OFFSET	Number of 25mV steps between linear regulator output voltage and switching converter target. 00b = 0 steps, recommended when powering VDDA from linear regulator and ENABLE_DCDC=DCDC_XFER=0. It is also recommended to set DISABLE_STEPPING when powering VDDA from the linear regulators. 01b = 1 step above. 10b = 1 step below (default), important when powering VDDA from DC-DC converter and linear regulator simultaneously. 11b = 2 steps below.
11 RSRVD2	Reserved.
10–8 BO_OFFSET	Brownout voltage offset in 25mV steps below the TRG value. Note that the hardware only supports brownout voltages between 1.4V and 2.175V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
7–5 RSRVD1	Reserved.
4–0 TRG	Voltage level of the VDDA supply. The step size of this field is 25 mV. 0x00 = 1.5 V, 0x1F = 2.275 V, and the reset value = 1.8 V. This field should not be set above the VDDA operating maximum voltage as specified in the Datasheet. It is also recommended to set DISABLE_STEPPING when powering VDDA from the integrated linear regulators. Setting DISABLE_STEPPING does set additional restrictions on TRG adjusments.

11.12.7 VDDIO Supply Targets and Brownouts Control Register (HW_POWER_VDDIOCTRL)

This register controls the voltage targets and brownout targets for the VDDIO supply generated from the switching DC-DC converter and integrated linear regulators. The brownout comparators default enabled.

Address: HW_POWER_VDDIOCTRL – 8004_4000h base + 60h offset = 8004_4060h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD5
ADJTN
RSRVD4
PWDN_BRNOUT
DISABLE_STEPPING
DISABLE_FET
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RSRVD3
LINREG_ OFFSET
RSRVD2
BO_OFFSET
RSRVD1
TRG
Reset 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0

HW_POWER_VDDIOCTRL field descriptions

Field	Description
31–24 RSRVD5	Reserved.
23–20 ADJTN	Two's complement number that can be used to adjust the duty cycle of VDDIO. This is intended for test purposes only
19 RSRVD4	Reserved.
18 PWDN_BRNOUT	Powers down the device after the DC-DC converter completes startup if a VDDIO brownout occurs. This function is only active when 5V is not present. Additionally, software should clear this bit and disable this function after the system is configured for a VDDIO brownout and the VDDIO brownout interrupt is enabled.
17 DISABLE_ STEPPING	Disables the default behavior of the voltage stepping algorithm when the TRG field is updated. By default, the control loop steps sequentially through 25mV steps on both target and brownout voltage to minimize transients during TRG adjustments. When this bit is set, the entire target voltage and brownout adjustment takes place at one time, which will speed transitions, but will result in increased supply transients as well as possible brownouts when the new (TRG - BO_OFFSET) >= old TRG. Thus, it is recommended to change TRG by less than the (BO_OFFSET-2) value when using DISABLE_STEPPING. This bit should be set high when powering VDDIO from the integrated linear regulators.
16 DISABLE_FET	Disable the VDDIO switching converter output. The switching converter is enabled by default when the battery is present or the ENABLE_DCDC is set.
15–14 RSRVD3	Reserved.
13–12 LINREG_ OFFSET	Number of 25mV steps between linear regulator output voltage and switching converter target.

HW_POWER_VDDIOCTRL field descriptions (continued)

Field	Description
	00b = 0 steps, recommended when powering VDDIO from linear regulator and ENABLE_DCDC=DCDC_XFER=0. It is also recommended to set DISABLE_STEPPING when powering VDDIO from the linear regulators. 01b = 1 step above. 10b = 1 step below (default), important when powering VDDIO from DC-DC converter and linear regulator simultaneously. 11b = 2 steps below.
11 RSRVD2	Reserved.
10–8 BO_OFFSET	Brownout voltage offset in 50 mV steps below the TRG value. Note that the hardware only supports brownout voltages between 2.7V and 3.475V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
7–5 RSRVD1	Reserved.
4–0 TRG	Voltage level of the VDDIO supply. The step size of this field is 50 mV. 0x00 = 2.8 V and the reset value = 3.1 V. Note that the maximum programmable voltage is 3.6V, therefore code values of 0x10 to 0x1F all map to 3.6 V. It is also recommended to set DISABLE_STEPPING when powering VDDIO from the integrated linear regulators. Setting DISABLE_STEPPING does set additional restrictions on TRG adjusments.

11.12.8 VDDMEM Supply Targets Control Register (HW_POWER_VDDMEMCTRL)

This register controls the voltage target for a memory supply generated from VDDA. This supply is intended for use with external memories such as LV-DDR that have unique voltage requirements not compatible with VDDIO, VDDA , or VDDD.

Address: HW_POWER_VDDMEMCTRL – 8004_4000h base + 70h offset = 8004_4070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2[15:11]					PULLDOWN_ ACTIVE	ENABLE_ILIMIT	ENABLE_LINREG	BO_OFFSET			TRG				
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0

HW_POWER_VDDMEMCTRL field descriptions

Field	Description
31–11 RSRVD2	Reserved.
10 PULLDOWN_ ACTIVE	Activates pulldown on external memory supply. This bit should be set before the regulator is enabled to be sure the supply voltage powers up from ground. Default is pulldown inactive.
9 ENABLE_ILIMIT	Controls the inrush limit (~10mA) for the memory supply voltage. Default is active. This should remain active until the supply settles after enabling the linreg. This should be disabled before accessing the memory.
8 ENABLE_ LINREG	Enables the regulator that creates the external memory supply voltage. After enabling the linreg need to wait until the VDDMEM rail is up before disabling the linreg current limit and accessing the memories. 500uS is usually an adequate delay, but it can be longer if VDDMEM cap is >1uF.
7–5 BO_OFFSET	Brownout voltage offset in 25mV steps below the TRG value. Note that the hardware only supports brownout voltages between 1.1V and 1.75V, and values outside this range should not be programmed. The brownout trip voltage will adjust as the target voltage changes.
4–0 TRG	Voltage level of the external memory supply. The step size of this field is 25 mV. 0x00 = 1.1 V, 0x1A...0x1F = 1.75 V, and the reset value = 1.5 V.

11.12.9 DC-DC Converter 4.2V Control Register (HW_POWER_DCDC4P2)

This register contains controls that need to be adjusted to select the 4.2V source as the input for the dcdc converter

Address: HW_POWER_DCDC4P2 – 8004_4000h base + 80h offset = 8004_4080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DROPOUT_CTRL				RSRVD5		ISTEAL_ THRESH		ENABLE_4P2	ENABLE_DCDC	HYST_DIR	HYST_THRESH	RSRVD3	TRG		
W					▨	▨							▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2			BO					RSRVD1			CMPTRIP				
W	▨	▨	▨						▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

HW_POWER_DCDC4P2 field descriptions

Field	Description
31–28 DROPOUT_ CTRL	Adjusts the behavior of the dcdc converter and 4.2V regulation circuit. The two msbs control the VDD4P2 brownout below the target set by DCDC4p2_trg before the regulation circuit steals battery charge current to support the voltage on VDD4P2. The two lsbs control which power source is selected by the dcdc converter after ENABLE_DCDC is set. 0b11XX: 200mV 0b10XX: 100mV 0b01XX: 50mV 0b00XX: 25mV 0bXX00: DcDc Converter power source is DCDC_4P2 regardless of BATTERY voltage 0bXX01: DcDc converter uses DCDC_4P2 always, and only enables DCDC_BATT when VDD4P2 is less than BATTERY. 0bXX1X: DcDc converter selects either VDD4P2 or BATTERY, which ever is higher.
27–26 RSRVD5	Reserved.
25–24 ISTEAL_ THRESH	Has no effect.
23 ENABLE_4P2	Enables the DCDC_4P2 regulation circuitry. The 4p2V load current has priority over the battery charge current when the sum of the two tries to exceed the limit set with CHARGE_4P2_ILIMIT.
22 ENABLE_DCDC	Enable the dcdc converter to use the DCDC_4P2 pin as a power source based on a voltage comparison between the BATTERY pin voltage and the VDD4P2 pin voltage. The trip point of this comparator is controlled by the CMPTRIP bitfield.
21 HYST_DIR	Enable hysteresis in analog comparator.
20 HYST_THRESH	Increase the threshold detection for DCDC_4P2/BATTERY analog comparator.
19 RSRVD3	Reserved.
18–16 TRG	Regulation voltage of the DCDC_4P2 pin. 0b000 : 4.2V 0b001 : 4.1V 0b010 : 4.0V

HW_POWER_DCDC4P2 field descriptions (continued)

Field	Description
	0b011 : 3.9V 0b1XX : BATTERY
15–13 RSRVD2	Reserved.
12–8 BO	Brownout voltage in 25mV steps for the DCDC_4P2 pin. 0b00000 : 3.6V .. 0b11111 : 4.375V
7–5 RSRVD1	Reserved.
4–0 CMPTRIP	Sets the trip point for the comparison between the DCDC_4P2 and BATTERY pin. When the comparator output is high then, the switching converter may use the DCDC_4P2 pin as the source for the switching converter, otherwise it will use the DCDC_BATT pin. 0b00000 DCDC_4P2 pin >= 0.85 * BATTERY pin 0b00001 DCDC_4P2 pin >= 0.86 * BATTERY pin 0b11000 DCDC_4P2 pin >= BATTERY pin (default) 0b11111 DCDC_4P2 pin >= 1.05 * BATTERY pin

11.12.10 DC-DC Miscellaneous Register (HW_POWER_MISC)

This register contains controls that may need to be adjusted to optimize DC-DC converter performance using the battery voltage information

Address: HW_POWER_MISC – 8004_4000h base + 90h offset = 8004_4090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2[15:7]									FREQSEL			DISABLEFET_BO_LOGIC	DELAY_TIMING	TEST	SEL_PLLCLK
W	▨	▨	▨	▨	▨	▨	▨	▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_MISC field descriptions

Field	Description
31–7 RSRVD2	Reserved.
6–4 FREQSEL	This register will select the PLL-based frequency that the dcdc uses when SEL_PLLCLK is set high. The decode is as follows: 0x0=Reserved 0x1=20MHz 0x2=24MHz 0x3=19.2MHz 0x4=14.4MHz 0x5=18MHz 0x6=21.6MHz 0x7=17.28Mhz .
3 DISABLEFET_ BO_LOGIC	This bit can be used to enable/disable the logic which stops the power-FETs in the DC-DC converter from switching when a brown-out condition is detected.
2 DELAY_TIMING	This bit delays the timing of the output fets in the switching dcdc converter. This may provide improved ground noise performance in high power applications.
1 TEST	Reserved. Do not set.
0 SEL_PLLCLK	This bit selects the source of the clock used for the DC-DC converter. The default is to use the 24-MHz clock. Setting this bit selects the PLL clock as a clock source for the DC-DC converter. It is required to program FREQSEL before setting this bit.

11.12.11 DC-DC Duty Cycle Limits Control Register (HW_POWER_DCLIMITS)

This register defines the upper and lower duty cycle limits of DC-DC. These values depend on details of switching converter implementation and should not be changed without guidance from Freescale.

Address: HW_POWER_DCLIMITS – 8004_4000h base + A0h offset = 8004_40A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2	POSLIMIT_BUCK							RSRVD1	NEGLIMIT						
W	▨								▨							
Reset	0	0	0	0	1	1	0	0	0	1	0	1	1	1	1	1

HW_POWER_DCLIMITS field descriptions

Field	Description
31–16 RSRVD3	Reserved.
15 RSRVD2	Reserved.
14–8 POSLIMIT_BUCK	Upper limit duty cycle limit in DC-DC converter. This field will limit the maximum VDDIO acheivable for a given battery voltage, and it's value may be increased if very low battery operation is desired.
7 RSRVD1	Reserved.
6–0 NEGLIMIT	Negative duty cycle limit of DC-DC converter.

11.12.12 Converter Loop Behavior Control Register (HW_POWER_LOOPCTRL)

This register defines the control loop parameters available for the DC-DC converter.

HW_POWER_LOOPCTRL: 0x0B0

HW_POWER_LOOPCTRL_SET: 0x0B4

HW_POWER_LOOPCTRL_CLR: 0x0B8

HW_POWER_LOOPCTRL_TOG: 0x0BC

Address: HW_POWER_LOOPCTRL – 8004_4000h base + B0h offset = 8004_40B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3											TOGGLE_DIF	HYST_SIGN	EN_CM_HYST	EN_DF_HYST	CM_HYST_THRESH
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DF_HYST_THRESH	RCSCALE_THRESH	EN_RCSCALE		RSRVD2	DC_FF			DC_R				RSRVD1		DC_C	
W					▨								▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1

HW_POWER_LOOPCTRL field descriptions

Field	Description
31–21 RSRVD3	Reserved.
20 TOGGLE_DIF	Set high to enable supply stepping to change only after the differential control loop has toggled as well. This should eliminate any chance of large transients when supply voltage changes are made.
19 HYST_SIGN	Invert the sign of the hysteresis in DC-DC analog comparators. This bit should set when using PFM mode.
18 EN_CM_HYST	Enable hysteresis in switching converter common mode analog comparator. This feature will improve transient supply ripple and efficiency.
17 EN_DF_HYST	Enable hysteresis in switching converter differential mode analog comparators. This feature will improve transient supply ripple and efficiency.
16 CM_HYST_ THRESH	Increase the threshold detection for common mode analog comparator.
15 DF_HYST_ THRESH	Increase the threshold detection for common mode analog comparator.
14 RCSCALE_ THRESH	Increase the threshold detection for RC scale circuit.
13–12 EN_RCSCALE	Enable analog circuit of DC-DC converter to respond faster under transient load conditions. 00: disabled 01: 2X increase 10: 4X increase

HW_POWER_LOOPCTRL field descriptions (continued)

Field	Description
	11: 8X increase
11 RSRVD2	Reserved.
10–8 DC_FF	Two's complement feed forward step in duty cycle in the switching DC-DC converter. Each time this field makes a transition from 0x0, the loop filter of the DC-DC converter is stepped once by a value proportional to the change. This can be used to force a certain control loop behavior, such as improving response under known heavy load transients.
7–4 DC_R	Magnitude of proportional control parameter in the switching DC-DC converter control loop.
3–2 RSRVD1	Reserved.
1–0 DC_C	Ratio of integral control parameter to proportional control parameter in the switching DC-DC converter, and can be used to optimize efficiency and loop response. 00: Maximum 01: Decrease ratio 2X 10: Decrease ratio 4X 11: Lowest ratio.

11.12.13 Power Subsystem Status Register (HW_POWER_STS)

This register contains status information for the battery charger, DCDC converter and USB/OTG connections.

Address: HW_POWER_STS – 8004_4000h base + C0h offset = 8004_40C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3		PWRUP_SOURCE						RSRVD2		PSWITCH		THERMAL_WARNING	VDDMEM_BO	AVALID0_STATUS	BVALID0_STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VBUSVALID0_ STATUS	SESSEND0_ STATUS	BATT_BO	VDD5V_FAULT	CHRGSTS	DCDC_4P2_BO	DC_ OK	VDDIO_BO	VDDA_BO	VDDD_BO	VDD5V_GT_ VDDIO	VDD5V_ DROOP	AVALID0	BVALID0	VBUSVALID0	SESSEND0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_POWER_STS field descriptions

Field	Description
31–30 RSRVD3	Reserved.
29–24 PWRUP_ SOURCE	These read-only bits determine which source was active when the dcdc converter powerup sequence was complete. This can be used to determine what event caused the device to powerup. bit5 : five volts bit4 : rtc wakeup bit3 : reserved bit2 : reserved bit1 : high level pswitch voltage bit0 : midlevel pswitch voltage
23–22 RSRVD2	Reserved.
21–20 PSWITCH	These read-only bits reflect the current state of the pswitch comparators. The lsb is high when voltage on the PSWITCH pin is above 0.8V, and the msb is high when the voltage on the PSWITCH pin is above 1.75V
19 THERMAL_ WARNING	Asserting this signal is a last warning to the cpu before the temp sensor shuts down the power system . It will be asserted at a small temperature offset below the chip thermal limit.
18 VDDMEM_BO	Output of VDDMEM brownout comparator. High when a brownout is detected. This comparator's power-up/power-down follows the VDDMEM regulator power-up.
17 AVALID0_ STATUS	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software like the AVALID bit below.
16 BVALID0_ STATUS	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the BVALID bit below.
15 VBUSVALID0_ STATUS	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the VBUSVALID bit below.

HW_POWER_STS field descriptions (continued)

Field	Description
14 SESSEND0_ STATUS	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V presents, 1 if VDD5V not presents.
13 BATT_BO	Output of battery brownout comparator.
12 VDD5V_FAULT	Battery charging fault status. If the battery charger is not powered down, the bit is high when the 5V supply falls below the battery voltage. If the charger is powered down, the bit asserts high when 5V falls to below roughly VDDIO/2. If the charger is not powered down, the bit is sticky and remains set until the PWD_CHARGE_4P2 bit is cycled. Otherwise, the bit is cleared when 5V is restored. NOTE: The default value depends on whether VDD5V is present, 0 if 5V presents, 1 if 5V not presents.
11 CHRGSTS	Battery charging status. High during Li-Ion battery charge until the charging current falls below the STOP_ILIMIT threshold.
10 DCDC_4P2_BO	Output of the brownout comparator on the DCDC_4P2 pin.
9 DC_OK	High when switching DC-DC converter control loop has stabilized after a voltage target change. When linear regulators are active, this bit goes high when the actual voltage is above the target voltage. Therefore, DC_OK will go high when changing a linear regulator output to a lower value before the actual voltage decreases to the new target value.
8 VDDIO_BO	Output of VDDIO brownout comparator. High when a brownout is detected. This comparator defaults powered up, but can be powered down through the POWER_MINPWR register.
7 VDDA_BO	Output of VDDA brownout comparator. High when a brownout is detected. It is not possible to power-down this comparator.
6 VDDD_BO	Output of VDDD brownout comparator. High when a brownout is detected. It is not possible to power-down this comparator.
5 VDD5V_GT_ VDDIO	Indicates the voltage on the VDD5V pin is higher than VDDIO by a Vt voltage, nominally 500 mV.
4 VDD5V_DROOP	Indicates the voltage on the VDD5V pin is below the VBUSDROOP_TRSH defined in the 5VCTRL register.
3 AVALID0	Indicates VBus is above the VA_SESS_VLD threshold, that is, high if VBus greater than 2.0, low if VBus less than 0.8, otherwise unknown.
2 BVALID0	Indicates VBus is above the VB_SESS_VLD threshold, high if VBus greater than 4.0, low if VBus less than 0.8, otherwise unknown.
1 VBUSVALID0	Accurate detection of the presence of 5v power. This can be used for detection of 5v in all modes of operation including USB OTG. See POWER_5VCTRL to enable and set threshold for comparison.
0 SESSEND0	Indicates VBus is below the VB_SESS_END threshold, that is, 0 if VBus is greater than 0.8 V, 1 if VBus is less than 0.2 V, otherwise unknown. See POWER_5VCTRL to enable comparators. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V presents, 1 if VDD5V not presents.

11.12.14 Transistor Speed Control and Status Register (HW_POWER_SPEED)

This register contains the setup and controls needed to measure silicon speed.

HW_POWER_SPEED: 0x0D0

HW_POWER_SPEED_SET: 0x0D4

HW_POWER_SPEED_CLR: 0x0D8

HW_POWER_SPEED_TOG: 0x0DC

Address: HW_POWER_SPEED – 8004_4000h base + D0h offset = 8004_40D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1								STATUS[23:16]							
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STATUS[15:8]								STATUS_SEL		RSRVD0				CTRL	
W	▨										▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

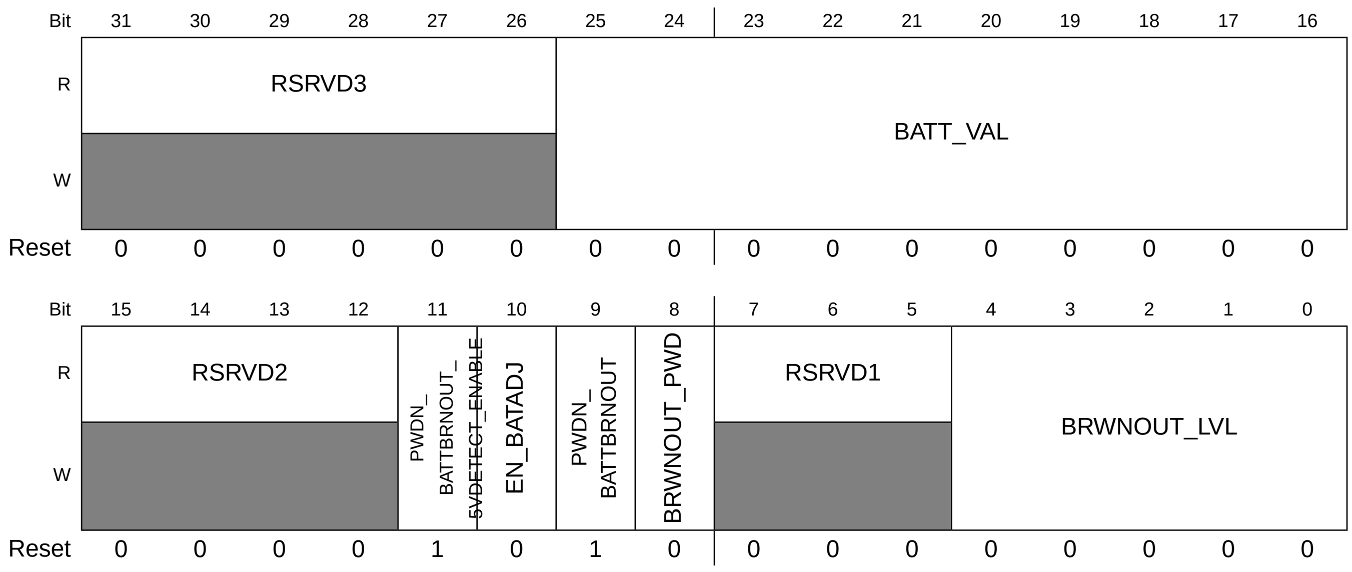
HW_POWER_SPEED field descriptions

Field	Description
31–24 RSRVD1	Reserved.
23–8 STATUS	Result from the speed sensor. This result is only valid when SPEEDCTRL=0b11; otherwise this field contains debug information from the switching DC-DC converter.
7–6 STATUS_SEL	Speed Sensor Status Select. This defines the meaning on the STATUS field in this register. 0x0 **DCDC_STAT** — DC-DC speed sensor (lower 8 bits) 0x1 **CORE_STAT** — core logic speed sensor 0x2 **ARM_STAT** — arm9 speed sensor
5–2 RSRVD0	Reserved.
1–0 CTRL	Speed Control bits. 00: Speed sensor off, 0b01: Speed sensor enabled, 11: Enable speed sensor measurement. Every time a measurement is taken, the sequence of 0x00 ; 01 ; 11 must be repeated. This sequence should proceed no faster than 1.5 MHz to ensure proper operation.

11.12.15 Battery Level Monitor Register (HW_POWER_BATTMONITOR)

This register provides brownout controls and monitors the battery voltage.

Address: HW_POWER_BATTMONITOR – 8004_4000h base + E0h offset = 8004_40E0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD3
BATT_VAL
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RSRVD2
PWDN_ BATTBRNOUT_ 5VDETECT_ENABLE
EN_BATADJ
PWDN_ BATTBRNOUT
BRWNOUT_PWD
RSRVD1
BRWNOUT_LVL
Reset 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0

HW_POWER_BATTMONITOR field descriptions

Field	Description
31–26 RSRVD3	Reserved.
25–16 BATT_VAL	Software should be configured to place the battery voltage in this register measured with an 8-mV LSB resolution through the LRADC. This value is used by the DC-DC converter and must be correct before setting EN_BATADJ.
15–12 RSRVD2	Reserved.
11 PWDN_ BATTBRNOUT_ 5VDETECT_ ENABLE	Enable the use of the programmed 5V detect signal to gate off the powering down of the device when a battery brownout occurs. Clear this bit to power-down the device when a battery brown-out occurs regardless of the state of the 5 V detect signal.
10 EN_BATADJ	This bit enables the DC-DC to improve efficiency and minimize ripple using the information from the BATT_VAL field. It is very important that BATT_VAL contain accurate information before setting EN_BATADJ. Note: HW_LRADC_CONVERSION_SCALE_FACTOR must be programmed to 0x2 before setting this bit to 0x1.
9 PWDN_ BATTBRNOUT	Powers down the device after the DC-DC converter completeS startup if a battery brownout occurs. This function is only active when 5V is not present. Additionally, software should clear this bit and disable this function after the system is configured for a battery brownout and the battery brownout interrupt is enabled.
8 BRWNOUT_PWD	Power-down circuitry for battery brownout detection. This bit should only be set when it is not important to montior battery brownouts and minimum system power consumption is required.

HW_POWER_BATTMONITOR field descriptions (continued)

Field	Description
7–5 RSRVD1	Reserved.
4–0 BRWNOUT_LVL	The default setting of the brownout settings decode to a voltage as follows: Li-Ion = 2.4 V The voltage level can be calculated for other values by the following equation: Li-Ion brownout voltage = 2.4 V + 0.04 * BRWNOUT_LVL

11.12.16 Power Module Reset Register (HW_POWER_RESET)

This register allows software to put the chip into the off state.

HW_POWER_RESET: 0x100

HW_POWER_RESET_SET: 0x104

HW_POWER_RESET_CLR: 0x108

HW_POWER_RESET_TOG: 0x10C

Address: HW_POWER_RESET – 8004_4000h base + 100h offset = 8004_4100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNLOCK															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1													FASTFALLPSWITCH_OFF	PWD_OFF	PWD
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_RESET field descriptions

Field	Description
31–16 UNLOCK	Write 0x3E77 to unlock this register and allow other bits to be changed. NOTE: This register must be unlocked on a write-by-write basis, so the UNLOCK bitfield must contain the correct key value during all writes to this register in order to update any other bitfield values in the register. 0x3E77 **KEY** — Key needed to unlock HW_POWER_RESET register.
15–3 RSRVD1	Reserved.
2 FASTFALLPSWITCH_ OFF	Set this bit to 1'b1 to disable the chip shutting down by a fast falling edge of PSWITCH. Setting this bit will be useful for preventing fast falling edges on the PSWITCH pin from resetting the chip. It may also be useful increasing system tolerance of noisy EMI environments.
1 PWD_OFF	Optional bit to disable all paths to power off the chip except the watchdog timer. Setting this bit will be useful for preventing fast falling edges on the PSWITCH pin from resetting the chip. It may also be useful increasing system tolerance of noisy EMI environments.
0 PWD	Powers down the chip.

11.12.17 Power Module Debug Register (HW_POWER_DEBUG)

Debug Register.

HW_POWER_DEBUG: 0x110

HW_POWER_DEBUG_SET: 0x114

HW_POWER_DEBUG_CLR: 0x118

HW_POWER_DEBUG_TOG: 0x11C

Address: HW_POWER_DEBUG – 8004_4000h base + 110h offset = 8004_4110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:4]												VBUSVALIDPIOLOCK	AVALIDPIOLOCK	BVALIDPIOLOCK	SESSENDPIOLOCK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_DEBUG field descriptions

Field	Description
31–4 RSRVD0	Reserved.
3 VBUSVALIDPIOLOCK	Reserved for Freescale Debugging Purposes Only.
2 AVALIDPIOLOCK	Reserved for Freescale Debugging Purposes Only.
1 BVALIDPIOLOCK	Reserved for Freescale Debugging Purposes Only.
0 SESSENDPIOLOCK	Reserved for Freescale Debugging Purposes Only.

11.12.18 Power Module Thermal Reset Register (HW_POWER_THERMAL)

Thermal Reset Register.

HW_POWER_THERMAL: 0x120

HW_POWER_THERMAL_SET: 0x124

HW_POWER_THERMAL_CLR: 0x128

HW_POWER_THERMAL_TOG: 0x12C

This register contains the controls for the thermal temp sensor that can be used to reset the chip when a temperature threshold is reached.

Address: HW_POWER_THERMAL – 8004_4000h base + 120h offset = 8004_4120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:9]							TEST	PWD	LOW_POWER	OFFSET_ADJ		OFFSET_ADJ_ENABLE	TEMP_THRESHOLD		
W	▨															
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

HW_POWER_THERMAL field descriptions

Field	Description
31–9 RSRVD0	Reserved.
8 TEST	This bit is used for debug purposes only.
7 PWD	Power-down the thermal temp sensor block.
6 LOW_POWER	Set to operate thermal temp sensor with less power.
5–4 OFFSET_ADJ	Add offset to the programmed trip point. 00 adds 5degC, 01 adds 10degC, 10 subtracts 5degC, 11 subtracts 10degC. These are valid only when HW_POWER_THERMAL_OFFSET_ADJ_ENABLE=1
3 OFFSET_ADJ_ ENABLE	Enable the offset adjustment capability.
2–0 TEMP_ THRESHOLD	This field programs the thermal reset trip point. A value of 000 sets to 115degC and 111 sets to 150degC. The intermediate values are in steps of 5degC. When TEST=1, all the trip points are set to 65degC less than their programmed values.

11.12.19 Power Module USB1 Manual Controls Register (HW_POWER_USB1CTRL)

USB1 control Register.

HW_POWER_USB1CTRL: 0x130

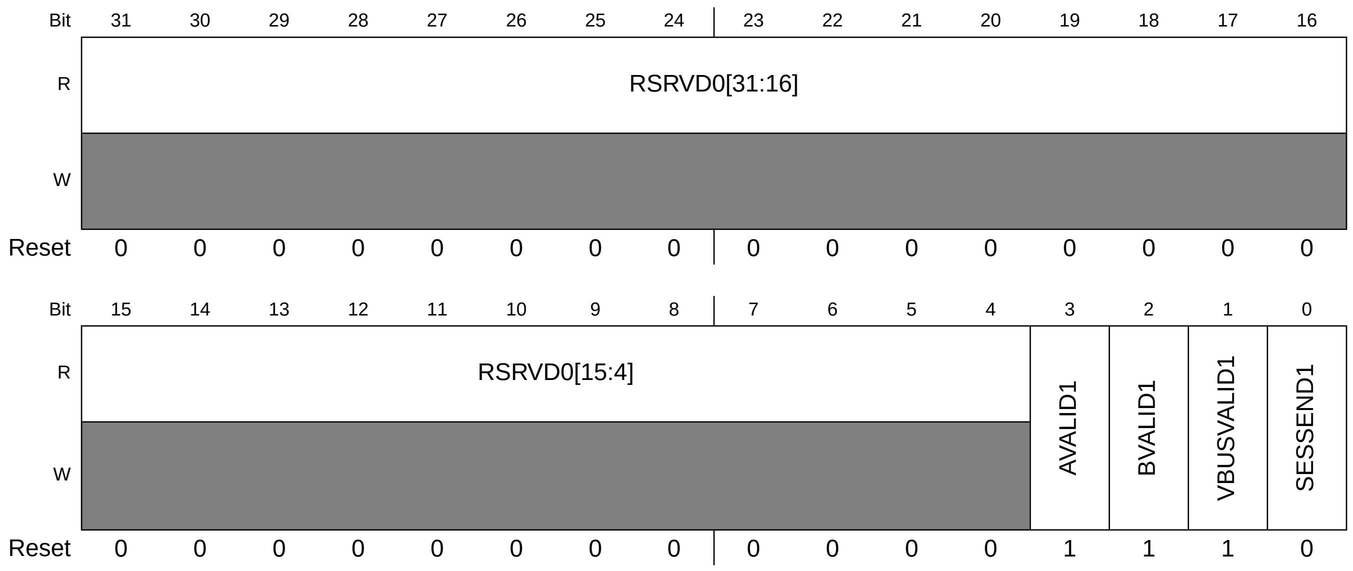
HW_POWER_USB1CTRL_SET: 0x134

HW_POWER_USB1CTRL_CLR: 0x138

HW_POWER_USB1CTRL_TOG: 0x13C

This register contains the connection controls for the USB instance 1.

Address: HW_POWER_USB1CTRL – 8004_4000h base + 130h offset = 8004_4130h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD0[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD0[15:4]
AVALID1
BVALID1
VBUSVALID1
SESSEND1
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0

HW_POWER_USB1CTRL field descriptions

Field	Description
31–4 RSRVD0	Reserved.
3 AVALID1	This bit indicates the value for the sessend signal for USB instance 1. It is analogous to the HW_POWER_STS_AVALID0 bit but this bit is not set by the hardware. It is controlled by software.
2 BVALID1	This bit indicates the value for the sessend signal for USB instance 1. It is analogous to the HW_POWER_STS_BVALID0 bit but this bit is not set by the hardware. It is controlled by software.
1 VBUSVALID1	This bit indicates the value for the vbusvalid signal for USB instance 1. It is analogous to the HW_POWER_STS_VBUSVALID0 bit but this bit is not set by the hardware. It is controlled by software.
0 SESSEND1	This bit indicates the value for the sessend signal for USB instance 1. It is analogous to the HW_POWER_STS_SESSEND0 bit but this bit is not set by the hardware. It is controlled by software.

11.12.20 Power Module Special Register (HW_POWER_SPECIAL)

Special test functionality.

HW_POWER_SPECIAL: 0x140

HW_POWER_SPECIAL_SET: 0x144

HW_POWER_SPECIAL_CLR: 0x148

HW_POWER_SPECIAL_TOG: 0x14C

Address: HW_POWER_SPECIAL – 8004_4000h base + 140h offset = 8004_4140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEST																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_SPECIAL field descriptions

Field	Description
31–0 TEST	Reserved for Freescale Debugging Purposes Only

11.12.21 Power Module Version Register (HW_POWER_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

Address: HW_POWER_VERSION – 8004_4000h base + 150h offset = 8004_4150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

11.12.22 Analog Clock Control Register (HW_POWER_ANACLKCTRL)

HW_POWER_ANACLKCTRL: 0x160

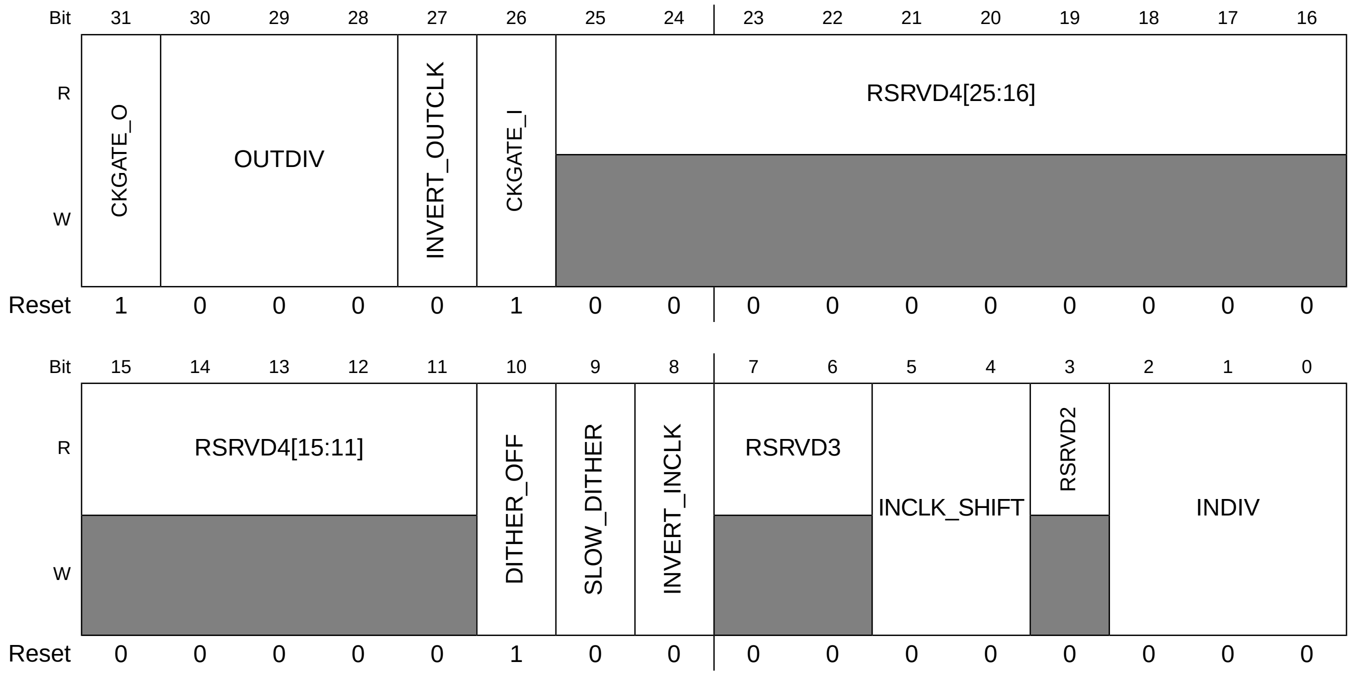
HW_POWER_ANACLKCTRL_SET: 0x164

HW_POWER_ANACLKCTRL_CLR: 0x168

HW_POWER_ANACLKCTRL_TOG: 0x16C

This register provides analog clock control.

Address: HW_POWER_ANACLKCTRL – 8004_4000h base + 160h offset = 8004_4160h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
CKGATE_O
OUTDIV
INVERT_OUTCLK
CKGATE_I
RSRVD4[25:16]
Reset 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD4[15:11]
DITHER_OFF
SLOW_DITHER
INVERT_INCLK
RSRVD3
INCLK_SHIFT
RSRVD2
INDIV
Reset 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

HW_POWER_ANACLKCTRL field descriptions

Field	Description
31 CKGATE_O	Analog Output Clock Gate, Reserved for Freescale Debugging Purposes Only
30–28 OUTDIV	Analog Output Clock Divider, Reserved for Freescale Debugging Purposes Only
27 INVERT_ OUTCLK	Analog Output Clock Invert, Reserved for Freescale Debugging Purposes Only
26 CKGATE_I	Analog Input Clock Gate, Reserved for Freescale Debugging Purposes Only
25–11 RSRVD4	Reserved
10 DITHER_OFF	Clock Dither Disable, Reserved for Freescale Debugging Purposes Only
9 SLOW_DITHER	Set 0x1 to make the configuration INDIV<1:0> and INCLK_SHIFT available, but there would be <= 0.67us timing delay between HSADC start conversion and 'start run' command from software trigger or PWM trigger. If this delay is not desired, it is recommended to set INDIV<1:0> to 0x2 and using PWM trigger for HSADC.
8 INVERT_INCLK	Analog Input Clock Invert, Reserved for Freescale Debugging Purposes Only

HW_POWER_ANACLKCTRL field descriptions (continued)

Field	Description
7–6 RSRVD3	Reserved
5–4 INCLK_SHIFT	Recommended value 0x2 when HSADC works at 1.5 MHz sampling rate with DCDC working. This configuration can attenuate the impact of DCDC noise on HSADC performance only when HSADC working at 1.5 MHz sampling rate, and the configuration is available only when set SLOW_DITHER to 0x1.
3 RSRVD2	Reserved
2–0 INDIV	INDIV<2>: Set 1 for HSADC working at 1.5 MHz sampling rate. INDIV<1:0>: Recommended value 0x2 when HSADC works at 1.5 MHz sampling rate with DCDC working. This configuration can attenuate the impact of DCDC noise on HSADC performance only when HSADC working at 1.5 MHz sampling rate, and the configuration is available when set SLOW_DITHER to 0x1 or using PWM trigger for HSADC.

11.12.23 POWER Reference Control Register (HW_POWER_REFCTRL)

This register provides control bits for changing analog voltage and current references.

HW_POWER_REFCTRL: 0x170

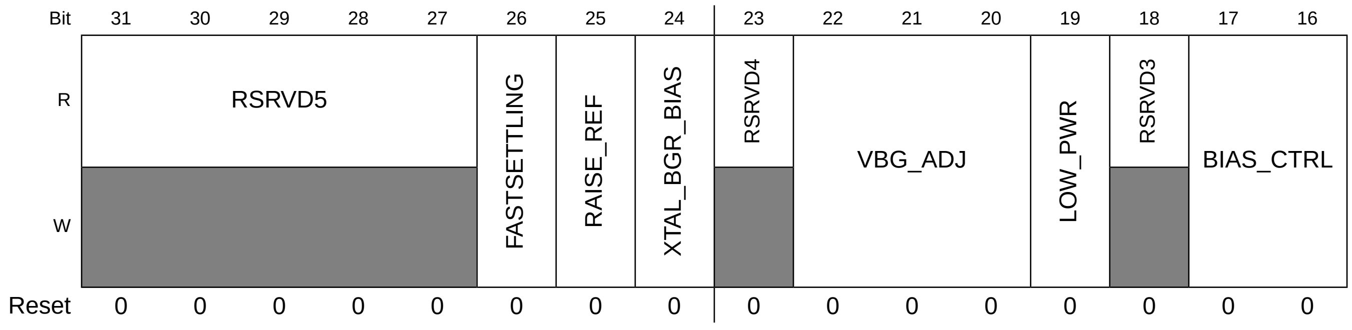
HW_POWER_REFCTRL_SET: 0x174

HW_POWER_REFCTRL_CLR: 0x178

HW_POWER_REFCTRL_TOG: 0x17C

The power reference control register provides control over the voltage and power for the analog circuits.

Address: HW_POWER_REFCTRL – 8004_4000h base + 170h offset = 8004_4170h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2	VDDXTAL_TO_VDDD	ADJ_ ANA	ADJ_ VAG	ANA_REFVAL				VAG_VAL				RSRVD1			
W	▨												▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_POWER_REFCTRL field descriptions

Field	Description
31–27 RSRVD5	Reserved. Always write zeroes to this bit field.
26 FASTSETTLING	Set to high to detect the existence of battery. After this bit is set to high, if no battery brownout is detected, then the battery really exists. Reset this bit to low after complete detecting.
25 RAISE_REF	Reserved for Freescale Debugging Purposes Only
24 XTAL_BGR_BIAS	Switch the XTAL bias from self-bias to bandgap-based bias current. Also switches the source of the XTAL supply to the core supply to save power. Note that this bit is not reset by the POWER's SFTRST bit. It is reset by a power-on reset only, and CLKGATE has no effect on reads/writes.
23 RSRVD4	Reserved. Always write zeroes to this bit field.
22–20 VBG_ADJ	Small adjustment for VBG value. Will affect ALL reference voltages. Expected to be used to tweak final Li-Ion charge voltage. 000=Nominal. 001=+0.3%. 010=+0.6%. 011=0.85%. 100=-0.3%. 101=-0.6%. 110=-0.9%. 111=-1.2%. Note that this bit is not reset by the POWER's SFTRST bit. It is reset by a power-on reset only, and CLKGATE has no effect on reads/writes.
19 LOW_PWR	Lowers power (~100 uA) in the bandgap amplifier. This mode is useful in USB suspend or standby when bandgap accuracy is not critical. Note that this bit is not reset by the POWER's SFTRST bit. It is reset by a power-on reset only, and CLKGATE has no effect on reads/writes.
18 RSRVD3	Reserved. Always write zeroes to this bit field.
17–16 BIAS_CTRL	Bias current control for all analog blocks: 00=Nominal. 01=-20%. 10=-10%. 11=+10%. Note that this bit not reset by the POWER's SFTRST bit. It is reset by a power-on reset only, and CLKGATE has no effect on reads/writes. These bits should only be used for test/debug, and not in an application.
15 RSRVD2	Reserved. Always write zeroes to this bit field.
14 VDDXTAL_TO_ VDDD	Shorts the supply of the XTAL oscillator to VDDD. This bit may be used to reduce power consumption, but should only be used on the advice of Freescale.
13 ADJ_ANA	Reserved for Freescale Debugging Purposes Only
12 ADJ_VAG	Reserved for Freescale Debugging Purposes Only

HW_POWER_REFCTRL field descriptions (continued)

Field	Description
11–8 ANA_REFVAL	Reserved for Freescale Debugging Purposes Only
7–4 VAG_VAL	Reserved for Freescale Debugging Purposes Only
3–0 RSRVD1	Reserved. Always write zeroes to this bit field.

Chapter 12
Boot Modes

12.1 Overview

The i.MX28 boot process begins at Power On Reset (POR) where the hardware reset logic forces the ARM core to begin execution starting from the on-chip boot ROM. The boot ROM logic reads hardware inputs such as the boot pins on the IC or OCOTP bits to determine the boot flow behavior of the i.MX28.

The main features of the ROM include:

- Support for booting from various boot devices
- USB recovery support
- Encrypted boot image
- Device Configuration Data (DCD) support in boot image
- Digital signature based High Assurance Boot (HAB)
- Secondary boot from NAND and SD/MMC boot devices

The i.MX28 boot ROM supports the following boot devices:

- NAND Flash
- ONFI 2.x BA-NAND
- SD/eSD/MMC/eMMC
- Serial ROM devices, including SPI NOR/EEPROM and I2C EEPROM

Secure Boot

A key feature of the i.MX28 ROM is the ability to perform a secure boot. This is supported by

- The encrypted boot feature and
- The High Assurance Boot (HAB)

These are independent features and are not dependent on each other.

The encrypted boot feature is an inherent element of the SB boot format used by the i.MX28 ROM. The encrypted boot feature is easily enabled through the use of tools provided by Freescale to provide image confidentiality..

A HAB secure boot makes use of a security library which is a subcomponent of the i.MX28 ROM. The HAB uses a combination of hardware and software together with a Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized programs. HAB differs from the encrypted boot feature in that image authentication is performed. Before the HAB allows a user's image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as part of the final Program Image. If configured to do so, the i.MX28 verifies the signatures using the public keys included in the Program Image. A secure boot with HAB can be performed on all boot devices supported on i.MX28. The HAB library in the i.MX28 boot ROM also provides API functions, allowing additional boot chain components (bootloaders) to extend the secure boot chain. The out-of-fab setting for SEC_CONFIG is the Open configuration in which the ROM/HAB performs image authentication but all authentication errors are ignored and the image is still allowed to execute.

The remainder of this chapter provides information on above features and boot modes and how to configure and use boot features of i.MX28 ROM.

12.2 Boot Modes

Table 12-1 lists all the boot modes supported by i.MX28 ROM. The boot mode can be selected either through external resistors or through OTP eFuse bit programming.

Table 12-1. Boot Modes

PORT	BOOT MODE
USB	Encrypted/unencrypted USB slave boot mode.
I^2C	Encrypted/unencrypted I^2C0 master—Boots from 1.8- and 3.3-V EEPROM.
SPI2	Encrypted/unencrypted SPI2 master from SSP2—Boots from 1.8- and 3.3-V flash memory.
SPI3	Encrypted/unencrypted SPI3 master from SSP3—Boots from 1.8- and 3.3-V flash and EEPROM.
SSP0	Encrypted/unencrypted SD/MMC master from SSP1—Boots from 1.8- and 3.3-V 1-bit, 4-bit, and 8-bit SD/MMC/eSD/eMMC cards.
SSP1	Encrypted/unencrypted SD/MMC master from SSP2—Boots from 1.8- and 3.3-V 1-bit, 4-bit, and 8-bit SD/MMC/eSD/eMMC cards.
GPMI	Encrypted/unencrypted NAND, 1.8- and 3.3-V, 8-bit wide, BCH2 to BCH20.
JTAG	Wait JTAG connection

12.2.1 Boot Mode Selection Map

Table 12-2. Boot Mode Selection Map

LCD_ DATA[5]	VOLTAGE SELECT-OR/ LCD_ DATA[4]	BM3/ LCD_ DATA[3]	BM2/ LCD_ DATA[2]	BM1/ LCD_ DATA[1]	BM0/ LCD_ DATA[0]	PORT	BOOT MODE
x	x	0	0	0	0	USB0	USB (unencrypted vs. encrypted is under OTP control)
x	0	0	0	0	1	I2C0	I2C0 master, 3.3 V
x	1	0	0	0	1	I2C0	I2C0 master, 1.8 V
x	0	0	0	1	0	SPI2	SPI master SSP2 boot from flash, 3.3 V
x	1	0	0	1	0	SPI2	SPI master SSP2 boot from flash, 1.8 V
x	0	0	0	1	1	SPI3	SPI master SSP3 boot from flash, 3.3 V
x	1	0	0	1	1	SPI3	SPI master SSP3 boot from flash, 1.8 V
x	0	0	1	0	0	GPMI	NAND, 3.3 V
x	1	0	1	0	0	GPMI	NAND, 1.8 V
x	0	0	1	0	1		Reserved
x	0	0	1	1	0	JTAG	Wait JTAG connection mode
x	0	0	1	1	1		Reserved
x	0	1	0	0	0	SPI3	SPI master SSP3 boot from EEP-ROM, 3.3 V
x	1	1	0	0	0	SPI3	SPI master SSP3 boot from EEP-ROM, 1.8 V
x	0	1	0	0	1	SSP0	SD/MMC master on SSP0, 3.3 V
x	1	1	0	0	1	SSP0	SD/MMC master on SSP0 1.8 V
x	0	1	0	1	0	SSP1	SD/MMC master on SSP1, 3.3 V
x	1	1	0	1	0	SSP1	SD/MMC master on SSP1, 1.8 V
x	0	1	0	1	1		Reserved

LCD_ DATA[5]	VOLTAGE SELECT-OR/ LCD_ DATA[4]	BM3/ LCD_ DATA[3]	BM2/ LCD_ DATA[2]	BM1/ LCD_ DATA[1]	BM0/ LCD_ DATA[0]	PORT	BOOT MODE
x	0	1	1	0	0		Reserved
x	0	1	1	0	1		Reserved
x	0	1	1	1	0		Reserved
x	0	1	1	1	1		Reserved

12.3 OTP eFuse and Persistent Bit Definitions

This section provides tables that show the location and function of the OTP eFuse and persistent bits.

12.3.1 OTP eFuse

The device contains a 1280-bit array of OTP eFuse bits, some of which are used by ROM. The bits listed in Table 12-3 – Table 12-5 can be configured by the customers and are typically programmed on the customer's board assembly line. For more information about the OTP bits, see OCOTP Overview.

Table 12-3. General ROM Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM0:0x8002C1A0:31:24	Boot Modes OCOTP bits, please refer to table12-3 for definitions
HW_OCOTP_ROM0:0x8002C1A0:23:22	SD_MMC_MODE[1:0] 00 = MBR 01 = RESERVED 10 = eMMC fast boot 11 = eSD fast boot
HW_OCOTP_ROM0:0x8002C1A0:21:20	POWER_GATE_GPIO-SD/MMC card power gate GPIO pin select. 00 = PWM3 01 = PWM4 10 = LCD_DOTCLK 11 = NO_GATE

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM0:0x8002C1A0:19:14	POWER_UP_DELAY—SD/MMC card power up delay required after enabling GPIO power gate. 000000 = 20 ms (default) 000001 = 10 ms 000010 = 20 ms …. 111111 = 630 ms
HW_OCOTP_ROM0:0x8002C1A0:13:12	SD_BUS_WIDTH—SD/MMC card bus width. 00 = 4-bit 01 = 1-bit 10 = 8-bit 11 = Reserved
HW_OCOTP_ROM0:0x8002C1A0:11:8	Index to SSP clock speed. By default (0x0), the clock speed is set to 12 MHz. The value of the index will modify the SPI clock speed accordingly.
HW_OCOTP_ROM0:0x8002C1A0:7	EMMC_USE_DDR - Blow to enable DDR access mode when fast boot from eM-MC4.4 card.
HW_OCOTP_ROM0:0x8002C1A0:6	DISABLE_SPI_NOR_FAST_READ—Blow to disable SPI NOR fast reads, which are used by default. Some SPI NORs do not support this functionality.
HW_OCOTP_ROM0:0x8002C1A0:5	ENABLE_USB_BOOT_SERIAL_NUMBER—If set, the device serial number is reported to the host during USB boot mode initialization, else no serial number is reported.
HW_OCOTP_ROM0:0x8002C1A0:4	ENABLE_UNENCRYPTED_BOOT—If clear, allows only booting of encrypted im-ages. If set, both encrypted/unencrypted images are valid.
HW_OCOTP_ROM0:0x8002C1A0:3	Reserved.
HW_OCOTP_ROM0:0x8002C1A0:2	DISABLE_RECOVERY_MODE—If set, does not allow booting in recovery mode.
HW_OCOTP_ROM0:0x8002C1A0:1:0	USE_ALT_DEBUG_UART_PINS[1:0] 00 - I2C0 01 - AUART0 10 - PWM 11 - reserved [note] on EVK/Armadillo, PWM0/1 are used as Debug UART Pins, so these two bits must be blown "10".

Table 12-4. NAND/SD-MMC-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM1:0x8002C1B0:30	DISABLE_SECONDARY_BOOT - Blow this bit to disable the secondary boot.
HW_OCOTP_ROM1:0x8002C1B0:29	SSP3_EXT_PULLUP - This bit is blown to enable external pull up of SSP3 signals.
HW_OCOTP_ROM1:0x8002C1B0:28	SSP2_EXT_PULLUP - This bit is blown to enable external pull up of SSP2 signals.
HW_OCOTP_ROM1:0x8002C1B0:27	ENABLE_NAND7_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE7 and GPMI_RDY7 pins.
HW_OCOTP_ROM1:0x8002C1B0:26	ENABLE_NAND6_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE6 and GPMI_RDY6 pins.
HW_OCOTP_ROM1:0x8002C1B0:25	ENABLE_NAND5_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE5 and GPMI_RDY5 pins.
HW_OCOTP_ROM1:0x8002C1B0:24	ENABLE_NAND4_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE4 and GPMI_RDY4 pins.
HW_OCOTP_ROM1:0x8002C1B0:23	ENABLE_NAND3_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE3 and GPMI_RDY3 pins.
HW_OCOTP_ROM1:0x8002C1B0:22	ENABLE_NAND2_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE2 and GPMI_RDY2 pins.
HW_OCOTP_ROM1:0x8002C1B0:21	ENABLE_NAND1_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE1 and GPMI_RDY1 pins.
HW_OCOTP_ROM1:0x8002C1B0:20	ENABLE_NAND0_CE_RDY_PULLUP - If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE0 and GPMI_RDY0 pins.
HW_OCOTP_ROM1:0x8002C1B0:19	UNTOUCH_INTERNAL_SSP_PULLUP - If this bit is blown then internal pull-ups for SSP are neither enabled nor disabled. This bit is used only if the external pull-ups are implemented and ROM1:18 and/or ROM1:17 are blown.
HW_OCOTP_ROM1:0x8002C1B0:18	SSP1_EXT_PULLUP - Blow to indicate external pull-ups implemented for SSP1.
HW_OCOTP_ROM1:0x8002C1B0:17	SSP0_EXT_PULLUP - Blow to indicate external pull-ups implemented for SSP0.
HW_OCOTP_ROM1:0x8002C1B0:16	SD_INCREASE_INIT_SEQ_TIME - Blow to increase the SD card initialization sequence time from 1 ms (default) to 4 ms.
HW_OCOTP_ROM1:0x8002C1B0:15	SD_INIT_SEQ_2_ENABLE - Blow to enable the second initialization sequence for SD boot.
HW_OCOTP_ROM1:0x8002C1B0:14	SD_CMD0_DISABLE - Cmd0 (reset cmd) is called by default to reset the SD card during startup. Blow this bit to not to reset the card during SD boot.
HW_OCOTP_ROM1:0x8002C1B0:13	SD_INIT_SEQ_1_DISABLE - Blow to disable the first initialization sequence for SD.

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM1:0x8002C1B0:11:8	BOOT_SEARCH_COUNT - Number of 64-page blocks skipped by the NAND driver when searching for information saved into the NAND (see NAND Boot Mode for details). Default value of 0 means 4 blocks to skip.
HW_OCOTP_ROM1:0x8002C1B0:7:4	BOOT_SEARCH_STRIDE - the value of these bits is multiplied by 64 to get boot search stride. The default is 1 boot search stride, that is, 64 pages.
HW_OCOTP_ROM1:0x8002C1B0:2:0	NUMBER_OF_NANDS - Indicates the number of external NANDs. A 0 value means that the NAND driver will scan the chip selects to dynamically find the correct number of NANDs.

Table 12-5. USB-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM2:0x8002C1C0:31:16	USB_VID—Vendor ID used in recovery mode. If the field is 0, Freescale vendor ID is used.
HW_OCOTP_ROM2:0x8002C1C0:15:0	USB_PID—Product ID used in recovery mode.

Table 12-6. eMMC Fast Boot-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM3:0x8002C1D0:11	FAST_BOOT_ACK—Enable the fast boot acknowledge.
HW_OCOTP_ROM3:0x8002C1D0:10	ALT_FAST_BOOT—Enable the alternative fast boot mode.

Table 12-7. NAND Access-Related Bits

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM4:0x8002C1E0:31	NAND_BADBLOCK_MARKER_RESERVE If this bit is blown, the factory bad block marker preservation will be disabled.
HW_OCOTP_ROM4:0x8002C1E0:23:16	NAND_READ_CMD_CODE2 NAND Flash read command confirm code.
HW_OCOTP_ROM4:0x8002C1E0:15:8	NAND_READ_CMD_CODE1 NAND Flash read command setup code.
HW_OCOTP_ROM4:0x8002C1E0:7:4	NAND_COLUMN_ADDRESS_BYTES NAND Flash column address cycles.

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM4:0x8002C1E0:3:0	NAND_ROW_ADDRESS_BYTES NAND Flash row address cycles.

Table 12-8. General ROM Bit in ROM7 OCOTP Bank

eFuse Bank:Address:Bit	eFuse Function
HW_OCOTP_ROM7:0x8002C210:23	Reserved
HW_OCOTP_ROM7:0x8002C210:22	ARM_PLL_DISABLE PLL is enabled by default, the ARM clock will run at 240 MHz. If blown, the PLL will be disabled and ROM will run at 24 MHz during the boot except for USB mode.
HW_OCOTP_ROM7:0x8002C210:21:20	HAB_CONFIG 00 = Reserved 01 = HAB_OPEN, default mode Other = HAB_CLOSED
HW_OCOTP_ROM7:0x8002C210:19:12	Reserved
HW_OCOTP_ROM7:0x8002C210:8	ENABLE_SSP_12MA_DRIVE Blow to force SSP pins to drive 12 mA, default is 4 mA.
HW_OCOTP_ROM7:0x8002C210:7:4	Reserved.
HW_OCOTP_ROM7:0x8002C210:3	I2C_USE_400 KHZ Blow to force the I^{2}C to be programmed by the boot loader to run at 400 KHz. 100 KHz is the default.
HW_OCOTP_ROM7:0x8002C210:2	ENABLE_ARM_ICACHE Blow to enable the ARM 926 ICache during boot.
HW_OCOTP_ROM7:0x8002C210:1	MMU_DISABLE 0 =MMU and D-Cache are enabled in default to speed up HAB functions execution speed. 1 =MMU and D-Cache are disabled during boot if blown.
HW_OCOTP_ROM7:0x8002C210:0	ENABLE_PIN_BOOT_CHECK Blow to enable boot loader to first test the LCD_RS pin to determine if the pin boot mode is enabled. If this bit is blown and LCD_RS is pulled high, then boot mode is determined by the state of LCD_D[5:0] pins. If this bit is not blown, skip testing the LCD_RS pin and go directly to determine the boot mode by reading the state of LCD_D[5:0].

The following bits are used to store super root key hash value, which will be used by the ROM HAB (high assurance boot) library to check the boot image has the correct super root key. The SRK hash value will be read by the ARM CPU, so those registers are shadowed and can also be locked. If HAB is not used, these fuses can be used for other purposes.

Table 12-9. Super Root Key Hash Bits

eFuse Bank:Add:Bit	eFuse Function
HW_OCOTP_SRK0: 0x8002C220:31:0	Super Root Key hash value bits 255-254
HW_OCOTP_SRK1: 0x8002C230:31:0	Super Root Key hash value bits 223-192
HW_OCOTP_SRK2: 0x8002C240:31:0	Super Root Key hash value bits 191-160
HW_OCOTP_SRK3: 0x8002C250:31:0	Super Root Key hash value bits 159-128
HW_OCOTP_SRK4: 0x8002C260:31:0	Super Root Key hash value bits 127-96
HW_OCOTP_SRK5: 0x8002C270:31:0	Super Root Key hash value bits 95-64
HW_OCOTP_SRK6: 0x8002C280:31:0	Super Root Key hash value bits 63-32
HW_OCOTP_SRK7: 0x8002C290:31:0	Super Root Key hash value bits 31-0

12.3.2 Persistent Bits

Persistent bits are used to control certain features in ROM, as shown in Table 12-10. For more information on persistent bits, see RTC Overview.

Table 12-10. Persistent Bits

Persistent Bit	Function
HW_RTC_PERSISTENT1:0x8005C070:3	SD_SPEED_ENABLE—If this bit is set, ROM puts the SD/MMC card in high-speed mode.
HW_RTC_PERSISTENT1:0x8005C070:2	NAND_SDK_BLOCK_REWRITE—The NAND driver sets this bit to indicate to the SDK that the boot image has ECC errors that reached the warning threshold. The SDK regenerates the firmware by copying it from the backup image. The SDK clears this bit.
HW_RTC_PERSISTENT1:0x8005C070:1	ROM_SECONDARY_BOOT—When this bit is set, ROM attempts to boot from the secondary image if the boot driver supports it. This bit is set by the ROM boot driver and cleared by the SDK after repair.
HW_RTC_PERSISTENT1:0x8005C070:0	FORCE_RECOVERY—When this bit is set, the ROM code forces the system to boot in recovery mode, regardless of the selected mode. The ROM clears the bit.

12.4 Memory Map

The illustration below shows the memory map of the boot loader.

ROM boot code resides in the top 128 K address space. The boot code uses the top 16 K of OCRAM for MMU first level page table. The lowest 60 K of OCRAM is used for loading application data. The 42 KB of OCRAM region called "ROM Data" is used for data by ROM code.

If a system uses external memory, then a boot image may be created which first loads a small SDRAM initialization program into OCRAM. The program will set up the SDRAM, and then the rest of the boot image may continue to load, overwriting the initialization program in OCRAM.

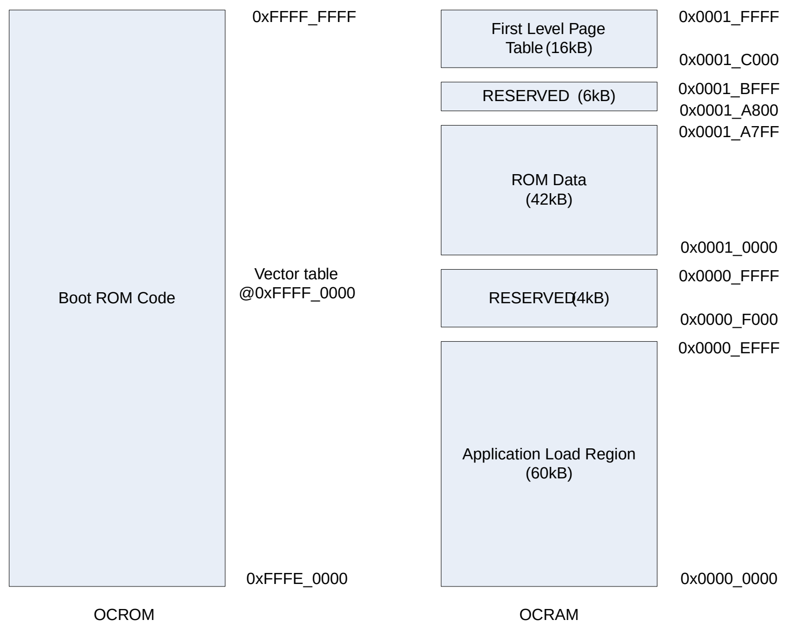


Figure 12-1. i.MX28 ROM Code Memory Map

12.5 General Boot Procedure

During ROM startup, the boot mode is sampled by boot pins, and then the ROM loader will take control. The ROM loader first calls an initialization function for the selected boot driver, which initializes the hardware port and external device corresponding to the boot mode. After that, the loader requests for the stream of boot image data from the driver. The boot image contains a stream of boot commands, such as LOAD, LOAD DCD, SKIP, HAB JUMP and HAB CALL commands. The loader implemented in ROM code will interpret these commands and load the boot image into memory and execute HAB JUMP or HAB CALL command. In the case of HAB JUMP, the ROM will pass control to the boot image and will leave the ROM context.

Boot images can be encrypted and customers have full control over the encryption keys through the CRYPTO_KEY OTP fuse bits. Encryption can be turned on/off by ENABLE_UNENCRYPTED_BOOT fuse (boot unencrypted image if the fuse is blown). Also, the HAB_CONFIG fuse (01'b for HAB_OPEN and 10'b and 11'b for HAB_CLOSED) control the type of HAB security type. If HAB_CONFIG is programmed as HAB_CLOSED, the boot image will not be executed unless the HAB authenticates the boot image. If HAB_CONFIG is programmed as HAB_Open , the boot image will be executed even if the HAB authentication fails. The Open configuration is used for non-secure products or for development purposes on secure products. Customers have the control over the HAB super root key hash through HW_OTP_SRK fuses, which will be used to compare the DCP SHA-256 generated super root key (public key) hash from the boot image.

To accelerate the boot process, the MMU/d-cache is enabled during the time consuming HAB authentication process (RSA). This feature can be turned off by blowing the MMU_DISABLE fuse. For the same purpose of speeding boot, the ARM clock has been boosted from 24 MHz to 240 MHz. The feature could be disabled by blowing the ARM_PLL_DISABLE fuse. The i-Cache can be turned on by blowing the ENABLE_ARM_ICACHE fuse.

12.6 Program Image

This section describes the data structures that are required to be included in a user's Program Image. A Program Image consists of:

- Image Vector Table - a list of pointers that the ROM examines to determine where other components of the program image are located.
- Boot Data: This field is not used on i.MX28
- Device Configuration Data: IC configuration data
- User Code and Data

Additional data is required if the High Assurance Boot (HAB) feature of the ROM is used. In this case, additional HAB Command Sequence File (CSF) data is also required. See High Assurance Boot (HAB) for additional details.

12.6.1 Image Vector Table (IVT)

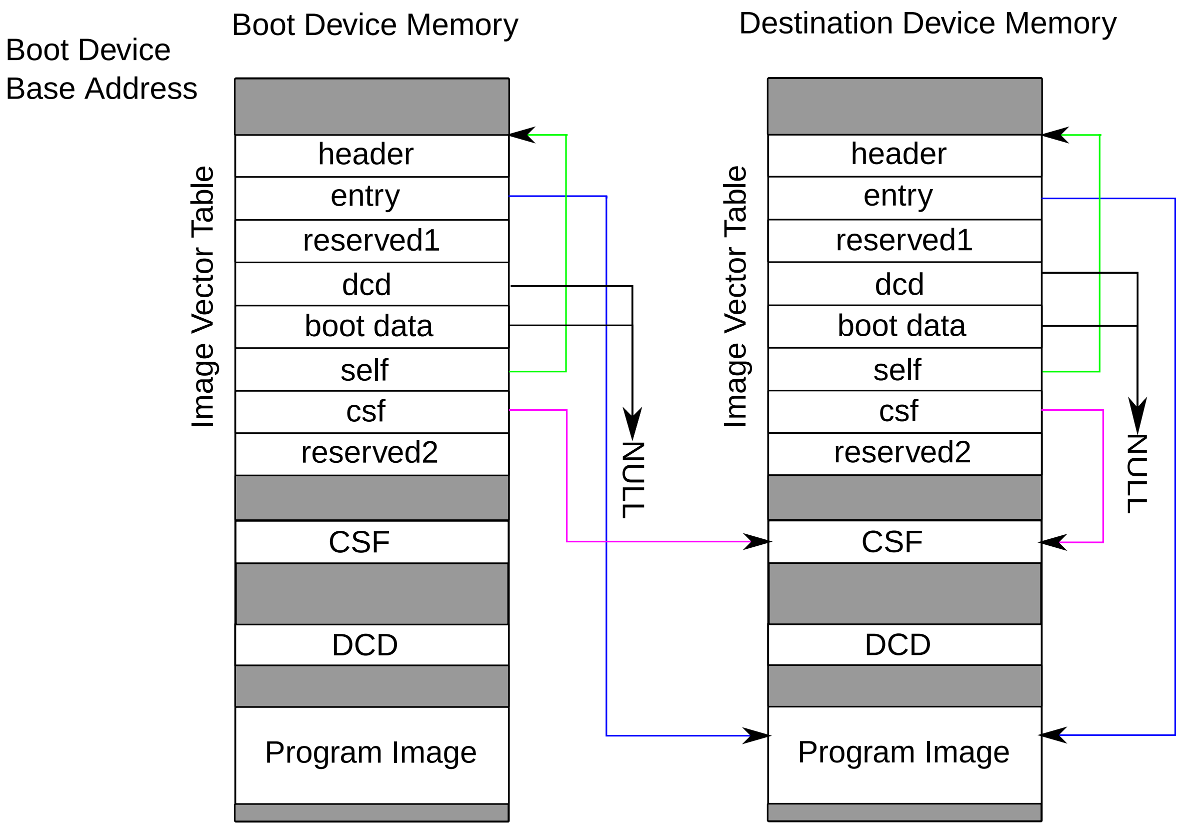
The IVT is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot. The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process. The ROM locates the IVT at a fixed address that is determined based on the boot device connected to i.MX28. The IVT offset from the base address and initial load region size for each boot device type is defined in the following table. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

NOTE

On i.MX28 the IVT must **not** be placed at offset 0x00000000. The ROM will interpret this as a NULL pointer and reject the location of the IVT as invalid.

NOTE

The i.MX28 ROM does not make use of the DCD or boot data fields of the IVT. These fields must be set to NULL.



12.6.1.1 Image Vector Table Structure

The IVT has the following format where each entry is a 32 bit word:

Table 12-11. IVT Format

header
entry
reserved1
dcd
boot data
self
csf
reserved2

- header: The IVT header has the following format:

  Table 12-12. IVT Header Format

Tag	Length	Version

  where:
  - Tag: A single byte field set to 0xD1
  - Length: a two byte field in big endian format containing the overall length of the IVT, in bytes, including the header. (the length is fixed and must have a value of 32 bytes)
  - Version: A single byte field set to 0x40

- entry: Absolute address of the first instruction to execute from the image.
- reserved1: Reserved and should be zero.
- dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on DCD.
- boot data: Absolute address of the Boot Data. For i.MX28 this field should be zero.
- self: Absolute address of the IVT. Used internally by the ROM.
- csf: Absolute address of Command Sequence File (CSF) used by the HAB library. See High Assurance Boot (HAB) for details on secure boot using HAB. This field must be set to NULL when not performing a secure boot when HAB is enabled.
- reserved2: Reserved and should be zero.

12.6.2 Device Configuration Data (DCD)

Upon reset the i.MX28 uses the default register values for all peripherals in the system. The i.MX50 ROM changes some of these defaults such as clock frequencies for boot purposes. However, these settings typically are not ideal for achieving optimal system performance and there are even some peripherals that must configured before they can be used. The DCD is configuration information contained in a Program Image, external to the ROM, that the ROM interprets to configure various peripherals on i.MX28. Some components such as SDRAM require some sequence of register programming as part of configuration before it is ready to be used. The DCD feature can be used to program the EIM registers and DATABAHN SDRAM control registers to the optimal settings.

NOTE

The i.MX28 ROM does not use the Image Vector Table (IVT) to locate the DCD. Instead, the i.MX28 ROM uses the Load DCD command. See ROM Commands for details on the Load DCD ROM command.

The DCD table shown in below is a big endian byte array of the allowable DCD commands. The maximum size of the DCD limited by the ROM to 1768 bytes.

Table 12-13. DCD Data Format

Header
[CMD]
[CMD]
...

The DCD header is 4 bytes with the following format:

Table 12-14. DCD Header

Tag	Length	Version

where:

- Tag: A single byte field set to 0xD2
- Length: a two byte field in big endian format containing the overall length of the DCD, in bytes, including the header.
- Version: A single byte field set to 0x40

Below is an example DCD for i.MX28. Note that this example is not a complete DCD and is shown here to illustrate how a DCD is structured.

```
// Multi Write Data Command
//
// Creates Write Data Command header which performs writes to multiple taget
// addresses
//
// inputs: flags - DCD command flags
//         bytes - size of write by command (1, 2, or 4)
//         number - number of writes performed by the DCD write data command
#define MULTI_WRT_DAT(flags, bytes, number)                         \
    HDR(HAB_CMD_WRT_DAT, (number * 2 + 1) * 4, WRT_DAT_PAR((flags), (bytes)))




// DCD Table definition
uint8_t input_dcd[] = {
      /* DCD header */
      DCD_HDR(HDR_BYTES + 26 * WRT_DAT_BYTES + (63 * 2 + 1) * 4, HAB_VER(4,0)),

//void POWER_Init(void) just pick one register since no impact on RTL
//    HW_POWER_LOOPCTRL.B.EN_RCSCALE = 3;
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800440b4, 0x00003000),


// PLL already turn on by ROM
// HW_CLKCTRL_FRAC0_CLR(BM_CLKCTRL_FRAC0_CLKGATEEMI);
// Turn on fractional clock control 0 EMI clkgate,
// setmem /32 0x800401b8=0x00008000
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800401b8, 0x00008000),


// Set up the EMI clock
//        case EMI_CLK_150MHz:
//            use_xtal_src = 0;
//            new_pll_frac_div = 29;
//            new_pll_int_div = 2;
// Write the PLL fractional divider W_CLKCTRL_FRAC0_WR(frac_val);
// Clear the EMI frac first
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800401b8, 0x00003F00),
// write new_pll_frac_div
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800401b4, 29 << 8),


                          .
                          .
                          .

// init_dram_regs(); // Write the Databahn SDRAM setup register values
// use mobile_ddr_mt46h32m16lf_5_150MHz_for_dcd.c -- for 150MHz mDDR, 63 entries
MULTI_WRT_DAT(0, HAB_DATA_WIDTH_WORD, 63),
//    DRAM_REG[0] =    0x00000000;
EXPAND_UINT32(0x800e0000 + 0 * 4), EXPAND_UINT32(0x00000000),
                          .
                          .
                          .

//    DRAM_REG[177] =   0x01030101;
//000_00001 tccd(RW) 0000_0011 trp_ab(RW)
//0000_0001 cksrx(RW) 0000_0001 cksre(RW)
EXPAND_UINT32(0x800e0000 + 177 * 4), EXPAND_UINT32(0x01030101),
//    DRAM_REG[178] =   0x01001901;
//0_0100001 axi5_bdw(RW) 0_0000000 axi4_current_bdw(RD)
//0_0100001 axi4_bdw(RW) 000_00001 tckesr(RW)
EXPAND_UINT32(0x800e0000 + 178 * 4), EXPAND_UINT32(0x01001901),
//    DRAM_REG[181] =   0x00320032;
//0_000000000110010 mr0_data_1(RW) 0_000000000110010 mr0_data_0(RW)
```

```
EXPAND_UINT32(0x800e0000 + 181 * 4), EXPAND_UINT32(0x00320032),
//    DRAM_REG[183] =   0x00000000;
//0_0000000000000000 mr1_data_1(RW) 0_0000000000000000 mr1_data_0(RW)
EXPAND_UINT32(0x800e0000 + 183 * 4), EXPAND_UINT32(0x00000000),
//    DRAM_REG[189] =   0xffffffff;
EXPAND_UINT32(0x800e0000 + 189 * 4), EXPAND_UINT32(0xffffffff),

//     HW_DRAM_CTL17.B.SREFRESH = 0;
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800e0044, 0x00000000),
//     HW_DRAM_CTL16_SET(0x00000001);  //set "start"
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x800e0040, 0x00000001),

// temp = HW_DRAM_CTL58_RD();  //Wait for EMI initialization completed
// while ( (temp & 0x00100000) != 0x00100000 ){
// temp = HW_DRAM_CTL58_RD();
CHK_DAT_FOREVER(HAB_CMD_CHK_DAT_SET, HAB_DATA_WIDTH_WORD, 0x800e00e8, 0x00100000),

// Now try to write something to ddr addresses 0x40200000 and 0x40200400
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x40200000, 0xAAAAAAAA),
WRT_DAT(0, HAB_DATA_WIDTH_WORD, 0x40200400, 0x12345678),

// Verify above two writes to ddr addresses
CHK_DAT_FOREVER(HAB_CMD_CHK_DAT_SET, HAB_DATA_WIDTH_WORD, 0x40200000, 0xAAAAAAAA),
CHK_DAT_FOREVER(HAB_CMD_CHK_DAT_SET, HAB_DATA_WIDTH_WORD, 0x40200400, 0x12345678),
};
```

12.6.2.1 Write Data Command

The Write Data Command is used to write a list of given 1, 2 or 4-byte values or bitmasks to a corresponding list of target addresses. The format of Write Data Command, again a big endian byte array, is shown in the Write Data Command table below.

Table 12-15. Write Data Command Format

Tag	Length	Parameter
Address		
Value/Mask		
[Address]		
[Value/Mask]		
...		
[Address]		
[Value/Mask]		

where:

- Tag: A single byte field set to 0xCC
- Length:

  A two byte field in big endian format containing the length of the Write Data Command, in bytes, including the header.

- Address: Target address to which the data should be written
- Value/Mask: Data value or bit mask to be written to the preceding address

The Parameter field is a single byte divided into bit fields as follows:

Table 12-16. Write Data Command Parameter Field

7	6	5	4	3	2	1	0
flags					bytes		

where:

- bytes: width of target location(s) in bytes. Either 1, 2 or 4.
- flags: control flags for command behavior.

One or more target address and value/bitmask pairs can be specified. The same bytes and flags parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 12-17. Interpretation of Write Data Command Flags

"Mask:	"Set"	Action	Interpretation
0	0	*address = val msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

Notes:

- If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.
- If any of the values is larger or any of the bitmasks is wider than permitted by the data width indicated in the parameter field, none of the values are written.
- If any of the target addresses do not lie within an allowed region, none of the values are written. The list of allowable modules and target addresses for i.MX28 are given below.

Table 12-18. Valid DCD Address Ranges for i.MX28

Address Range	Start Address	Last Address
DRAM Controller registers	0x800E0000	0x800EFFFF
BCH ECC Accelerator	0x8000A000	0x8000BFFF
GMPI registers	0x8000C000	0x8000DFFF
Synchronous Serial Port 0 registers	0x80010000	0x80011FFF
Synchronous Serial Port 1 registers	0x80012000	0x80013FFF

Address Range	Start Address	Last Address
Synchronous Serial Port 2 registers	0x80014000	0x80015FFF
Synchronous Serial Port 3 registers	0x80016000	0x80017FFF
Pin Control Block registers	0x80018000	0x80019FFF
Digital Control registers	0x8001C000	0x8001DFFF
GPIO Control registers	0x8003C500	0x8003C5FF
Clock Control registers	0x80040000	0x80041FFF
Power Control registers	0x80044000	0x80045FFF
Real Time Clock registers	0x80056000	0x80057FFF
I2C0 registers	0x80058000	0x80059FFF
I2C1 registers	0x8005A000	0x8005BFFF
UART0 registers	0x8006A000	0x80000000
UART1 registers	0x8006C000	0x8006DFFF
UART2 registers	0x8006E000	0x8006FFFF
UART3 registers	0x80070000	0x80071FFF
UART4 registers	0x80072000	0x80073FFF
UARTDBG registers	0x80074000	0x80075FFF
External memory	0x40000000	0x5FFFFFFF
OCRAM free space	0x00000000	0x0000E3FF

12.6.2.2 Check Data Command

The Check Data Command is used to test for a given 1, 2 or 4-byte bit masks from a source address. The Check Data Command is a big endian byte array with format shown below.

Table 12-19. Check Data Command Format

Tag	Length	Parameter
Address		
[Count]		

where:

- Tag: A single byte field set to 0xCF
- Length:

  A two byte field in big endian format containing the length of the Check Data Command, in bytes, including the header.

- Address: Source Address to test

- Mask: Bit mask to test
- Count: optional poll count. If count is not specified this command will poll indefinitely until the exit condition is met. If count = 0, this command behaves as for NOP.

The Parameter field is a single byte divided into bit fields as follows:

Table 12-20. Write Data Command Parameter Field

7	6	5	4	3	2	1	0
flags					bytes		

where:

- bytes: width of target location in bytes. Either 1, 2 or 4.
- flags: control flags for command behavior.
    - Data Mask = bit 3: if set, only specific bits may be overwritten at target address (otherwise all bits may be overwritten)
    - Data Set = bit 4: if set, bits at the target address overwritten with this flag (otherwise it is ignored)

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 12-21. Interpretation of Check Data Command Flags

"Mask:	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask) != mask	Any bit clear
1	1	(*address & mask) != 0	Any bit set

Notes:

- If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.
- If the bit mask is wider than permitted by the data width indicated in the parameter field, the value is not read.

12.6.2.3 NOP Command

This command has no effect. The format of NOP Command is a little endian four byte array as shown below:

Table 12-22. NOP Command Format

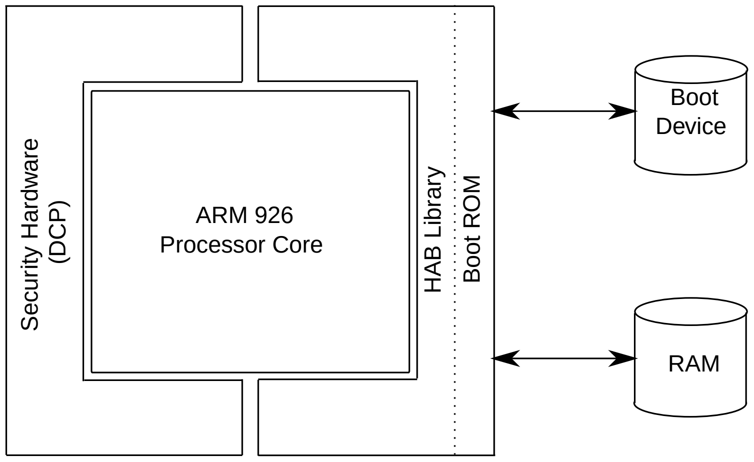
Tag	Length	Undefined

where:

- Tag: A single byte field set to 0xC0
- Length: A two byte field containing the length of the NOP Command in bytes. Fixed to a value of 4.
- Undefined: This byte is ignored and can be set to any value.

12.7 High Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying areas of code or data in programmable memory to make it behave in an incorrect manner. The HAB also prevents attempts to gain access to features which should not be available. The integration of the HAB feature with the ROM code ensures that i.MX28 does not enter an operational state if the existing hardware security modules have detected a condition that may be a security compromise or areas of memory deemed to be important have been modified. The HAB uses RSA digital signatures to enforce these policies. Note that the HAB feature is independent of the encrypted boot feature. The HAB checks the authenticity of a Progam Image using a public key infrastructure whereas the encrypted boot feature provides confidentiality.



The figure above illustrates the components used during a secure boot using HAB. The HAB makes use of the DCP2 hardware module to accelerate SHA-256 message digest operations performed during signature verifications. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The core RSA signature verification operations are performed by a software implementation contained in the HAB library. The main features supported by HAB are:

- X.509 pulic key certificate support
- CMS signature format support

For further details on making use of the High Assurance Boot feature of i.MX28, please contact your local Freescale representative.

12.7.1 ROM Vector Table Addresses

For devices that are performing a secure boot using HAB, the HAB library may be called from additional boot stages that execute after ROM code. The RVT table contains the pointers to the HAB API functions such that these boot stage can authenticate the stage that follows. The RVT is fixed in ROM and is located at 0xFFFF8508.

For additional information on using HAB on i.MX28, including the HAB API, please contact your local Freescale representative.

12.8 Constructing Boot Image (SB Files) to Be Loaded by ROM

Boot images are created by the Freescale supplied *elftosb* application, which handles both encryption and HAB signature if required. Freescale provides a sample code signing tool. It can be used for reference purposes in integrating with third-party tools or a proprietary signing infrastructure.

Preparing a bootable image for all boot modes includes the following high-level steps:

1. Prepare the unsigned Image ELF file for the firmware that is to be booted by the ROM.
2. Sign the image with the code signing tool, which generates one HAB.ELF file (Freescale provides example tools, and third party code signing tools may also be available. Contact your local Freescale representative for further details). This file contains the CSF (HAB command sequence file) and certificates.
3. Use the *elftosb* tool to combine the unsigned Image ELF and HAB.ELF together and convert them to a boot stream (SB file), which the ROM loader will understand. The *elftosb* tool also handles encryption if an encryption key is provided.

The following figures shows the process of creating a boot loader image from ELF files by *elftosb*. A key set must be input to the *elftosb* program to properly encrypt and authenticate the image.

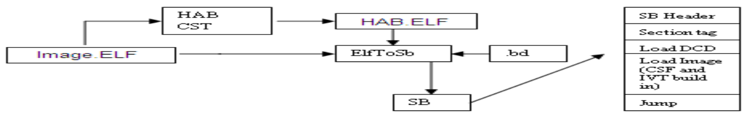


Figure 12-2. i.MX28 Boot Image Generation

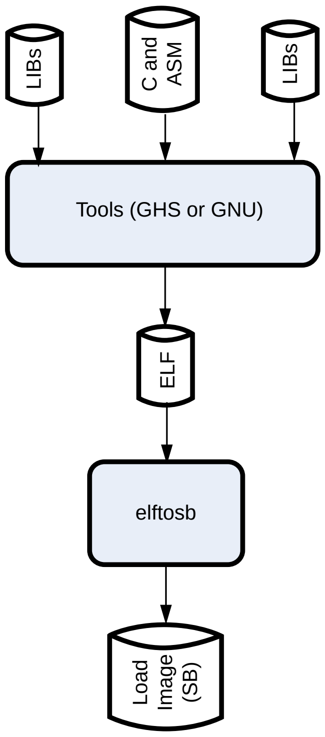


Figure 12-3. i.MX28 Boot Image Build Process

12.8.1 ROM Commands

The i.MX28 ROM boots from an image residing on an external boot device using a number of built in commands. These commands are used by the ROM to establish the boot image including the Image Vector Table (IVT). The ElftoSB tool provided by Freescale converts a typical image file in ELF format to the Freescale SB format consisting of the following commands:

LOAD DCD Command

LOAD DCD is an extension from LOAD command by adding an additional flag. When the DCD flag is set, ROM runs DCD to configure the external memory. The ROM command is 16 bytes in length. The following table shows the field description. Bold indicates the changes.

Table 12-23. LOAD DCD Command Field Descriptions

Field	Description
m_checksum	Simple checksum of other fields of boot_command_t.
m_tag	0x02 or ROM_LOAD_CMD.
m_flags	**Normal load 0x0000, DCD load 0x0001**.
m_address	Memory address to which data is stored.
m_count	Number of bytes to load. This is also the number of valid bytes in the data cipher blocks following this command.
m_data	CRC-32 over the data to be loaded.

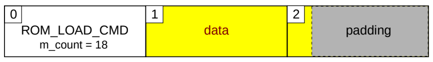


Figure 12-4. LOAD DCD Command

HAB Jump Command

The HAB Jump command is overload from Jump command by adding an additional flag. When the HAB flag is set, ROM calls the HAB4 Authenticate image function with IVT as argument. The authenticate image function returns the image entry, and then ROM performs a real jump to that address. The following table shows the HAB Jump command fields. Bold indicates the changes. For i.MX28, JUMP without HAB is no longer valid hence HAB flag should always be set.

Table 12-24. HAB Jump Command Field Descriptions

Field	Description
m_checksum	Simple checksum of other fields of boot_command_t.
m_tag	0x04 or ROM_JUMP_CMD.
m_flags	**HAB jump 0x0001. HAB flag should always be set for i.MX28**.
m_address	Address of the IVT pointer for HAB jump.
m_count	0
m_data	Argument to pass to the entry point in R0.

HAB Call Command

The HAB Call command is overload from Call command by adding an additional flag. When the HAB flag is set, ROM calls the HAB4 authenticate image function with IVT as an argument. The authenticate image function returns the image entry, and then ROM performs a real call to that address. Bold indicates the changes. For i.MX28,CALL without HAB is no longer valid hence HAB flag should always be set.

Table 12-25. HAB Call Command Field Descriptions

Field	Description
m_checksum	Simple checksum of other fields of boot_command_t.
m_tag	0x05 or ROM_CALL_CMD.
m_flags	**HAB call 0x0001. HAB flag should always be set for i.MX28**.
m_address	Address of the IVT pointer for HAB call.
m_count	0
m_data	Argument to pass to the function in R0.

12.9 I^{2}C Boot Mode

EEPROMs must have the slave address 0xA0 (that is, 1010000R, where R indicates a read if 1 and a write if 0). Also, the EEPROM must have exactly a two-byte subaddress.

Boot images must start at address 0x0 of the EEPROM and cannot exceed 64 Kbytes in size. The I^{2}C port is set to run at 100 KHz by default. The clock is increased to 400 KHz if I2C_USE_400 KHz is blown.

12.10 SPI Boot Mode

SPI memories are either EEPROMs or NORs.

By default, the SPI serial clock is set to 1 MHz for EEPROMs and 12 MHz for NORs. The SSP_SCK_INDEX OTP 4 bits are used to change the SPI serial clock from defaults. These bits serve as the index for the SSP HAL clock rate array (see SSP for details on the clock rate array).

The defaults may also be changed by using the ConfigBlock.Clocks field. If ConfigBlock.Clocks.SspClockConfig is non-zero, then that struct will be used to change the SPI SCK rate and will override the SSP_SCK_INDEX OTP setting.

This driver supports only 2-byte addresses for EEPROMs and 3-byte addresses for NORs.

SSP2 and SSP3 are used for SPI boot mode.

12.10.1 SSP Pin Configuration

SSP boot mode supports the standard package and the small package of i.MX28.

Table 12-26. SSP Ports Pin-Mux Configuration

SSP Pins	SSP0	SSP1	SSP2	SSP3 (standard package)	SSP3 (small package)
SCK	SSP0_CLK	GPMI_WRN	SSP2_SCK	SSP3_SCK	GPMI_RDN
DETECT	SSP0_DETECT	GPMI_RDY0	—	—	—
CMD	SSP0_CMD	GPMI_RDY1	SSP2_MOSI	SSP3_MOSI	GPMI_RESETN
DATA7	SSP0_DATA7	GPMI_D07	—	—	—
DATA6	SSP0_DATA6	GPMI_D06	—	—	—
DATA5	SSP0_DATA5	GPMI_D05	—	—	—
DATA4	SSP0_DATA4	GPMI_D04	—	—	—
DATA3	SSP0_DATA3	GPMI_D03	SSP2_S30	SSP3_S30	GPMI_CE1N
DATA2	SSP0_DATA2	GPMI_D02	—	—	—
DATA1	SSP0_DATA1	GPMI_D01	—	—	—
DATA0	SSP0_DATA0	GPMI_D00	SSP2_MISO	SSP3_MISO	GPMI_CE0N

12.10.2 Media Format

The media is arbitrarily partitioned into 128-byte sectors. An optional configuration block may reside on the media at byte address 0. This block has the following format:

```
//! \brief Spi media configuration block structs
typedef struct _spi_ConfigBlockFlags
{
    uint32_t  DisableFastRead:1; // Ignored for Spis
                                 // 0 - Do not disable fast reads
                                 // 1 - Disable fast reads
} spi_ConfigBlockFlags_t;
typedef struct _spi_ConfigBlockClocks
{
    uint32_t           SizeOfSspClockConfig; // sizeof(ssp_ClockConfig_t)
    ssp_ClockConfig_t  SspClockConfig;       // SSP clock configuration structure. A null
                                             //   structure indicates no clock change.
} spi_ConfigBlockClocks_t;
typedef struct _spi_ConfigBlock // Little Endian
{
    uint32_t  Signature;                    // 0x4D454D53, or "SMEM"
    uint32_t  BootStartAddr;                // Start address of boot image. Must be >=
                                            //   sizeof(spi_ConfigBlock_t)
    uint32_t  SectorSize;                   // Sector size in bytes. Overrides ROM default
                                            //   of 128-bytes. Max is 1024-bytes.  0 is
                                            //   default 128-bytes.
    spi_ConfigBlockFlags_t Flags;           // Various flags. See spi_ConfigBlockFlags
    spi_ConfigBlockClocks_t Clocks;         // SCK clock update structure.
} spi_ConfigBlock_t;
```

If the block is present, then the boot image is found at the address specified by BootStartAddr. If the block is not present, then it assumes that the boot image resides on the media starting at byte address 0.

12.10.3 SSP

The SSP2 and SSP3 ports are used for the SPI boot mode.

The following table is used to look up a requested speed. If the speed is not an exact match, the boot ROM picks the next lowest value. Speed values can range from 1 to 51.4 MHz. A speed value of 0 is not allowed.

```
//  Lookup Table entry
typedef struct _ssp_clockConfig
{
    int    clkSel   :1; //!< Clock Select (0=io_ref 1=xtal_ref)
    int    io_frac   :6; //!< IO FRAC 18-35 (io_frac+16)
    int    ssp_frac   :9; //!< SSP FRAC (1=default)
    int    ssp_div   :8; //!< Divider: Must be an even value 2-254
    int    ssp_rate   :8; //!< Serial Clock Rate
}
ssp_clockConfig_t;
```

The table is loaded with the clock rates listed in Table 12-27.

Table 12-27. SCK Clock Standard Values Lookup Table

SCK	N/A	.24	1	2	4	6	8	10	12	16	20	30	40	48	51.4	X
INDEX	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CLK_SEL	X	1	1	1	1	1	0	0	1	0	0	0	0	0	0	
IOFRAC	X	X	X	X	X	X	18	18	X	18	18	18	18	18	21	X
SSP_FRAC	X	X	X	X	X	X	6	6	X	5	6	8	6	5	4	X
SSP_CLK		24	24	24	24	24	80	80	24	96	80	60	80	96	102.8	X
SSP_DIV	X	100	24	12	6	4	10	8	2	6	4	2	2	2	2	X
SSP_RATE	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X

12.11 SD/MMC Boot Mode

SD/MMC boot mode supports booting from SD/MMC cards adhering to the following specifications:

- iNand Product Manual, Version 2.1
- SD Specifications, Part 1, Physical Layer Specification, Version 1.10
- SD Specifications, Part 1, Physical Layer Specifications, Version 2.0 Draft
- SD Specifications, Part 1, eSD (Embedded SD Specifications), Version 2.10 Draft Rev
- MultiMediaCard System Specification, Version 4.1
- MultiMediaCard System Specification, Version 4.2
- MultiMediaCard System Specification, Version 4.3
- MultiMediaCard System Specification, Version 4.4

Note, however, that this mode does not support dynamic insertion removal, so the systems will typically not include removable cards.

The following modes are supported:

- SD/MMC on SSP0
- SD/MMC on SSP1

The SD_POWER_GATE_GPIO eFuse bits indicate which GPIO pin to use for controlling an external power gate for the connected device.

SD_POWER_GATE_GPIO	Gate GPIO
00b	PWM3
01b	PWM4
10b	LCD DOTCK
11b	NONE

If a gate GPIO is used, then the driver will use the SD_POWER_UP_DELAY eFuse to determine the amount of time, in 10-ms increments, to wait until starting the 1-ms initialization sequence. This eFuse field is 6-bits wide, providing from 10–600 ms of delay. If the field is 000000b, then the delay is a default 20 ms. If no gate GPIO is specified in SD_POWER_GATE_GPIO, then the delay is skipped.

The SSP ports on the i.MX28 top out at 51.4 MHz with 20–40 pF loading. By default, the serial clock is set to 12 MHz. If the SD_SPEED_ENABLE persistent bit is set, then the driver will use a maximum speed based on the results of device identification and limited by choices available in the SSP clock index.

For eMMC fast boot mode, serial clock is just decided by SSP_SCK_INDEX OTP bits. If SSP_SCK_INDEX is not burned, that is invalid clock index, then ROM chooses the default clock (12 MHz).

This mode supports the 1-bit, 4-bit, and 8-bit data MMC/SD buses. The SD_BUS_WIDTH efuse bits selects how many bus pins are physically available for the SSP port. Bus width will be limited based on these bits, as well as the bus width capabilities indicated by the connected device.

SD_BUS_WIDTH	Width
00b	4-bit
01b	1-bit
10b	8-bit
11b	Reserved

The eFuse ROM0:23:22 defines one of four possible media formats, as shown in the following table.

Table 12-28. Media Formats

SD_MMC_MODE	Media Format
00b	MBR_MEDIA_FORMAT
01b	RESERVED
10b	eMMC_FASTBOOT_MEDIA_FORMAT
11b	eSD_FASTBOOT_MEDIA_FORMAT

12.11.1 Boot Control Block (BCB) Data Structure

The design of BCB is to allow multiple copies of firmware to be stored on media each identified by its unique tag. The tags can be defined either by the user or the firmware download application. The ROM is only interested in user-defined primary and secondary boot tags. The ROM loads primary firmware, if ROM_REDUNDANT_BOOT persistent bit is not set; otherwise it loads a secondary image, providing support for a redundant boot. The config block has the following format:

```
typedef struct _DriveInfo_t
{
    uint32_t    u32ChipNum;            //!< Chip Select, ROM does not use it
    uint32_t    u32DriveType;          //!< Always system drive, ROM does not use it
    uint32_t    u32Tag;                //!< Drive Tag
    uint32_t    u32FirstSectorNumber;  //!< Start sector/block address of firmware.
    uint32_t    u32SectorCount;        //!< Not used by ROM
} DriveInfo_t;

typedef struct _ConfigBlock_t
{
    uint32_t    u32Signature;          //!< Signature 0x00112233
    uint32_t    u32PrimaryBootTag;     //!< Primary boot drive identified by this tag
    uint32_t    u32SecondaryBootTag;   //!< Secondary boot drive identified by this tag
    uint32_t    u32NumCopies;          //!< Num elements in aFWSizeLoc array
    DriveInfo_t aDriveInfo[];          //!< Let array aDriveInfo be last in this data
                                      //!< structure to be able to add more drives in future

                                       //!< without changing ROM code
} ConfigBlock_t;
```

The driver first verifies the signature and version, then searches all NumRegions for the appropriate tag. The following table shows the expected values for these parameters.

Table 12-29. Media Config Block Parameters

Field	Value
Signature	0x00112233
u32PrimaryBootTag	User-defined primary boot firmware tag
u32SecondaryBootTag	User- defined secondary boot tag
u32NumCopies	Number of firmware copies present in array aDriveInfo
aDriveInfo	Each element in array describes the tag and start address for the image

12.11.2 Master Boot Record (MBR) Media Format

If the eFuse media format mode is MBR_MEDIA_FORMAT, then ROM expects a valid master boot record (MBR) to be present on the first block of media. The MBR is identified by its signature located at offset 0x1FE of the first sector. The partition table is stored at address 0x1BE. The Freescale firmware partition is identified by MBR_SIGMATEL_ID at an offset 0x04 from partition table. The firmware partition's start block address is located at offset 0x08 of firmware partition entry of partition table.

Field	Value
MBR Signature	0x55AA
MBR_SIGMATEL_ID	'S'

The first block of firmware partition contains BCB, allowing multiple copies of firmware to reside inside firmware partition and to support redundant boot feature of ROM. Refer to Boot Control Block (BCB) Data Structure for a detailed view of BCB data structure and its use. All firmware copies specified in BCB should be located inside the firmware partition.

12.11.3 eMMC Fast Boot Media Format

This section describes the behavior of ROM when the eFuse state of SD_MMC_MODE is 0x10 (eMMC_FASTBOOT_MEDIA_FORMAT). The eMMC4.3 and eMMC4.4 fast boot modes are supported. By default, primary fast boot mode, pull down CMD line after eMMC card power up, is enabled. The ALT_FAST_BOOT OTP bit enables the alternative fast boot mode, which sends CMD0 with an argument 0xFFFF_FFFA after power up.

The boot image for eMMC fast boot does not contain a MBR block or a configuration block. It is the user's responsibility to program the EXT_CSD register when loading the boot image into an eMMC card.

The bytes in EXT_CSD register have to be read or programmed to include the following:

- [228] BOOT_INFO
- [226] BOOT_SIZE_MULT
- [179] BOOT_CONFIG(eMMC4.3) PARTITION_CONFIG(eMMC4.4)
- [177] BOOT_BUS_WIDTH

The ALT_FAST_BOOT, FAST_BOOT_ACK, EMMC_USE_DDR and SD_BUS_WIDTH OTP bits must be consistent with the setting in EXT_CSD register.

12.11.4 eSD Fast Boot Media Format

The ROM supports eSD by following the same flow as SD 2.0, except that ROM selects the physical partition 1(boot code area) by sending CMD43. The CMD43 is reserved command in SD2.0. Therefore, for eSD boot mode, boot image has to be located in partition 1.

If eSD fast boot mode is enabled, the FAST_BOOT bit in the argument of ACMD41 is set. The eSD device partition 1 should be defined as a fast boot physical partition. Otherwise, the device still initializes the entire media and not a partial initialization.

The ROM expects the BCB data structure to be present on the first block of boot partition. Like the BCB and MBR media formats, the ROM uses the persistent bit ROM_REDUNDANT_BOOT to decide whether to load primary or secondary boot firmware.

12.11.5 Device Identification

SD/MMC boot mode uses the identification processes specified as follows:

- SD Specifications, Part 1, Physical Layer Specifications, Version 2.0 Draft.
- SD Specifications, Part 1, eSD (Embedded SD) Specification Version 2.10 Draft Rev. 0.91.
- MultiMediaCard System Specification Version 4.2, Version 4.3, Version 4.4.

12.12 NAND Boot Mode

NAND boot mode is used to boot from both raw NAND devices and block abstracted ONFi BA NAND devices, with error correction capability in the controller.

12.12.1 Raw NAND Device Boot

The ROM relies on the BCH hardware engine for handling error corrections when reading data from a raw NAND device.

The boot ROM expects the NAND Flash to be partitioned into the following areas:

- Search Area for Firmware Configuration Block (FCB)
- Search Area for Discovered Bad Block Table (DBBT)
- Firmware blocks with primary and secondary boot firmware

The rest of NAND Flash is available for non-boot purposes. FCB and DBBT together are referred to as boot control blocks, or BCB.

12.12.1.1 Search Area

The search area is defined by search count times search stride.

12.12.1.2 Search Count

Search count is defined as the number of copies of BCB data structures present in a given search area. It is 2^efNANDBootSearchCount (a value from OTP/eFuses). The default search count when OTP is not programmed is 4. This is the most common case. The minimum search count is 2 and the maximum search count is 32768.

12.12.1.3 Search Stride

Search stride is defined as the distance in pages between two BCB data structures in a given search area. It is 64 pages times efNANDBootSearchStride (a value from OTP/eFuses). When efNANDBootSearchStride is not blown, the boot ROM defaults it to 1. Therefore, the minimum search stride is always 64 pages apart (the most common case), and the maximum search stride is (64 * 15 = 960) pages apart.

12.12.1.4 Boot Control Blocks (BCB)

There are two BCB data structures: FCB and DBBT. As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, the page address of Discovered Bad Block Table (DBBT) Search Area and the start page address of the primary and secondary firmware.

In i.MX28, there are no separate boot modes for each type of ECC level. The hardware ECC level to use is embedded inside FCB block. The FCB data structure is itself protected using software ECC (SEC-DED Hamming Codes). Driver reads raw 2112 bytes of first sector and runs through software ECC engine that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments the page number to Search Stride number of pages to read for the next BCB until 2^n efNANDBootSearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into the recovery mode.

The FCB contains the page address of DBBT Search Area, and the page address of primary and secondary boot images. DBBT is searched in DBBT Search Area just like how FCB is searched. A flowchart for this process is shown in Figure 12-5. After the FCB is read, the DBBT is loaded, and the primary or secondary boot image is loaded using the starting page address from FCB.

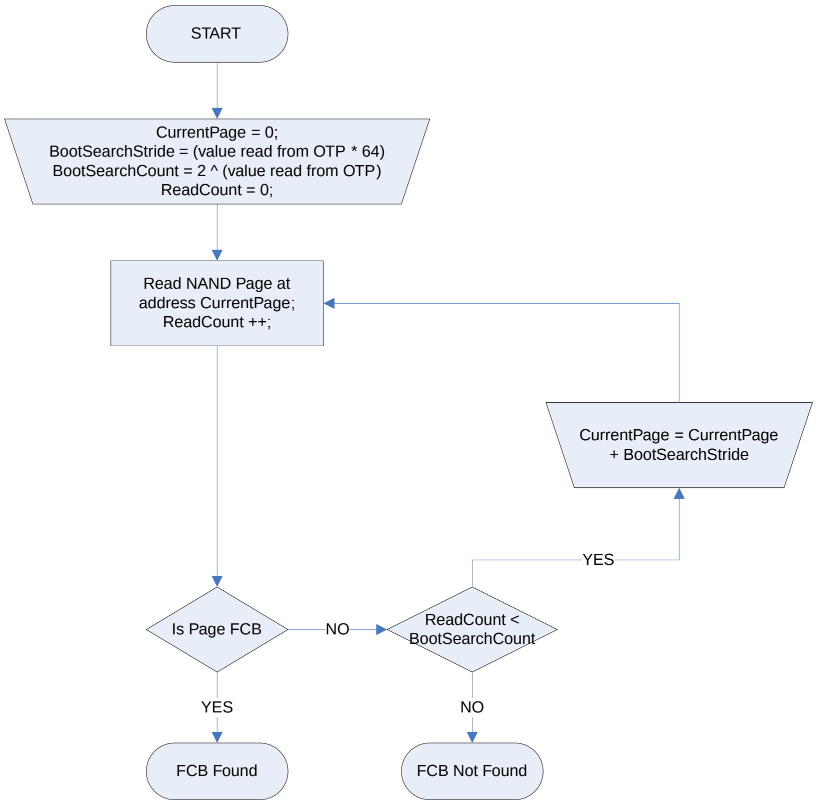

START
CurrentPage = 0;
BootSearchStride = (value read from OTP * 64)
BootSearchCount = 2 ^ (value read from OTP)
ReadCount = 0;
Read NAND Page at address CurrentPage; ReadCount ++;
CurrentPage = CurrentPage + BootSearchStride
YES
Is Page FCB
NO
ReadCount < BootSearchCount
YES
NO
FCB Found
FCB Not Found

Figure 12-5. Firmware Control Blocks Flowchart

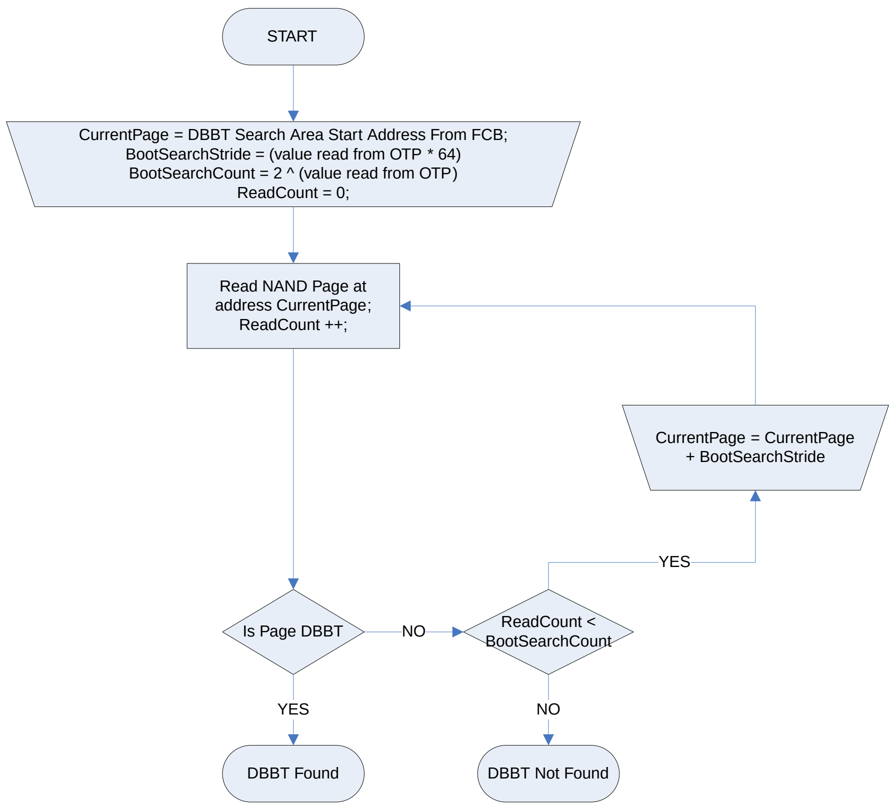

START
CurrentPage = DBBT Search Area Start Address From FCB;
BootSearchStride = (value read from OTP * 64)
BootSearchCount = 2 ^ (value read from OTP)
ReadCount = 0;
Read NAND Page at address CurrentPage; ReadCount ++;
CurrentPage = CurrentPage + BootSearchStride
YES
Is Page DBBT
NO
ReadCount < BootSearchCount
YES
NO
DBBT Found
DBBT Not Found

Figure 12-6. DBBT Search Process Flowchart

The BCB search and the load function also monitors the ECC correction threshold and sets the NAND_SDK_BLOCK_REWRITE persistent bit if the threshold exceeds the max ecc correction ability. One bit is used for all boot block images. If the NAND_SDK_BLOCK_REWRITE bit is set, the ROM continues loading the image, but the SDK needs to refresh the boot blocks at a later time.

12.12.1.5 Redundant Boot Support in ROM NAND driver

ROM checks the state of ROM_SECONDARY_BOOT persistent bit to decide which firmware to load, either primary or secondary. If the bit is not set, it will load the primary image. Otherwise, it will load the secondary image.

If ROM fails to load from primary boot image due to ECC failures, it will set the persistent bit ROM_SECONDARY_BOOT and soft resets the chip to let ROM come up again and load from secondary image. The persistent bit will retain its status until power down.

If ROM NAND driver fails to load from primary boot image due to non-ECC failures like invalid header, and so on, it will return an error code to the ROM loader. The secondary boot will be initiated by the loader if the driver supports redundant boot. ROM NAND driver and ROM SD driver are the only two drivers that currently support ROM redundant boot.

12.12.1.6 NAND Patch Boot using FCB

A secondary mechanism to boot NAND patch image is implemented in i.MX28 ROM. A flag is used in the FCB data structure which indicates to the ROM to boot the patch binary image present on the second page of first good block of NAND. If this flag is set, the ROM does not try to locate other boot blocks, but rather starts loading the patch image.

12.12.1.7 Expected NAND Layout

This section shows several potential expected NAND layouts.

The following figure shows the default layout if no efuses are blown.

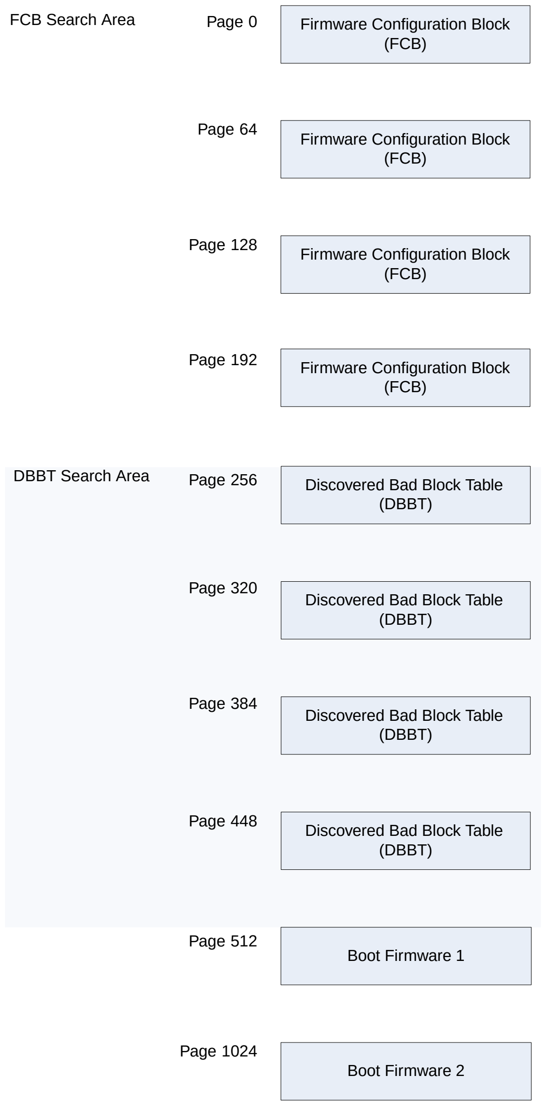

FCB Search Area
Page 0
Firmware Configuration Block (FCB)
Page 64
Firmware Configuration Block (FCB)
Page 128
Firmware Configuration Block (FCB)
Page 192
Firmware Configuration Block (FCB)
DBBT Search Area
Page 256
Discovered Bad Block Table (DBBT)
Page 320
Discovered Bad Block Table (DBBT)
Page 384
Discovered Bad Block Table (DBBT)
Page 448
Discovered Bad Block Table (DBBT)
Page 512
Boot Firmware 1
Page 1024
Boot Firmware 2

Figure 12-7. Expected NAND Layout with search count = 4 and search stride = 1*64

To work with the following layout, efuses need to be blown, efNANDBootSearchCount to 0x1 and efNANDBootSearchStride to 0x2.

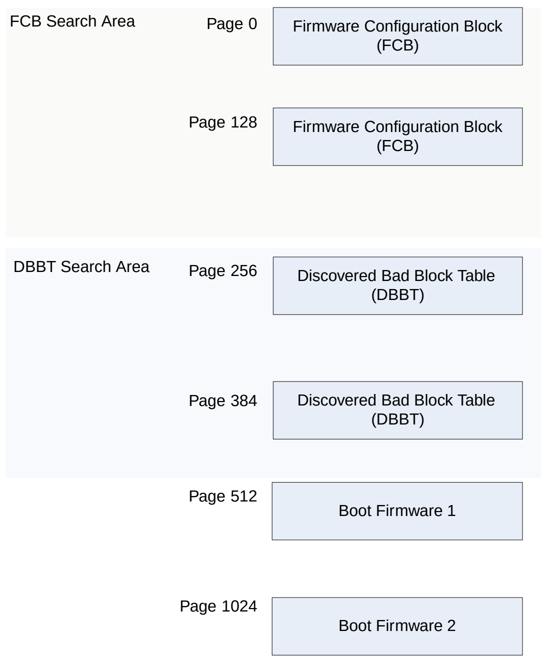

FCB Search Area
Page 0
Firmware Configuration Block (FCB)
Page 128
Firmware Configuration Block (FCB)
DBBT Search Area
Page 256
Discovered Bad Block Table (DBBT)
Page 384
Discovered Bad Block Table (DBBT)
Page 512
Boot Firmware 1
Page 1024
Boot Firmware 2

Figure 12-8. Expected NAND Layout with search count = 2 and search stride = 2*64

To work with the following layout efuses need to be blown, efNANDBootSearchCount to 0x1 and efNANDBootSearchStride to 0xF.

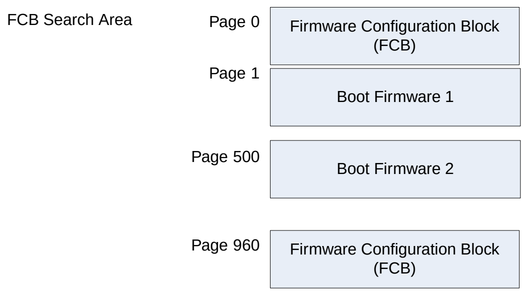


Figure 12-9. Expected NAND Layout for patch boot with search count = 2 and search stride = 15*64 and no DBBT

To work with the following layout, efuses need to be blown, efNANDBootSearchCount to 0x1 and efNANDBootSearchStride to 0xF.

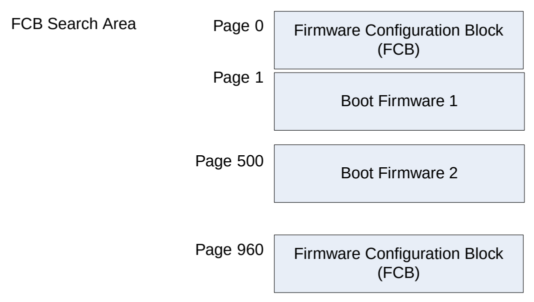


Figure 12-10. Expected NAND Layout for patch boot with search count = 2 and search stride = 15*64 and valid DBBT

12.12.1.8 Firmware Configuration Block

The FCB is the first sector in the first good block. The FCB should be present at each search stride of the search area. The search area contains copies of the FCB at each stride distance, so in case the first NAND block becomes corrupted, the ROM will find its copy in the next NAND block. The search area should span over at least two NAND blocks. The location information of DBBT search area, FW1 and FW2 are all specified in the FCB. This case is shown in more detail in Firmware Layout on the NAND.

The layout of the first good page containing FCB is shown in the following figure.

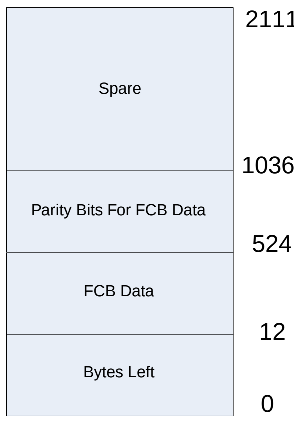


Figure 12-11. Layout of Boot Page Containing FCB

The FCB is located on the first good page of the NAND; the minimum size of a page is 2112 bytes. The first 12 bytes of an FCB page are reserved and left blank (all zeros); the next 512 bytes are reserved for FCB data. The remaining 512 bytes are available for software ECC, and the rest are all zeros. FCB is protected using SEC-DED Hamming codes.

12.12.1.9 Single Error Correct and Double Error Detect (SEC-DED) Hamming

For each 8-bit of data in the 512 byte FCB, 5-bit parity is used. Each byte of parity area contains 5 parity bits (LSB) and 3 unused bits (MSB).

For each 8-bit of FCB data, parity is calculated and compared with the corresponding parity bits read from the parity area of the FCB page to detect errors and correct any single error.

If there is more than one error, a flag is raised against the block.

To determine a good FCB, all three fingerprints must match and the ECC must not fail.

The data held in the FCB includes the following:

- NAND Timing parameters
- Geometry information (sectors per block, sectors per page, and so on)
- The page address of the discovered bad block table (DBBT)
- BCH ECC type
- A flag to boot the NAND patch image located at sector 2 of first block
- Starting page addresses of primary and secondary firmware
- Bad block marker bit offset in page data

Additionally, the FCB is marked with three fingerprints in the sector.

12.12.1.10 Firmware Layout on the NAND

The boot image is typically located on the first good block after the FCB, DBBT blocks and any additional blocks reserved for BCBs in case they go bad.

ROM shall support boot from only first NAND. In case of multi-NAND system, both firmware copies should be located on first NAND, same as in the single-NAND system.

The DBBT will be located in its own search area and a copy of DBBT will be present at each stride of the search area.

12.12.1.11 Recovery From a Failed Boot Firmware Image Read

The mechanism for recovering from a failed FCB read is covered in Boot Control Blocks (BCB). The SDK is warned about impending errors with the NAND_SDK_BLOCK_REWRITE persistent bit and is notified of firmware boot errors with the ROM_SECONDARY_BOOT persistent bit.

In the case of a warning, the read routine will monitor the ECC threshold and set the NAND_SDK_BLOCK_REWRITE bit if the threshold is within one symbol of the maximum correctable number of symbols. The ROM continues to load from the primary boot image. At a later time, the SDK will refresh the primary boot images by copying and rewriting the primary boot image blocks.

If an error is discovered while reading the boot firmware image, the NAND driver sets the ROM_SECONDARY_BOOT persistent bit and resets the device. Booting will proceed the second time from the secondary boot images. If booting continues uninterrupted, no unwinding needs to take place at the loader level.

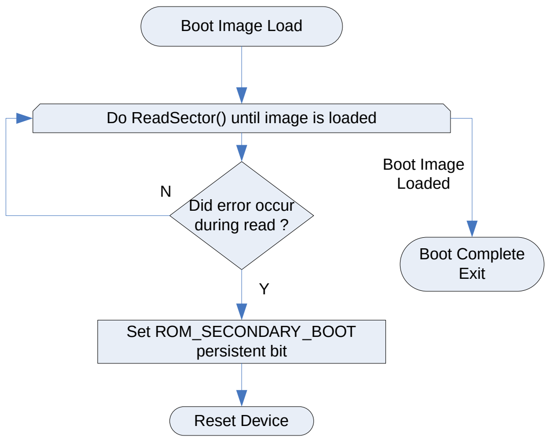

Boot Image Load
Do ReadSector() until image is loaded
Boot Image Loaded
N
Did error occur during read ?
Boot Complete Exit
Y
Set ROM_SECONDARY_BOOT persistent bit
Reset Device

Figure 12-12. Boot Image Recovery

12.12.1.12 Bad Block Handling in the ROM

Bad blocks are not an issue for the FCB, because FCB found from a search. The search for the DBBT works with a similar mechanism. The search starts where the FCB indicates the DBBT Search Area should be and progresses until efNANDBootSearchLimit times in the same fashion as the search described in Boot Control Blocks (BCB).

ROM uses DBBT to skip any bad block that falls within firmware data on NAND Flash device.

If the address of DBBT Search Area in FCB is 0, ROM will rely on factory marked bad block markers to find out if a block is good or bad. The location of bad block information is at the first 3 or last 3 pages in every block of the NAND Flash. NAND manufacturers normally use one byte in the spare area of certain pages within a block to mark a block to be good or bad. 0xFF means good block, non FF means bad block.

In order to preserve the BI (bad block information), flash updater or gang programmer applications need to swap Bad Block Information (BI) data to byte 0 of metadata area for every page before programming NAND Flash. ROM when loading firmware, copies back the value at metadata[0] to BI offset in page data. The following figure shows how the factory bad block marker is preserved.

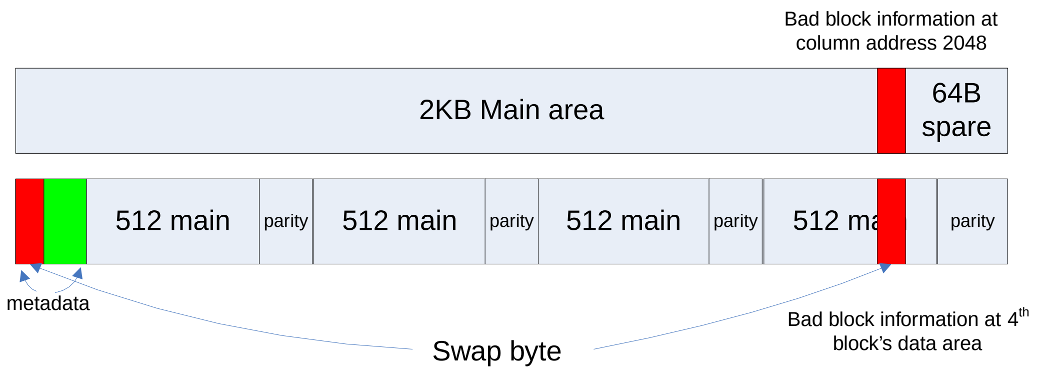


Figure 12-13. Factory Bad Block Marker Preservation

In the FCB structure, there are two elements m_u32BadBlockMarkerByte and m_u32BadBlockMarkerStartBit to indicate the byte offset and start bit of BI. ROM will use 8 bits from start bit as BI.

The DBBT structure is contained within a block and is discussed in more detail below. The figure below depicts the layout of the Discovered Bad Block Table block. The first 8K are reserved for the DBBT Header. The following pages are used by the DBBT for each NAND.

The Bad Block table for each NAND begins on a 2 KB boundary that coincides with current NAND page sizes and is a subset of future NAND page sizes. The BBT can extend beyond 2K, which is the purpose of the #2K_num, and translates to the number of 2K pages that NAND0 through NAND3 require. In this way, the ROM can quickly index to the appropriate NAND table.

Only the Bad Blocks for NAND0 are required and loaded.

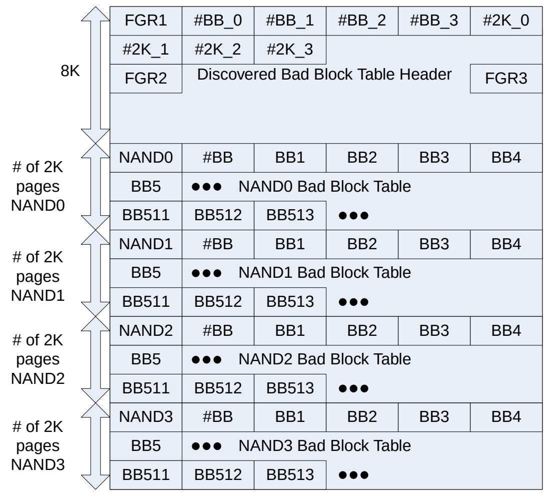


Figure 12-14. DBBT Structure

12.12.1.13 Firmware Configuration Block Structure and Definitions

The FCB structure is as follows:

Table 12-30. Firmware Configuration Block Structure

Name	Start Byte	Size in bytes	Description
m_u32Checksum	0	4	32 bit checksum of 508 bytes of FCB data from offset 4 to 512 XOR with 0xFFFFFFFF.
m_u32FingerPrint	4	4	32 bit word with a value of 0x20424346, in ascii “FCB ”.
m_u32Version	8	4	32 bit version number; this version of FCB is 0x01000000.
m_NANDTiming	12	4	8 bytes of data for 8 NAND Timing Parameters from NAND datasheet. The 8 parameters are: data_setup, data_hold, address_setup, dsample_time, nand_timing_state, REA, RLOH, RHOH.
m_u32PageDataSize	20	4	Number of bytes of data in a page. Typically, this is 2048 bytes for 2112 bytes page size or 4096 bytes for 4314 bytes page size.
m_u32TotalPageSize	24	4	Total number of bytes in page. Typically, 2112 for 2K page or 4224 or 4314 for 4K page.
m_u32SectorsPerBlock	28	4	Number of pages per block. Typically 64 or 128 or depending on NAND device type.
m_u32NumberOfNANDs	32	4	Number of NAND devices present on the chip, this is ignored in i.MX28 ROM. May be useful for by other applications.

Name	Start Byte	Size in bytes	Description
m_u32TotalInternalDie	36	4	Number of internal dice present on the NAND chip. Not used by ROM.
m_u32CellType	40	4	MLC or SLC, not used by ROM.
m_u32EccBlockNEccType	44	4	Type of BCH Error Correction Level used for encoding BCH Blocks B1 through BN. Should be one of the following: 0 - BCH0 1 - BCH2 2 - BCH4 3 - BCH6 4 - BCH8 5 - BCH10 6 - BCH12 7 - BCH14 8 - BCH16 9 - BCH18 10 - BCH20
m_u32EccBlock0Size	48	4	Size of BCH Block B0. Typically 512 but can be any value limited to 900 bytes.
m_u32EccBlockNSize	52	4	Size of BCH Blocks B1 through BN. Typically 512 but can be any value limited to 900 bytes.
m_u32EccBlock0EccType	56	4	Type of BCH Error Correction Level used for encoding BCH Block B0. Should be one of the following: 0 - BCH0 1 - BCH2 2 - BCH4 3 - BCH6 4 - BCH8 5 - BCH10 6 - BCH12 7 - BCH14 8 - BCH16 9 - BCH18 10 - BCH20
m_u32MetadataBytes	60	4	Number of bytes in metadata of a page. Minimum value allowed is 1. If set to 0, ROM may not boot the image successfully.

Name	Start Byte	Size in bytes	Description
m_u32NumEccBlocksPerPage	64	4	Number of BCH Blocks excluding B0.
m_u32EccBlockNEccLevelSDK	68	4	This is for SDK/Application use only. Not used by ROM.
m_u32EccBlock0SizeSDK	72	4	This is for SDK/Application use only. Not used by ROM.
m_u32EccBlockNSizeSDK	76	4	This is for SDK/Application use only. Not used by ROM.
m_u32EccBlock0EccLevelSDK	80	4	This is for SDK/Application use only. Not used by ROM.
m_u32NumEccBlocksPerPageSDK	84	4	This is for SDK/Application use only. Not used by ROM.
m_u32MetadataBytesSDK	88	4	This is for SDK/Application use only. Not used by ROM.
m_u32EraseThreshold	92	4	This value will be set into BCH_MODE register.
m_u32BootPatch	96	4	0 for normal boot and 1 for NAND patch boot.
m_u32PatchSectors	100	4	Size of patch in pages.
m_u32Firmware1_StartingSector	104	4	Starting page address of primary boot firmware.
m_u32Firmware2_StartingSector	108	4	Starting page address of secondary boot firmware.
m_u32SectorsInFirmware1	112	4	Number of pages occupied by primary boot firmware.
m_u32SectorsInFirmware2	116	4	Number of pages occupied by secondary boot firmware.
m_u32DBBTSearchAreaStartAddress	120	4	Starting page address of DBBT Search Area.
m_u32BadBlockMarkerByte	124	4	Byte offset in page data containing manufacturer marked bad block marker information.
m_u32BadBlockMarkerStartBit	128	4	For BCH ECC Levels other than BCH8 or BCH16 BI byte does not start at a byte boundary. This field will give the start bit number of BI in m_u32BadBlockMarkerByte.
m_u32BBMarkerPhysicalOffset	132	4	This is the physical byte offset where manufacturer marked bad block marker.
-	136	376	Spare bytes, should be set to 0.

12.12.1.14 Discovered Bad Block Table Header Layout Block Structure and Definitions

The first 512 bytes of the DBBT header structure are as follows:

Table 12-31. DPPT Header Structure

Name	Start Byte	Size in Bytes	Description
m_u32Checksum	0	4	32 bit checksum of 508 bytes of DBBT data from offset 4 to 512 XOR with 0xFFFF_FFFF
m_u32FingerPrint	4	4	32 bit word with a value of 0x54424244, in ascii “DBBT”
m_u32Version	8	4	32 bit version number; this version of DBBT is 0x0100_0000
m_u32NumberBB	12	4	Number of bad blocks present in the table.
m_u32Number2KPagesBB	16	4	Bad blocks consume these many 2048 byte size pages.
—	20	492	Spare bytes, should be set to 0.

12.12.1.15 Discovered Bad Block Table Layout Block Structure and Definitions

The actual bad block table structure is as follows:

Table 12-32. Bad Block Table Structure

Name	Start Byte	Size in Bytes	Description
uNAND	0	4	NAND number
uNumberBB	4	4	Number of entries in DBBT
u32BadBlock	8	2040	Array or Table for bad block entries, each 4 byte word in this array contain the block number that is bad.

12.12.2 Typical NAND Page Organization

This section discusses the typical NAND page organization. In particular, it discusses the BCH ECC page organization, metadata requirements, and efuses/OTP bits used by the ROM NAND driver

12.12.2.1 BCH ECC Page Organization.

For NAND boot, ROM restricts the size of a BCH data block to 512 bytes. The first data block is called block 0 and the rest of the blocks are called block N. Separate ECC levels can be used for block 0 and block N. The metadata bytes should be located at the beginning of a page, starting at byte 0, followed by data block 0, followed by ECC bytes for data block 0, followed by block 1 and its ECC bytes, and so on until N data blocks. The ECC level for block 0 can be different from the ECC level of rest of the blocks. Metadata bytes can be 0.

For NAND boot, with page size restrictions and data block size restricted to 512 bytes, only few combinations of ECC for block 0 and block N are possible.

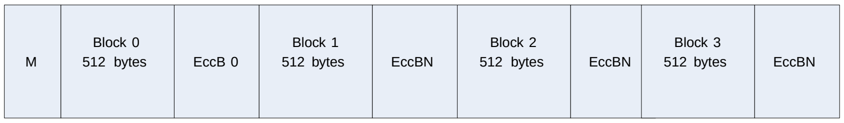


Figure 12-15. Valid layout for 2112 bytes sized page

The number of ECC bits required for a data block is calculated using (ECC_Correction_Level * 13) bits.

In the above layout, the ECC size for EccB0 and EccBN should be selected to not exceed a total page size of 2112 bytes. EccB0 and EccBN can be one of 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 bits ECC correction level. The total bytes would then be:

[M + (data_block_size * 4) + ([EccB0 + (EccBN * 3)] * 13) / 8] <= 2112; M=metadata bytes and data_block_size is 512.

There are four data blocks of 512 bytes, each in a page of 2K page sized NAND. The values of EccB0 and EccBN should be such that the above calculation would not result in a value greater than 2112 bytes.

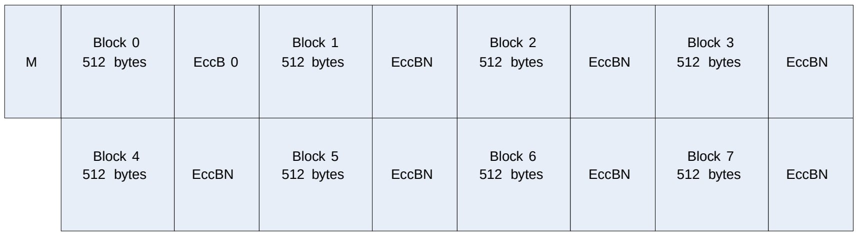


Figure 12-16. Valid layout for 4K bytes sized page

Different NAND manufacturers have different sizes for a 4K page, 4314 bytes is typical.

[M + (data_block_size * 8) + ([EccB0 + (EccBN * 7)] * 13) / 8] <= 4314; M=metadata bytes and data_block_size is 512.

There are eight data blocks of 512 bytes, each in a page of a 4K page sized NAND. The values of EccB0 and EccBN should be such that above calculation should not result in a value greater than the size of a page in a 4K page NAND.

12.12.2.2 Metadata

The number of bytes used for metadata are specified in FCB. Metadata for BCH encoded pages will be placed at the beginning of a page. ROM only cares about the first byte of metadata to swap it with bad block marker byte in page data after each page read. It is therefore important to have at least one byte for metadata bytes field in FCB data structure.

12.12.2.3 efuses/OTP bits used by ROM NAND Driver

Table 12-33. eFuses

Register	Name	Description
ROM1:2..0	NUMBER_OF_NANDS	Three bits used for number of NANDs on the device. If left unblown, boot ROM will default to 1 device. It is important to program these bits to exact number of NANDs in the system, using this information ROM will enable internal pullups. The boot may fail if this field is incorrectly programmed. Max number of NAND devices allowed in MX28 is 8, but ROM only supports boot from NAND0.
ROM1:7..4	BOOT_SEARCH_STRIDE	Four bits for boot search stride. Boot ROM defaults it to 1. ROM multiplies this number with 64 to get search stride in pages. Typically, this should be left 0 or 1 for NAND devices with number of pages per block as 64. For 128 pages per block, it is recommended to program these bits to a value of 2 in order for each BCB to be placed in a new block.
ROM1:11..8	BOOT_SEARCH_COUNT	Four bits for boot search count. If not programmed, boot ROM will default to 2. This value is raised to the power of 2 to get actual boot search count. Boot search count will tell ROM how many times a BCB is duplicated in a search area.
ROM1:27..20	ENABLE_NAND_CE_RDY_PULLUPS	Eight bits to enable internal pullups on CE and RDY pins. Bit 20 is used to enable pullups on NAND0, Bit 21 is used to enable pullups on NAND1,... ... Bit 27 is used to enable pullups on NAND7.
ROM1:30	DISABLE_SECONDARY_BOOT	One bit to disable redundant boot. If this bit is programmed to 1, ROM will not try to boot from the secondary image if the primary image failed to boot.
ROM4:3..0	NAND_ROW_ADDRESS_BYTES	Four bits for number of row address bytes. If not programmed, ROM will default to 3 bytes for row (page) address.
ROM4:7..4	NAND_COLUMN_ADDRESS_BYTES	Four bits for number of column address bytes. If not programmed, ROM will default to 2 column address bytes.

Register	Name	Description
ROM4:15..8	NAND_READ_CMD_CODE1	Eight bits for NAND read cmd code 1. If not programmed, ROM will default to 0 as NAND read cmd code 1. For BA NAND, these bits should be programmed to a value of 0xC0.
ROM4:23..16	NAND_READ_CMD_CODE2	Eight bits for NAND read cmd code 2. If not programmed, ROM will default to 0x30 as NAND read cmd code 2.
ROM4:31	NAND_BADBLOCK_MARKER_PRE-SERVE_DISABLE	One bit to indicate that bad block marker byte is not preserved for page data, this will result in ROM not swapping metadata[0] with bad block byte offset in page data. This bit should never be programmed. It is available to cover up any defective ROM code in handling bad block marker byte swapping.

12.12.3 ONFi BA NAND Device Boot

i.MX28 ROM supports boot from ONFI-BA (Open NAND Flash Interface - Block Abstracted) NAND. BA-NAND manages ECC, bad-blocks and wear-leveling inside its controller. ROM reads ONFI NAND device's parameter page to determine if the NAND device is ONFI-BA. The command set of BA NAND is different from the traditional raw NAND devices. For ONFI-BA NAND, ROM expects the first sector to contain a MBR with partition table. One of the partitions is firmware partition. ROM then expects a configuration block to be present on the first sector of firmware partition. The configuration block will have start address for FW1 and FW2.

Here is the data structure of configuration block, it is same as config block described in SD Boot.

```
#define FIRMWARE_CONFIG_BLOCK_SIGNATURE     (0x00112233) //STMP

 typedef struct _DriveInfo_t
 {
 uint32_t    u32ChipNum;      //!< Chip Select, ROM does not use it
 uint32_t    u32DriveType;    //!< Always system drive, ROM does not use it
 uint32_t    u32Tag;   //!< Drive Tag
 // Below field u32FirstSectorNumber should be divisible by 4. Protocol is set to 4 sectors
 // of 512 bytes. Firmware can start at sectors 4, 8, 12, 16, ...
 uint32_t    u32FirstSectorNumber;   //!< Start sector/block address of firmware.
 uint32_t    u32SectorCount;  //!< Not used by ROM
 } DriveInfo_t;

 typedef struct _ConfigBlock_t
 {
 uint32_t    u32Signature;    //!< Signature 0x00112233
 uint32_t    u32PrimaryBootTag;      //!< Primary boot drive identified by this tag
 uint32_t    u32SecondaryBootTag;    //!< Secondary boot drive identified by this tag
 uint32_t    u32NumCopies;    //!< Num elements in aFWSizeLoc array
 DriveInfo_t aDriveInfo[];    //!< Let array aDriveInfo be last in this data
 //!< structure to be able to add more drives in future
 //!< without changing ROM code
 } ConfigBlock_t;
```

12.13 USB Boot Driver

The USB Boot Driver is implemented as a USB HID class and is referred to as the Recovery HID, or RHID. The RHID serves as a fail-safe mechanism for downloading and communicating with application-specific code.

The system is based on two HID Application collections: the Boot Loader Transfer Controller (BLTC) and the Plug-in Transfer Controller (PITC). Each collection has its own set of HID reports.

12.13.1 Boot Loader Transaction Controller (BLTC)

The BLTC provides a tunnel to download application-specific PITCs to the local system memory. The BLTC runs completely from ROM and interfaces directly to the ROM Loader. Typically, a PITC will be packaged on the host and downloaded through the BLTC and ROM Loader straight to OCRAM (or SDRAM).

Four HID reports are provided for communication with the BLTC:

- BLTC Command Out (BLCO)
- BLTC Data Out (BLDO)
- BLTC Data In (BLDI)
- BLTC Status In (BLSI)

The BLTC provides a command/data protocol for downloading code to the ROM Loader.

The BLTC has no knowledge of the contents of the data passing through so, it is really possible to download anything (not just PITCs).

12.13.2 Plug-in Transaction Controller (PITC)

The PITC is a generic command/data/status tunnel that may be used for any type of application. The implementation only specifies the HID report structure and the ROM HID-stack installation for a PITC—the protocol implementation is specific to a given PITC. Typically, a PITC will be downloaded to memory through the BLTC.

Four HID reports are provided for communication with a PITC:

- PITC Command Out (PICO)
- PITC Data Out (PIDO)

- PITC Data In (PIDI)
- PITC Status In (PISI)

The command/data protocol is specific to any given PITC.

Only one PITC may be installed at any given time.

12.13.3 USB IDs and Serial Number

By default, the USB Device Descriptor Vendor ID, Product ID, and serial number are reported as follows:

- Vendor ID—0x15A2
- Product ID—0x004F
- Serial Number String—none

If any of the HW_OCOTP_ROM2 bits are blown, then the full contents of that OTP register are used to report the Vendor ID and Product ID. If the ENABLE_USB_BOOT_SERIAL_NUMBER OTP is blown, then a unique serial number will be generated from the factory-programmed SGTL_OPS3 OTP registers.

12.13.4 USB Recovery Mode

USB boot mode is provided as a fail-safe mechanism for writing system firmware to the boot media. The boot mode is not usually entered by the normal methods of setting the boot pins or OTP, the other methods of entering USB boot mode are referred to generally as recovery mode.

An end user can manually start the recovery mode by holding the recovery switch for several seconds while plugging in USB. Holding the recovery switch is defined as reading the i.MX28 PSWITCH input as a 0x3. There are several switch circuits that will produce this input. The loader also automatically starts recovery mode if a non-recoverable error is detected from any boot mode other than USB.

If the DISABLE_RECOVERY_MODE OTP bit is blown, then USB boot mode is disabled completely. Attempts to enter USB boot mode through boot pins, OTP, or recovery methods will result in a chip power-down.

Chapter 13
Data Co-Processor (DCP)

13.1 Data Co-Processor (DCP) Overview

The DCP module provides support for general encryption and hashing functions typically used for security functions. Because the basic job of the DCP module is moving data from memory to memory, the DCP module also incorporates memory-copy (memcopy) function for both debugging and as a more efficient method of copying data between memory blocks than the DMA-based approach. The memcopy function also has a blit mode of operation where it can transfer a rectangular block of data to a video frame buffer.

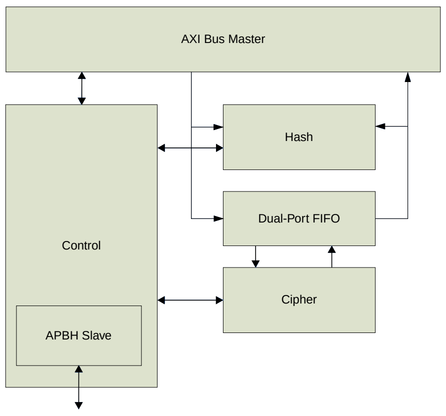


Figure 13-1. Data Co-Processor (DCP) Block Diagram

The i.MX28 DCP module implementation supports AES-128 encryption and CRC32, SHA-1, and SHA-256 hashing (see the Capability Register).

The DCP module processes data based on chained command structures written to the system memory by software (in a manner similar to the DMA engine). The control block maintains registers to support four independent and concurrent chains, allowing software to virtualize access to the DCP block. Each command in a chain represents a work unit that the module will process to completion. At the end of each work unit, the control logic arbitrates among chains with outstanding commands and processes a command from that chain. Arbitration among the channels is round-robin, allowing all active channels equal access to the data engine. Each channel also supports a high-priority mode that allows it to have priority over the remaining channels. If multiple channels are selected as high-priority, the channel arbiter selects among the high-priority channels in a round-robin fashion.

The data flow through the DCP module can be configured in one of five fashions, depending on the functionality activated by the control packet:

- **Memcopy/Blit Mode**—Data is moved unchanged from one memory buffer to another.

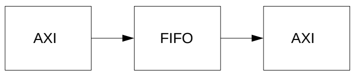


Figure 13-2. Memcopy/Blit Mode

- **Encryption Only**—Data from source buffer is encrypted/decrypted into the destination buffer.

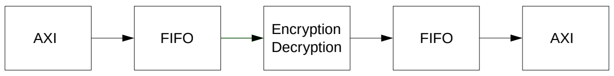


Figure 13-3. Encryption Only

- **Hashing Only**—Data from source buffer is read, and a hash is generated.

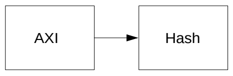


Figure 13-4. Hashing Only

- **Encryption and Input Hashing**—Data from source buffer is encrypted/decrypted into destination, and source buffer is hashed.

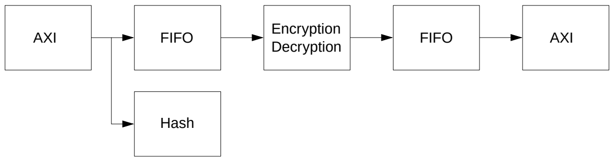


Figure 13-5. Encryption and Input Hashing

- **Encryption and Output Hashing**—Data from source buffer is encrypted/decrypted into destination, and output data is hashed.

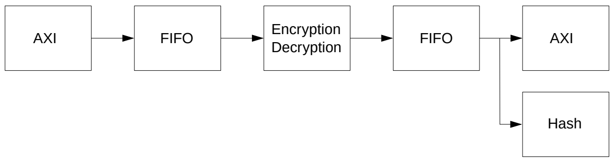


Figure 13-6. Encryption and Output Hashing

13.1.1 DCP Limitations for Software

While the DCP module has been designed to be as flexible as possible, there are a few limitations to which software must adhere:

- Buffer sizes for all operations MUST be aligned to the natural size of the transfer algorithm used. Memcopy operations can transfer any number of bytes (one-byte granularity) and AES operations must be multiples of 16 bytes (four-word granularity). For all operations, if the byte count is not a word granularity, the hardware rounds up to the next word. Hashing is supported at a byte granularity.
- The DCP module supports buffer operations to any byte alignment, but performance will be improved if buffers are aligned to a four-byte boundary, since fetch/store operations can be performed without having to do byte operations to accommodate the misaligned addresses.
- Hash operations are limited to a 512 Mbyte buffer size. The hardware only implements a 32-bit hash length counter instead of the 64-bit counter supported by the SHA-1/SHA-256 algorithm (counter counts bits, not bytes, therefore a total of 512 Mbytes).
- For chained hashing operations (operations involving multiple descriptors), every descriptor except the last must have a byte count that is a 16-word multiple (granularity of the hash algorithm).
- Key values cannot be written while the AES block is active. This limitation exists because the key RAM is in use while AES is operational. Any writes from the APB cannot be held in wait states; therefore, the RAM must be accessible during key writes.
- The byte-swap controls can only be used with modulo-4 length buffers. For non-modulo-4 lengths, the final partial word will contain incorrect data. Any address alignment can be used with byte swapping, however.
- The word-swap controls are only useful with cipher operations, because the logic forms the 128-bit cipher data from four words from system memory. The word-swap controls are ignored for memcopy or hashing operations.
- DCP only supports writes to word boundaries to OCRAM. This is not required for EMI DRAM address.

13.2 Operation

The top-level DCP module contains the AXI master, APB slave bus interface units, the main control block and FIFO, and any encryption or hashing blocks included with the design.

The controller manages the fetching of work blocks, the fetching/storing of context information when switching between chain pointers, and the data flow through FIFO, SHA, and AES blocks. Data entering the block from the AXI master is placed in the FIFO for consumption by the cipher block. After the cipher module has finished its operation, data is placed back into the FIFO and stored back to memory through the AXI master. When hashing is enabled, the SHA block takes its inputs from the bus side of the FIFO to allow it to operate without waiting for the cipher block to complete. The APB slave provides all register controls and interfaces mainly with the control block.

13.2.1 Memory Copy, Blit, and Fill Functionality

In its most basic operation, the DCP supports moving unmodified data from one place in system memory to another. This functionality is referred to as memcopy, because it operates only on memory and it copies data from one place to another. Typical uses of memcopy might be for fast virtual memory page moves or repositioning data blocks in memory. Memcopy buffers can be aligned to any memory address and can be of any length (byte granularity). For best performance, buffers should be word-aligned, although the DCP includes enhancements to improve performance for unaligned cases.

The DCP also has the ability to perform basic blit operations that are typical in graphics operations. To specify a blit, the control packet must have the ENABLE_BLIT bit set in the packet control register. Blit source buffers must be contiguous. The output destination buffer for a blit operation is defined as a "M runs of N bytes" that define a rectangular region in a frame buffer. For blit operations, each line of the blit may consist of any number of bytes. After performing a run, the DCP updates the destination pointer such that the next destination address falls on the pixel below the start of the previous run operation. This is performed by incrementing the starting pointer by the frame buffer width, which is specified in Control1 field.

In addition to being able to copy data within memory, the DCP also provides a fill operation, where source data comes not from another memory location, but from an internal register (the source buffer address in the control packet). This is performed whenever the CONSTANT_FILL flag is set in the packet control register. This feature may be used with memcopy to prefill memory with a specified value or during a blit operation to fill a rectangular region with a constant color.

13.2.2 Advanced Encryption Standard (AES)

The AES block implements a 128-bit key/data encryption/decryption block as defined by the National Institute of Standards and Technology (NIST) as US FIPS PUB 197, dated November 2001 (see references for specifications and toolkits)[1].

There are three variations of AES, each corresponding to the key size used: AES-128, AES-192 or AES-256. AES always operates on 128 bits of data at a time. Only the AES-128 algorithm is implemented at this time.

13.2.2.1 Key Storage

The DCP implements four SRAM-based keys that may be used by software to securely store keys on a semi-permanent basis. The keys may be written through the PIO interface by specifying a key index to specify which key to load and a subword pointer that indicates which word to write within the key. After a subword is written, the logic automatically increments the subword pointer so that software can program the higher-order words of the key without rewriting the key index. Keys written into the key storage are not readable.

To use a key in the key storage, the cipher descriptor packet should select the key by setting the KEY_SELECT field in the Control1 descriptor field without setting the OTP_KEY or PAYLOAD_KEY fields in the Control0 register.

13.2.2.2 OTP Key

After a system reset, the OTP controller reads the e-fuse devices and provides the OTP key information over a parallel 128-bit interface. The key transfer interface runs on HCLK and provides the key over the serialized otp_data signal. The otp_crypto_key_smpl signal indicates when the key value is valid and causes the control logic to capture the key into the key RAM.

To use the OTP key, the descriptor packet should set the OTP_KEY field in the Control1 register.

1. *The AES core used in the design was derived from the AES design available from OpenCores.org under a modified BSD license as described here: http://www.opencores.org/projects.cgi/web/aes_core/overview The license for this code is documented in Disclaimer for compliance.*

13.2.2.3 Encryption Modes

The most basic form of encryption is the Electronic Code Book (ECB) mode. In this mode, the encryption output is a function only of the key and the plaintext, therefore, it can be visualized as a giant lookup table. While this provides a great deal of security, there are a few limitations. For instance, if the same plaintext appears more than once in a block of data, the same ciphertext will also appear. This can be very evident if the plaintext contains large blocks of constant data (0s for example) and can be used to formulate attacks against a key.

In order to make ciphers stronger, several modes of operation can be implemented around the basic ECB cipher to provide additional security. One such mode is CBC mode (Cipher Block Chaining), which takes the previous encrypted data and logically XORs it with the next incoming plaintext before performing the encryption. During decryption, the process is reversed and the previous encrypted data is XORed with the decrypted ECB data to provide the plaintext again.

The AES engine supports handling the various modes of operation. The core AES block supports ECB mode and other algorithms are handled in the wrapper around the encryption blocks. The DCP module supports Cipher Block Chaining (CBC), which chains the data blocks by XORing the previously encrypted data with the plaintext before encryption. Cipher block chaining encryption is illustrated in Figure 13-7.

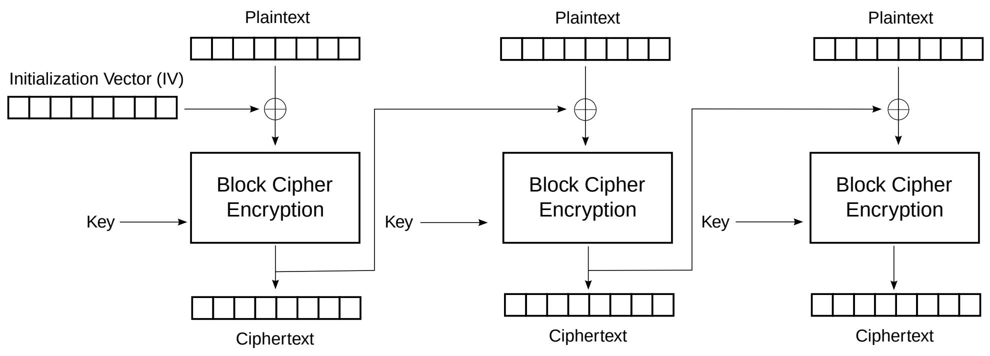


Figure 13-7. Cipher Block Chaining (CBC) Mode Encryption

Decryption (shown in the following figure) works in a similar manner, where the cipher text is first decrypted and then XORed with the previous ciphertext. For the first encryption/decryption operation, an initialization vector (IV) is used to seed the operation. The IV must be the same for both the encryption and decryption steps; otherwise, decrypted data will not match the original plaintext.

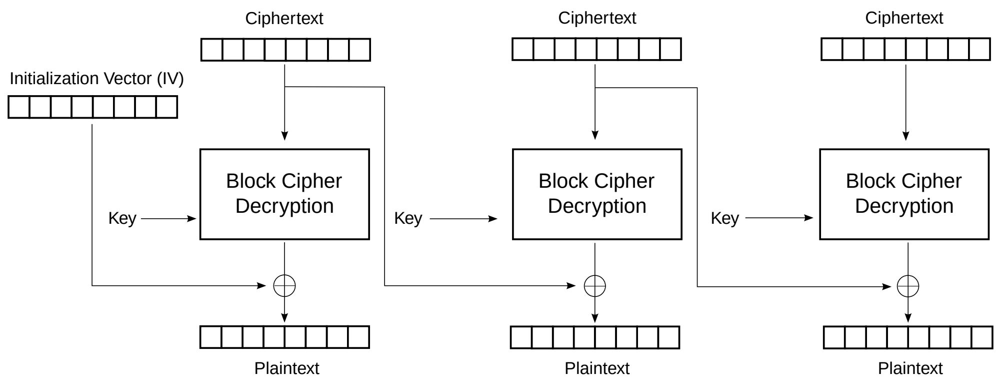


Figure 13-8. Cipher Block Chaining (CBC) Mode Decryption

13.2.3 Hashing

The hashing module implements SHA-1 and SHA-256 hashing algorithms and a modified CRC-32 checksum algorithm. These algorithms produce a signature for a block of data that can be used to determine whether the data is intact.

The CRC-32 algorithm implements a 32-bit CRC algorithm similar to the one used by Ethernet and many other protocols. The CRC differs from Unix cksum() function in three ways:

- The CRC is initialized as 0xFFFFFFFF instead of 0x00000000.
- Logic pads zeros to a 32-bit boundary for trailing bytes.
- Logic does not post-pend the file length.

The SHA-1 block implements a 160-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-1 in 1995. The SHA-256 mode implements a 256-bit hashing algorithm that operates on 512-bit (64-byte) blocks as defined by US FIPS PUB 180-2 in 2002. The purpose of the hashing module is to generate a unique signature for a block of data that can be used to validate the integrity of the data by comparing the resulting digest with the original digest.

Results from hash operations are written to the beginning of the payload for the descriptor. The DCP also has the ability to check the resulting hash against a value in the payload and issue an interrupt if a mismatch occurs.

13.2.4 OTP Key

After a system reset, the OTP controller will read the e-fuse devices and provide the OTP key information over a private parallel 32-bit interface. While the OTP key is normally not available to read operations, the module will allow the key to be read if the otp_crypto_key_ren (read-enable) input is active (high). This allows verification of the key after it has been programmed in to the e-fuse device. After programming has been validated, another fuse will be set to disable the read capability.

The OTP key may be selected using the OTP_KEY bit in the control field of the packet descriptor or by using the key select 0xFF in the CTRL1 field of the descriptor. The DCP also supports a second hardware key called the UNIQUE_KEY which is generated from the OTP KEY (OCOTP_CRYPTO0,1,2,3) and key modifier bits from another OTP fields (OCOTP_OPS2 and OCOTP_UN2) with unique number for every chip. This key is unique to the device and may be used for encrypting private data stored on the NAND. This key may be selected by writing 0xFE to the KEY_SELECT field in the CTRL1 packet data.

13.2.5 Managing DCP Channel Arbitration and Performance

The DCP can have four channels all competing for DCP resources to complete their operations. Depending on the situation, critical operations may need to be prioritized above less important operations to ensure smooth system operation. To help software achieve this goal, the DCP implements a programmable arbiter for internal DCP operations and provides recovery timers on each channel to throttle channel activity.

13.2.5.1 DCP Arbitration

The DCP implements a multi-tiered arbitration policy that allows software a lot of flexibility in scheduling DCP operations. The following figure illustrates the arbitration levels and where each channel fits into the arbitration scheme.

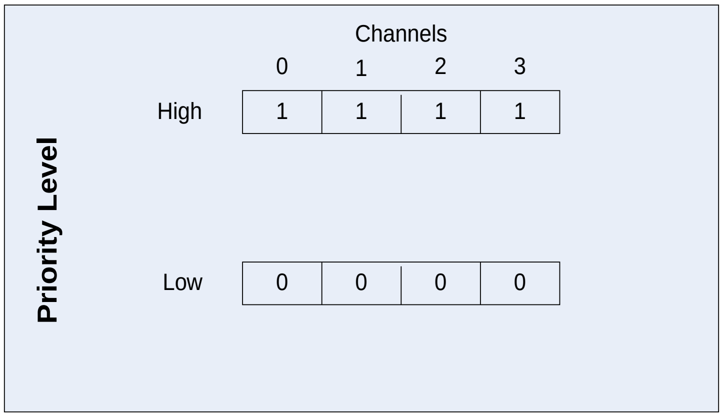


Figure 13-9. DCP Arbitration

Each channel can be programmed as being in either the high-priority or low-priority arbitration pool, depending on the setting in the HIGH_PRIORITY_CHANNEL field of the HW_DCP_CHANNELCTRL register. When the corresponding bit is programmed as a 1, the channel arbitrates in the high-priority pool; otherwise, it arbitrates in the low-priority pool. Once a channel has been selected, it completes one packet and then the arbiter re-arbitrates. The channel that just completed participates in the new arbitration round.

13.2.5.2 Channel Recovery Timers

Each channel also contains a channel recovery timer in its HW_DCP_CHnOPTS register. The purpose of the recovery timer is to keep the channel inactive for a period of time after it completes an operation. This capability could be used for a high-priority channel to ensure that at least some lower-priority requests get serviced between packets or to simply allow more timeslices for other channels to perform operations. The value programmed into the recovery timers register delays the channel from operations until 16 times the programmed value. Any non-zero value should prevent the channel from participating in the next arbitration cycle.

13.2.6 Programming Channel Operations

The control logic block maintains the channel pointers and manages arbitration and context switching between the different channels. It also manages the fetching of work packets and data fetch/store operations from the AXI master interface and coordinates the actions of the hashing and encryption blocks.

The control logic maintains four channels that allow software to effectively create four independent work sets for the DCP module. Software can construct chained control packets in memory that describe encryption/hashing/memcopy operations to the hardware unit. The address for this first control packet can be written to one of the four virtual channels and enabled. When one or more of the channels is enabled, the controller fetches the control packet pointed to by that channel and initiates data fetches from the source buffer. Data is then processed as described in the control packet and stored back to the system memory.

Once the processing for a control packet is complete, the controller writes completion status information back to the control packet, and optionally stores relevant state information to the context buffer. If the control packet specifies a subsequent control packet, the channel's pointer is updated to the address for the next packet and an optional interrupt can be issued to the processor.

At this point, the DCP module arbitrates among the virtual channels for the next operation and processes its control packet. If a subsequent operation is continued from a previous operation, the controller automatically loads the context from the previous session into the working registers before resuming operation for that channel.

13.2.6.1 Virtual Channels

The DCP module processes data through work packets stored in memory. Each of the channels contains a pointer to the current work packet and enough control logic to determine whether the channel is active and to provide status to the processor. Each channel also provides a recovery timer to help throttle processing by a particular channel. After processing a packet, the channel enables its 16-bit recovery timer (if the recovery time is set to a non-zero value). The channel will not become active again until its recovery timer has expired. The recovery timer timebase is 16 HCLK cycles, so the timer acts as a 20-bit timer with the bottom four bits implicitly tied to 0. This provides an effective range of zero to 220-1 clocks or 0 ns to 7.8 ms at 133 MHz.

A channel is activated any time its semaphore is non-zero and its recovery timer is cleared. The semaphore can be incremented by software to indicate that the chain pointer has been loaded with a valid pointer. As the hardware completes the work packets, it decrements the semaphore if the Decrement Semaphore flag in the Control 0 field is set. Work packets may be chained together using the CHAIN or CHAIN_CONTIGUOUS bits in the Control0 field, which causes the channel to automatically update the work packet pointer to the value in the NEXT_COMMAND_ADDRESS field at the end of the current work packet.

13.2.6.2 Context Switching

The control logic maintains four virtual channels that allow the DCP block to time-multiplex encryption, hashing and memcopy operations, it must also retain state information when changing channels so that when a channel is resumed, it can resume the operation from where it left off. This process is called context-switching.

To minimize the number of registers used in the design, the controller saves context information from each channel into a private context area in the system memory. When initializing the DCP module, software must allocate memory for the context buffer and write the address into the Context Buffer Pointer register. As the DCP module processes packets, it saves the context information for each channel to the buffer after completing each control packet. When the channel is subsequently activated, the DCP module's internal registers are then reloaded with the proper context before resuming the operation.

Each channel reserves one-fourth of the context buffer area for its context storage. The context buffer consumes 208 bytes of system memory and is formatted as shown in the following table.

Table 13-1. DCP Context Buffer Layout

RANGE	Channel	Data	Size
0x00–0x0C	3	Cipher Context	16 bytes
0x10–0x30		Hash Context	36 bytes
0x34–0x40	2	Cipher Context	16 bytes
0x44–0x64		Hash Context	36 bytes
0x68–0x74	1	Cipher Context	16 bytes
0x78–0x98		Hash Context	36 bytes
0x9C–0xA8	0	Cipher Context	16 bytes
0xAC–0xCC		Hash Context	36 bytes

The control logic writes to the context buffer only if the function is being used. This effectively means that the cipher context is stored for CBC encryption/decryption operations only and the hash context is written only if SHA-1/SHA-256 is utilized. If neither of these modes are used for a given channel, the memory for the context buffer need not be allocated by the software.

Since channel 0 is likely to be used for VMI in an SDRAM-based system-to-page data from the SDRAM to on-chip SRAM, the buffer allocation table has been organized so that the *highest* numbered channel uses the *lowest* area in the context buffer. For this reason, software should allocate the higher numbered channels for encryption/hashing operations and the lower numbered channels for memcopy operations to reduce the size of the context buffer.

If only a single channel is used for CBC mode operations or hashing operations, the controller provides a bit in the control register to disable context switching. In this scenario, context switching is not required, because other channels will not corrupt the state of the hashing or cipher modes. Therefore, when the channel resumes, a context load is not required.

Additionally, the control logic monitors the use of CBC/hashing, so that a context reload is not performed if the previous channel resumes an operation without an intermediate operation from another channel.

13.2.6.3 Working with Semaphores

Each channel has a semaphore register to indicate that control packets are ready to be processed. Several techniques can be used when programming the semaphores to control the execution of packets within a channel. The channel will continue to execute packets as long as the semaphore is non-zero, a chain bit is set in the descriptor and no error has occurred.

- Software can write the number of pending packets into the semaphore register with the Decrement Semaphore bit in each control packet set. In this scenario, the channel simply decrements the semaphore for each packet set and terminates at the end of the packet chain. The benefit of this method is that software can easily determine how many packets have executed by reading the semaphore status register.
- Software can create a packet chain with the Decrement Semaphore bit set only in the last packet. In this case, software would write a 1 to the semaphore register to start the chains and the DCP will terminate after executing the last packet.
- Software can create a packet chain with the Decrement Semaphore bit set for each packet and write either a 1 or a number less than the number of packets to the semaphore register. This can be useful when debugging, because it allows the channel to execute only a portion of the packets and software can inspect intermediate values before restarting the channel again.

If an error occurs, the channel issues an interrupt and clears the semaphore register. The channel does not perform any further operations until the error bits in the status register have been cleared. Software can manually clear a non-zero semaphore by writing 0xFF to the CLR (clear) address of the semaphore register.

13.2.6.4 Work Packet Structure

Work packets for the DCP module are created in memory by the processor. Each work packet includes all information required for the hardware to process the data. The general structure of the packet is shown below.

Table 13-2. DCP Work Packet Structure

Word0	Next Cmd Addr
Word1	Control0

Word2	Control1
Word3	Source Buffer Addr
Word4	Destination Buffer Addr
Word5	Buffer Size
Word6	Payload Pointer
Word7	Status

For some operations, additional information is required to process the data in the packet. This additional information is provided in the variable-sized payload buffer, which can be found at the address specified in the payload pointer field. When encryption is specified, the payload may include the encryption key (if the key selected resides in the packet), an initialization vector (for modes of operation such as CBC), and expected hash values when data hashing is indicated. The hardware can automatically compare the expected hash with the actual hash and interrupt the processor only on a mismatch. The payload area is used by software to store the calculated hash value at the end of hashing operations (when the HASH_TERM control bit is set).

13.2.6.4.1 Next Command Address Field

The NEXT_COMMAND_ADDRESS field (as shown in the following table) is used to point to the next work packet in the chain. This field is loaded into the channel's command pointer after the current packet has completed processing if the CHAIN bit in the CONTROL0 field (see Table 13-4) is set. The Next Command Address field should be programmed to a non-zero value when the CHAIN bit is set; otherwise, the channel will flag an invalid setup error.

Table 13-3. DCP Next Command Address Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NEXT_COMMAND_ADDRESS																															

13.2.6.4.2 Control0 Field

The main functions of the DCP module are enabled with the ENABLE_MEMCOPY, ENABLE_BLIT, ENABLE_CIPHER and ENABLE_HASH bits from the Control0 field in the work packet. The combinations of these bits determine the function performed by the DCP. Table 13-5 summarizes the function performed for each combination.

Table 13-4. DCP Control0 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

TAG	OUTPUT_WORDSWAP	OUTPUT_BYTESWAP	INPUT_WORDSWAP	INPUT_BYTESWAP	KEY_WORDSWAP	KEY_BYTESWAP	TEST_SEMA_IRQ	CONSTANT_FILL	HASH_OUTPUT	HASH_CHECK	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_KEY	CIPHER_INIT	CIPHER_ENCRYPT	ENABLE_BLIT	ENABLE_HASH	ENABLE_CIPHER	ENABLE_MEMCOPY	CHAIN_CONTINUOUS	CHAIN	DECR_SEMAPHORE	INTERRUPT_ENABLE

Table 13-5. DCP Function Enable Bits

HASH	Cipher	Blit	Memcopy	Description
0	0	0	X	Simple memcopy operation.
0	0	1	0	Blit operation.
0	1	0	0	Simple encrypt/decrypt operation.
1	0	0	0	Simple hash. Only reads performed.
1	0	0	1	Memcopy and hash operation.
1	1	0	0	Hash with encryption/decryption.
All Others				Invalid setup.

The CHAIN bit should be set if the NEXT_COMMAND_ADDRESS field has a valid pointer to the next work packet. When set, this bit causes the channel to update its pointer to the next packet. The CHAIN_CONTINUOUS bit is similar, but it indicates that the next packet follows immediately after the current packet. This allows software to generate templates of operations without regard to the actual physical addresses used, which makes programming easier, especially in a virtual memory environment.

If the INTERRUPT_ENABLE bit is set, the channel generates an interrupt to the processor after completing the work for this packet. When the interrupt is issued, the packet will have been completely processed, including the update of the status/payload fields in the work packet.

Each channel contains an eight-bit counting semaphore that controls whether it is in the run or idle state. When the semaphore in non-zero, the channel is ready to run and process commands. Whenever a command finishes its operation, it checks the DECR_SEMAPHORE bit. If set, it decrements the counting semaphore. If the semaphore goes to 0 as a result, then the channel enters the idle state and remains there until the semaphore is incremented by software. When the semaphore goes to non-zero and the channel is in its idle state, it then uses the value in the NEXT_COMMAND_ADDRESS field to begin processing.

If the ENABLE_CIPHER bit is set, the data is encrypted or decrypted based on the CIPHER_ENCRYPT bit using the key/encryption settings from the Control1 field. The OTP_KEY and PAYLOAD_KEY indicates the source of the keys for the operation. If the OTP_KEY value is set, the KEY_SELECT field from the Control1 register indicates which OTP key is to be used. If the PAYLOAD_KEY bit is set, the first entry in the payload is the key to be used for the operation. If neither bit is set, the KEY_SELECT field indicates an index in the key RAM that contains the key to be used. (For cases when both the OTP_KEY and PAYLOAD_KEY bits are set, the PAYLOAD_KEY takes precedence.)

The HASH_ENABLE enables hashing of the data with the HASH_OUTPUT bit controlling whether the input data (HASH_OUTPUT=0) or output data (HASH_OUTPUT=1) is hashed. In addition, the hardware can be programmed to automatically check the hashed data if the HASH_CHECK bit is set. If the hash does not match, an interrupt is generated and the channel terminates operation on the current chain. The hashing algorithms also require two additional fields in order to operate properly. The HASH_INIT bit should be set for the first block in a hashing pass to properly initialize the hashing function. The HASH_TERM bit must be set on the final block of a hash to notify the hardware that the hashing operation is complete so that it can properly pad the tail of the buffer according to the hashing algorithm. This flag also triggers the write-back of the hash output to the work packet's payload area.

The CONSTANT_FILL flag may be used for both memcopy and blit operations. When this is set, the source address field is used instead as a constant that is written to all locations in the output buffer.

The WORDSWAP and BYTESWAP bits enable muxes around the FIFO to swap data to handle little-endian and big-endian storage of data in system memory. When these bits are cleared, data is assumed to be in little-endian format. When all bits are set, data is assumed to be in big-endian format.

The TAG field is used to identify the work packet and the associated completion status. As each packet is completed, the channel provides the status and tag information of the last packet processed.

13.2.6.4.3 Control1 Field

The Control1 field (shown in the following table) provides additional values used when hashing or encrypting/decrypting data. For blit operations, this field indicates the number of bytes in a frame buffer line, which is used to calculate the address for successive lines in each blit operation.

Table 13-6. DCP Control1 Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

CIPHER_CONFIG					HASH_SE-LECT	KEY_SELECT	CIPHER_MODE	CIPHER_SE-LECT
						FRAMEBUFFER_LENGTH (in bytes, blit mode)		

The CIPHER_SELECT field selects from one of sixteen possible encryption algorithms (0=AES128). The CIPHER_MODE selects the mode of operation for that cipher (0=ECB, 1=CBC).

The KEY_SELECT field allows key selection for one of several sources. Keys can be included in the packet payload or may come from OTP or write-only registers.

The HASH_SELECT field selects a hashing algorithm for the operation (0=SHA-1, 1=CRC32, 2=SHA-256).

The CIPHER_CONFIG field provides the optional configuration information for the selected cipher. An example would be a key length for the RC4 algorithm.

13.2.6.4.4 Source Buffer

The SOURCE_BUFFER pointer (shown in the following table) specifies the location of the source buffer in memory. The buffer may reside at any byte alignment and should refer to an on-chip SRAM of off-chip SDRAM location. For optimal performance, buffers should be word aligned. When the CONSTANT_FILL flag is set in the Control0 field, the value in this field is used as the data written to all destination buffer locations.

Table 13-7. DCP Source Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
SOURCE_BUFFER																															
CONSTANT (CONSTANT_FILL mode)																															

13.2.6.4.5 Destination Buffer

The DESTINATION_BUFFER (shown in the following table) specifies the location of the destination buffer in on-chip SRAM or off-chip SDRAM memory and may be set to any byte alignment. For in-place encryption, the destination buffer and source buffer should be the same value. For optimal performance, the buffer location should be word-aligned.

Table 13-8. DCP Destination Buffer Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
DESTINATION_BUFFER																															

13.2.6.4.6 Buffer Size Field

The BUFFER_SIZE field (shown in the following table) indicates the size of the buffers for memcopy, encryption, and hashing modes. For memcopy and hashing operations, the value may be any number of bytes (byte granularity) and for encryption modes, the length must be a multiple of the selected encryption algorithm's natural data size (16 bytes for AES).

Table 13-9. DCP Buffer Size Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
NUMBER_LINES (blit mode)																BLIT_WIDTH (in bytes, blit mode)															
BUFFER_SIZE (in bytes, memcopy, encryption, hashing modes)																															

For blit operations, the buffer size field is split into two portions, a BLIT_WIDTH value, which specifies the number of bytes in each line of the blit operation and a NUMBER_LINES value, which specifies how many vertical rows of pixels are in the image buffer.

13.2.6.4.7 Payload Pointer

Some operations require additional control values that are stored in a Payload Buffer (shown in the table below), which is pointed to by this field. After the DCP reads the control packet, it examines the Control0 register and determines whether any payload information is required. If so, the DCP loads the payload from the address specified in this field. (See Payload for more details.)

Table 13-10. DCP Payload Buffer Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PAYLOAD_POINTER																															

The payload area is also written to by the DCP at the completion of a hash operation (when the HASH_TERM) bit is set. Software must allocate the appropriate amount of storage (20 bytes for SHA-1 and 4 bytes for CRC32) in the payload or risk having the DCP write to an unallocated address.

13.2.6.4.8 Status

After the DCP engine has completed processing a descriptor, it writes the packet status (shown in the following table) back to the descriptor In the Status field. The packet status is the value of the channel status register at the time the packet completed processing.

Table 13-11. DCP Status Field

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
TAG								ERROR_CODE																		ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	COMPLETE

For various error conditions, the DCP encodes additional error information in the ERROR_CODE field. See Programmable Registers for more details on assignments of error codes. For any completion codes besides the COMPLETE flag, the channel suspends processing the command chain until the error status values are cleared in the channel status register.

The format for this field is same as the channel status register, with the exception that the COMPLETE bit is written to the payload status but is not present in the channel status register.

13.2.6.4.9 Payload

The payload is a variable-length field that is used to provide key, initialization values, and expected results from hashing operations. The payload may consist of the data fields listed in the following table.

Table 13-12. DCP Payload Field

FIELD NAME	SIZE	DESCRIPTION	CONDITION
Cipher Key	16 bytes	AES key	Cipher enable with PAYLOAD_KEY
Cipher IV	16 bytes	CBC initialization vector	Cipher enable with CBC mode.
Hash Check	20 bytes	Hash completion value	Hash enabled with HASH_TERM fields set.

If fields are not used, they do not appear in the payload and the other payload values move upwards in the payload area. For instance, if only hashing is used, then the HASH_CHECK value would appear at offset 0 in the payload area. The following table should be used by software to determine the amount of payload to allocate and initialize.

Table 13-13. DCP Payload Allocation by Software

CONTROL BITS PRESENT			
HASH_CHECK	CIPHER_INIT	PAYLOAD_KEY	PAYLOAD SIZE
0	0	0	0 words / 0 bytes

CONTROL BITS PRESENT			PAYLOAD SIZE
HASH_CHECK	CIPHER_INIT	PAYLOAD_KEY	
0	0	1	4 words / 16 bytes
0	1	0	4 words / 16 bytes
0	1	1	8 words / 32 bytes
1	0	0	SHA-1: 5 words / 20 bytes SHA-2: 8 words / 32 bytes
1	0	1	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes
1	1	0	SHA-1: 9 words / 36 bytes SHA-2: 12 words / 48 bytes
1	1	1	SHA-1: 13 words / 52 bytes SHA-2 16 words / 64 bytes

For hashing operations, the DCP module writes the final hash value back to the start of the payload area for descriptors with the HASH_TERM bit set in the control packet. It is important that software allocates the required payload space, even though it is not required to set up the payload for control purposes.

13.2.7 Programming DCP Functions

13.2.7.1 Basic Memory Copy Programming Example

To perform a basic memcopy operation, only a single descriptor is required. The DCP simply copies data from the source to the destination buffer. This process is illustrated in the following figure.

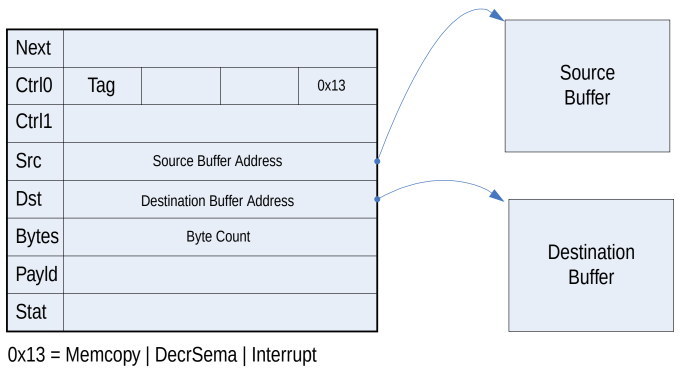


Figure 13-10. Basic Memory Copy Operation

```
<code>

typedef struct _dcp_descriptor
{
    u32              *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32              *src,
                     *dst,
                      buf_size,
                     *payload,
                      stat;
}  DCP_DESCRIPTOR;

  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer, *dstbuffer;

  // set up control packet
  dcp1.next = 0;                     // single packet in chain
  dcp1.ctrl0.U = 0;                  // clear ctrl0 field
  dcp1.ctrl0.B.ENABLE_MEMCOPY = 1; // enable memcopy
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;      // interrupt
  dcp1.ctrl1.U = 0;                  // clear ctrl1
  dcp1.src = srcbuffer;              // source buffer
  dcp1.dst = dstbuffer;              // destination buffer
  dcp1.buf_size = 512;               // 512 bytes
  dcp1.payload = NULL;               // not required
  dcp1.status = 0;                   // clear status

  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);         // increment semaphore by 1

  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);

</code>
```

13.2.7.2 Basic Hash Operation Programming Example

To perform a basic hash operation, only a single descriptor is required. The DCP simply reads data from the source buffer and computes the hash value on the contents. This process is illustrated below.

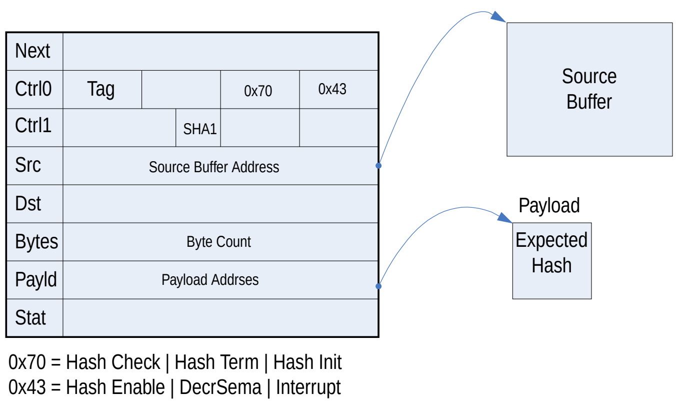


Figure 13-11. Basic Hash Operation

```
<code>

typedef struct _dcp_descriptor
{
    u32                *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32                *src,
                       *dst,
                        buf_size,
                       *payload,
                        stat;
}  DCP_DESCRIPTOR;

  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 payload[5];

  // set up expected hash check value
  payload[0]=0x01234567;
  payload[1]=0x89ABCDEF;
  payload[2]=0x00112233;
  payload[3]=0x44556677;
  payload[4]=0x8899aabb;

  // set up control packet
  dcp1.next = 0;                     // single packet in chain
  dcp1.ctrl0.U = 0;                  // clear ctrl0 field
  dcp1.ctrl0.B.HASH_CHECK = 1;       // check hash when complete
  dcp1.ctrl0.B.HASH_INIT = 1;        // initialize hash with this block
  dcp1.ctrl0.B.HASH_TERM = 1;        // terminate hash with this block
  dcp1.ctrl0.B.ENABLE_HASH = 1;      // enable hash
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1;   // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;        // interrupt
  dcp1.ctrl1.U = 0;                  // clear ctrl1
  dcp1.src = srcbuffer;              // source buffer
  dcp1.dst = 0;                      // no destination buffer
  dcp1.buf_size = 512;               // 512 bytes
  dcp1.payload = payload;            // holds expected hash value
  dcp1.status = 0;                   // clear status

  // Enable channel 0
  HW_DCP_CHnCMDPTR_WR(0, dcp1);      // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);           // increment semaphore by 1

  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );
  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);

</code>
```

13.2.7.3 Basic Cipher Operation Programming Example

To perform a basic cipher operation, only a single descriptor is required. The DCP reads data from the source buffer, encrypts it, and writes it to the destination buffer. For this example, the key is provided in the payload and the algorithm uses the AES CBC mode of operation. This requires a payload with both the key and CBC initialization value. This process is illustrated below.

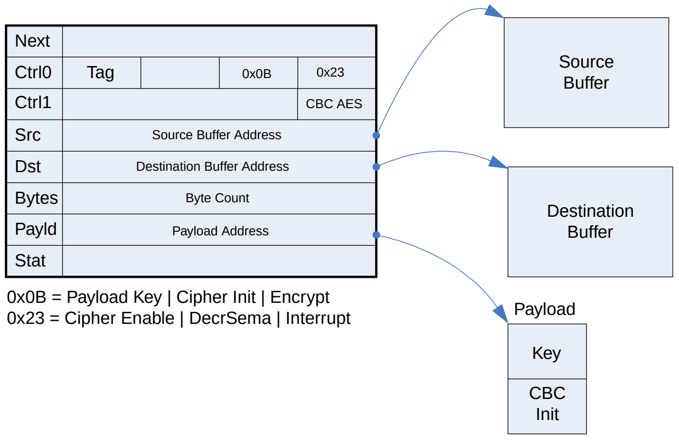


Figure 13-12. Basic Cipher Operation

```
<code>

typedef struct _dcp_descriptor
{
    u32               *next;
    hw_dcp_packet1_t  ctrl0;
    hw_dcp_packet2_t  ctrl1;
    u32               *src,
                      *dst,
                       buf_size,
                      *payload,
                       stat;
}  DCP_DESCRIPTOR;

  DCP_DESCRIPTOR dcp1;
  u32 *srcbuffer;
  u32 *dstbuffer;
  u32 payload[8];

  // set up key/CBC init in the payload
  payload[0]=0x01234567;           // key
  payload[1]=0x89ABCDEF;
  payload[2]=0xfedcba98;
  payload[3]=0x76543210;
  payload[4]=0x00112233;           // CBC initialization
  payload[5]=0x44556677;
  payload[6]=0x8899aabb;
  payload[7]=0xccddeeff;

  // set up control packet
  dcp1.next = 0;                   // single packet in chain
  dcp1.ctrl0.U = 0;                // clear ctrl0 field
  dcp1.ctrl0.B.PAYLOAD_KEY = 1;    // key is located in payload
  dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
  dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
  dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
  dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
  dcp1.ctrl0.B.INTERRUPT = 1;      // interrupt
  dcp1.ctrl1.U = 0;                // clear ctrl1
  dcp1.ctrl1.B.CIPHER_MODE = 1;    // select CBC mode of operation
  dcp1.src = srcbuffer;            // source buffer
  dcp1.dst = dstbuffer;            // destination buffer
  dcp1.buf_size = 512;             // 512 bytes
  dcp1.payload = payload;          // holds key/CBC init
  dcp1.status = 0;                 // clear status

  // Enable channel 0

  HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
  HW_DCP_CHnSEMA_WR(0, 1);         // increment semaphore by 1

  // now wait for interrupt or poll
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);

</code>
```

13.2.7.4 Multi-Buffer Scatter/Gather Cipher and Hash Operation Programming Example

For this example, three separate buffers are encrypted and hashed with the results being directed to a unified buffer (gather operation). Three descriptors are used for the operation because there are three separate source buffer pointers. The DCP reads data from the source buffer and computes a hash on the unencrypted data. It then encrypts the data and writes it to the destination buffer. For this example, the key is located in the key RAM, and the algorithm uses the AES CBC mode of operation with a SHA-1 hash. The payload for the first operation contains the CBC initialization value, and the payload for the last packet contains the expected hash value. The middle packet requires no payload. This process is illustrated below.

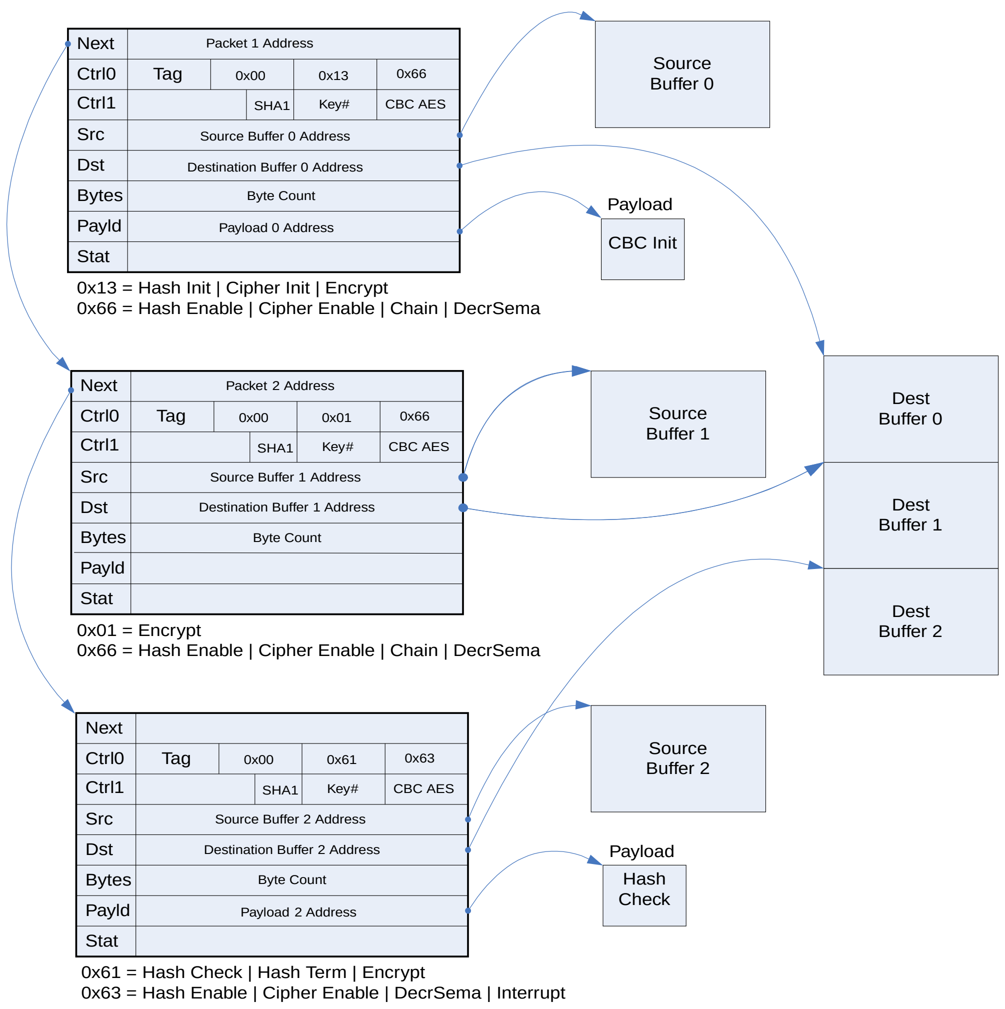


Figure 13-13. Multi-Buffer Scatter/Gather Cipher and Hash Operation

```
<code>

typedef struct _dcp_descriptor
{
    u32                 *next;
    hw_dcp_packet1_t     ctrl0;
    hw_dcp_packet2_t     ctrl1;
    u32                 *src,
                        *dst,
                         buf_size,
                        *payload,
                         stat;
}  DCP_DESCRIPTOR;

  DCP_DESCRIPTOR dcp1, dcp2, dcp3;
  u32 *srcbuffer0, *srcbuffer2, *srcbuffer3;
  u32 *dstbuffer;
  u32 payload0[4], payload2[5];

  // set up CBC init in the payload
  payload0[0]=0x01234567;           // key
  payload0[1]=0x89ABCDEF;
  payload0[2]=0xfedcba98;
  payload0[3]=0x76543210;

  payload2[0]=0x00112233;           // CBC initialization
```

```
payload2[1]=0x44556677;
payload2[2]=0x8899aabb;
payload2[3]=0xccddeeff;
payload2[3]=0xaabbccdd;

// set up control packet
dcp1.next = dcp2;                // point to packet 2
dcp1.ctrl0.U = 0;                // clear ctrl0 field
dcp1.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp1.ctrl0.B.CIPHER_INIT = 1;    // init CBC for this block (from payload)
dcp1.ctrl0.B.HASH_INIT = 1;      // init hash this block
dcp1.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp1.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp1.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp1.ctrl0.B.CHAIN = 1;          // chain to next packet
dcp1.ctrl1.U = 0;                // clear ctrl1
dcp1.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp1.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1;    // select SHA1 hash
dcp1.ctrl1.B.KEY_SELECT  = 2;    // select key #2
dcp1.src = srcbuffer0;            // source buffer
dcp1.dst = dstbuffer;            // destination buffer
dcp1.buf_size = 512;             // 512 bytes
dcp1.payload = payload0;          // holds key/CBC init
dcp1.status = 0;                 // clear status

// set up control packet
dcp2.next = dcp3;                // point to packet 2
dcp2.ctrl0.U = 0;                // clear ctrl0 field
dcp2.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp2.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp2.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp2.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp2.ctrl0.B.CHAIN = 1;          // chain to next packet
dcp2.ctrl1.U = 0;                // clear ctrl1
dcp2.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp2.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1;    // select SHA1 hash
dcp2.ctrl1.B.KEY_SELECT  = 2;    // select key #2
dcp2.src = srcbuffer1;           // source buffer
dcp2.dst = dstbuffer+512;        // destination buffer
dcp2.buf_size = 512;             // 512 bytes
dcp2.payload = NULL;             // no payload required
dcp2.status = 0;                 // clear status

// set up control packet
dcp3.next = dcp3;                // point to packet 2
dcp3.ctrl0.U = 0;                // clear ctrl0 field
dcp3.ctrl0.B.HASH_TERM      = 1; // terminate hash block
dcp3.ctrl0.B.HASH_CHECK     = 1; // check hash against payload value
dcp3.ctrl0.B.CIPHER_ENCRYPT = 1; // encryption operation
dcp3.ctrl0.B.ENABLE_CIPHER = 1;  // enable cipher
dcp3.ctrl0.B.ENABLE_HASH = 1;    // enable cipher
dcp3.ctrl0.B.DECR_SEMAPHORE = 1; // decrement semaphore
dcp3.ctrl0.B.INTERRUPT = 1;      // interrupt on completion
dcp3.ctrl1.U = 0;                // clear ctrl1
dcp3.ctrl1.B.CIPHER_MODE = BV_DCP_PACKET2_CIPHER_MODE_CBC; //select CBC mode of operation
dcp3.ctrl1.B.HASH_SELECT = BV_DCP_PACKET2_HASH_SELECT_SHA1;    // select SHA1 hash
dcp3.ctrl1.B.KEY_SELECT  = 2;    // select key #2
dcp3.src = srcbuffer1;           // source buffer
dcp3.dst = dstbuffer+1024;       // destination buffer
dcp3.buf_size = 512;             // 512 bytes
dcp3.payload = payload2;         // payload is hash check value
dcp3.status = 0;                 // clear status

// Enable channel 0
HW_DCP_CHnCMDPTR_WR(0, dcp1);    // write packet address to pointer register
HW_DCP_CHnSEMA_WR(0, 3);         // increment semaphore by 3 (for 3 packets)

// now wait for interrupt or poll
```

```
  // polling code
  while ( (HW_DCP_STAT_RD() & 0x01) == 0x00 );

  // now check/clear channel status
  if ( (HW_DCP_CHnSTAT_RD(0) & 0xFF) != 0 ) {
    // an error occurred
    HW_DCP_CHnSTAT_CLR(0, 0xff);
  }
  // clear interrupt register
  HW_DCP_STAT_CLR(1);

</code>
```

13.3 Programmable Registers

DCP Hardware Register Format Summary

HW_DCP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_8000	DCP Control Register 0 (HW_DCP_CTRL)	32	R/W	F080_0000h	13.3.1/1025
8002_8010	DCP Status Register (HW_DCP_STAT)	32	R/W	1000_0000h	13.3.2/1027
8002_8020	DCP Channel Control Register (HW_DCP_CHANNELCTRL)	32	R/W	0000_0000h	13.3.3/1028
8002_8030	DCP Capability 0 Register (HW_DCP_CAPABILITY0)	32	R/W	0000_0404h	13.3.4/1030
8002_8040	DCP Capability 1 Register (HW_DCP_CAPABILITY1)	32	R/W	0007_0001h	13.3.5/1031
8002_8050	DCP Context Buffer Pointer (HW_DCP_CONTEXT)	32	R/W	0000_0000h	13.3.6/1031
8002_8060	DCP Key Index (HW_DCP_KEY)	32	R/W	0000_0000h	13.3.7/1032
8002_8070	DCP Key Data (HW_DCP_KEYDATA)	32	R/W	0000_0000h	13.3.8/1033
8002_8080	DCP Work Packet 0 Status Register (HW_DCP_PACKET0)	32	R/W	0000_0000h	13.3.9/1034
8002_8090	DCP Work Packet 1 Status Register (HW_DCP_PACKET1)	32	R/W	0000_0000h	13.3.10/1034
8002_80A0	DCP Work Packet 2 Status Register (HW_DCP_PACKET2)	32	R/W	0000_0000h	13.3.11/1036
8002_80B0	DCP Work Packet 3 Status Register (HW_DCP_PACKET3)	32	R/W	0000_0000h	13.3.12/1038
8002_80C0	DCP Work Packet 4 Status Register (HW_DCP_PACKET4)	32	R/W	0000_0000h	13.3.13/1038
8002_80D0	DCP Work Packet 5 Status Register (HW_DCP_PACKET5)	32	R/W	0000_0000h	13.3.14/1039
8002_80E0	DCP Work Packet 6 Status Register (HW_DCP_PACKET6)	32	R/W	0000_0000h	13.3.15/1039
8002_8100	DCP Channel 0 Command Pointer Address Register (HW_DCP_CH0CMDPTR)	32	R/W	0000_0000h	13.3.16/1040
8002_8110	DCP Channel 0 Semaphore Register (HW_DCP_CH0SEMA)	32	R/W	0000_0000h	13.3.17/1040
8002_8120	DCP Channel 0 Status Register (HW_DCP_CH0STAT)	32	R/W	0000_0000h	13.3.18/1041
8002_8130	DCP Channel 0 Options Register (HW_DCP_CH0OPTS)	32	R/W	0000_0000h	13.3.19/1043
8002_8140	DCP Channel 1 Command Pointer Address Register (HW_DCP_CH1CMDPTR)	32	R/W	0000_0000h	13.3.20/1044
8002_8150	DCP Channel 1 Semaphore Register (HW_DCP_CH1SEMA)	32	R/W	0000_0000h	13.3.21/1045

HW_DCP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_8160	DCP Channel 1 Status Register (HW_DCP_CH1STAT)	32	R/W	0000_0000h	13.3.22/1046
8002_8170	DCP Channel 1 Options Register (HW_DCP_CH1OPTS)	32	R/W	0000_0000h	13.3.23/1048
8002_8180	DCP Channel 2 Command Pointer Address Register (HW_DCP_CH2CMDPTR)	32	R/W	0000_0000h	13.3.24/1048
8002_8190	DCP Channel 2 Semaphore Register (HW_DCP_CH2SEMA)	32	R/W	0000_0000h	13.3.25/1049
8002_81A0	DCP Channel 2 Status Register (HW_DCP_CH2STAT)	32	R/W	0000_0000h	13.3.26/1050
8002_81B0	DCP Channel 2 Options Register (HW_DCP_CH2OPTS)	32	R/W	0000_0000h	13.3.27/1052
8002_81C0	DCP Channel 3 Command Pointer Address Register (HW_DCP_CH3CMDPTR)	32	R/W	0000_0000h	13.3.28/1053
8002_81D0	DCP Channel 3 Semaphore Register (HW_DCP_CH3SEMA)	32	R/W	0000_0000h	13.3.29/1054
8002_81E0	DCP Channel 3 Status Register (HW_DCP_CH3STAT)	32	R/W	0000_0000h	13.3.30/1055
8002_81F0	DCP Channel 3 Options Register (HW_DCP_CH3OPTS)	32	R/W	0000_0000h	13.3.31/1056
8002_8400	DCP Debug Select Register (HW_DCP_DBGSELECT)	32	R/W	0000_0000h	13.3.32/1057
8002_8410	DCP Debug Data Register (HW_DCP_DBGDATA)	32	R/W	0000_0000h	13.3.33/1058
8002_8420	DCP Page Table Register (HW_DCP_PAGETABLE)	32	R/W	0000_0000h	13.3.34/1058
8002_8430	DCP Version Register (HW_DCP_VERSION)	32	R/W	0201_0000h	13.3.35/1059

13.3.1 DCP Control Register 0 (HW_DCP_CTRL)

The CTRL register contains controls for the DCP module.

HW_DCP_CTRL: 0x000

HW_DCP_CTRL_SET: 0x004

HW_DCP_CTRL_CLR: 0x008

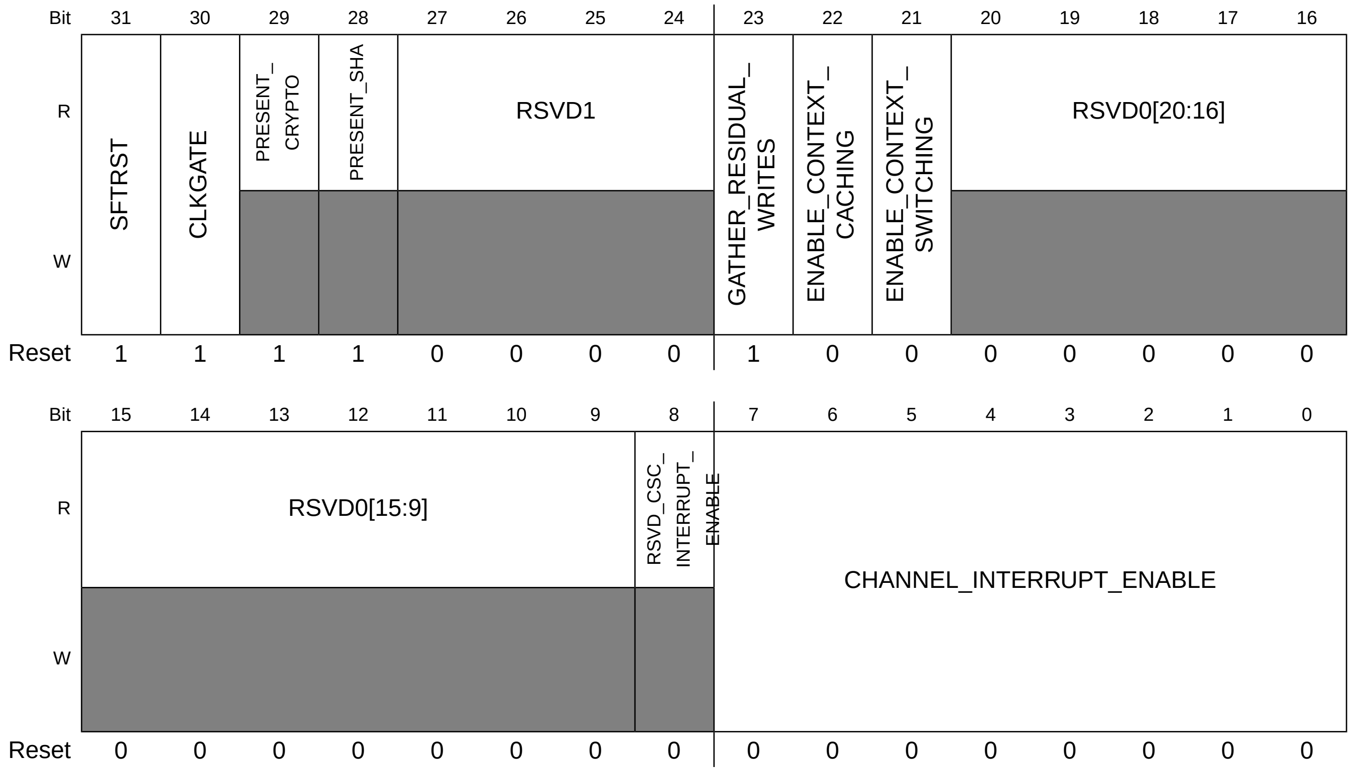
HW_DCP_CTRL_TOG: 0x00C

The Control register contains the primary controls for the DCP block. The present bits indicate which of the sub-features of the block are present in the hardware. The context control bits control how the DCP utilizes it's context buffer and the gather residual writes bit controls how the master handles writing misaligned data to the bus. Each channel and the color-space converter contains an independent interrupt enable.

EXAMPLE

```
HW_DCP_CTRL_SET(BM_DCP_CTRL_SFTRST);
HW_DCP_CTRL_CLR(BM_DCP_CTRL_SFTRST | BM_DCP_CTRL_CLKGATE);
```

Address: HW_DCP_CTRL – 8002_8000h base + 0h offset = 8002_8000h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
SFTRST
CLKGATE
PRESENT_CRYPTO
PRESENT_SHA
RSVD1
GATHER_RESIDUAL_WRITES
ENABLE_CONTEXT_CACHING
ENABLE_CONTEXT_SWITCHING
RSVD0[20:16]
Reset 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSVD0[15:9]
RSVD_CSC_INTERRUPT_ENABLE
CHANNEL_INTERRUPT_ENABLE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_DCP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal DCP operation. Set this bit to one (default) to disable clocking with the DCP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the DCP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 PRESENT_ CRYPTO	Indicates whether the crypto (Cipher/Hash) functions are present. 0x1 **Present** — 0x0 **Absent** —
28 PRESENT_SHA	Indicates whether the SHA1/SHA2 functions are present. 0x1 **Present** — 0x0 **Absent** —
27–24 RSVD1	Reserved, always set to zero.
23 GATHER_ RESIDUAL_ WRITES	Software should set this bit to enable ragged writes to unaligned buffers to be gathered between multiple write operations. This improves performance by removing several byte operations between write bursts. Trailing byte writes are held in a residual write data buffer and combined with a subsequent write to the buffer to form a word write.
22 ENABLE_ CONTEXT_ CACHING	Software should set this bit to enable caching of contexts between operations. If only a single channel is used for encryption/hashing, enabling caching causes the context to not be reloaded if the channel was the last to be used.

HW_DCP_CTRL field descriptions (continued)

Field	Description
21 ENABLE_ CONTEXT_ SWITCHING	Enable automatic context switching for the channels. Software should set this bit if more than one channel is doing hashing or cipher operations that require context to be saved (for instance, when CBC mode is enabled). By disabling context switching, software can save the 208 bytes used for the context buffer.
20–9 RSVD0	Reserved, always set to zero.
8 RSVD_CSC_ INTERRUPT_ ENABLE	Reserved, always set to zero.
7–0 CHANNEL_ INTERRUPT_ ENABLE	Per-channel interrupt enable bit. When set, the channel's interrupt will get routed to the interrupt controller. Channel 0 is routed to the dcp_vmi_irq signal and the other channels are combined (along with the CRC interrupt) into the dcp_irq signal. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

13.3.2 DCP Status Register (HW_DCP_STAT)

The DCP Interrupt Status register provides channel interrupt status information.

HW_DCP_STAT: 0x010

HW_DCP_STAT_SET: 0x014

HW_DCP_STAT_CLR: 0x018

HW_DCP_STAT_TOG: 0x01C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

Address: HW_DCP_STAT – 8002_8000h base + 10h offset = 8002_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2			OTP_ KEY_ READY	CUR_CHANNEL				READY_CHANNELS							
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1							RSVD_ IRQ	RSVD0				IRQ			
W	▨							▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_STAT field descriptions

Field	Description
31–29 RSVD2	Reserved, always set to zero.
28 OTP_KEY_ READY	When set, indicates that the OTP key has been shifted from the fuse block and is ready for use.
27–24 CUR_CHANNEL	Current (active) channel (encoded). 0x0 **None** — 0x1 **CH0** — 0x2 **CH1** — 0x3 **CH2** — 0x4 **CH3** —
23–16 READY_ CHANNELS	Indicates which channels are ready to proceed with a transfer (active channel also included). Each bit is a one-hot indicating the request status for the associated channel. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
15–9 RSVD1	Reserved, always set to zero.
8 RSVD_IRQ	Reserved, always set to zero.
7–4 RSVD0	Reserved, always set to zero.
3–0 IRQ	Indicates which channels have pending interrupt requests. Channel 0's interrupt is routed through the dcp_vmi_irq and the other interrupt bits are routed through the dcp_irq.

13.3.3 DCP Channel Control Register (HW_DCP_CHANNELCTRL)

The DCP Channel Control register provides controls for channel arbitration and channel enables.

HW_DCP_CHANNELCTRL: 0x020

HW_DCP_CHANNELCTRL_SET: 0x024

HW_DCP_CHANNELCTRL_CLR: 0x028

HW_DCP_CHANNELCTRL_TOG: 0x02C

This register provides status feedback indicating the channel currently performing an operation and which channels have pending operations.

EXAMPLE

```
        BW_DCP_CHANNELCTRL_ENABLE_CHANNEL(BV_DCP_CHANNELCTRL_ENABLE_CHANNEL__CH0);  // enable
 channel 0
```

Address: HW_DCP_CHANNELCTRL – 8002_8000h base + 20h offset = 8002_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD															CH0_IRQ_MERGED
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	HIGH_PRIORITY_CHANNEL								ENABLE_CHANNEL							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CHANNELCTRL field descriptions

Field	Description
31–17 RSVD	Reserved, always set to zero.
16 CH0_IRQ_ MERGED	Indicates that the interrupt for channel 0 should be merged with the other interrupts on the shared dcp_irq interrupt. When set to 0, channel 0's interrupt will be routed to the separate dcp_vmi_irq. When set to 1, the interrupt will be routed to the shared DCP interrupt.
15–8 HIGH_ PRIORITY_ CHANNEL	Setting a bit in this field causes the corresponding channel to have high-priority arbitration. High priority channels will be arbitrated round-robin and will take precedence over other channels that are not marked as high priority. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —
7–0 ENABLE_ CHANNEL	Setting a bit in this field will enabled the DMA channel associated with it. This field is a direct input to the DMA channel arbiter. When not enabled, the channel is denied access to the central DMA resources. 0x01 **CH0** — 0x02 **CH1** — 0x04 **CH2** — 0x08 **CH3** —

13.3.4 DCP Capability 0 Register (HW_DCP_CAPABILITY0)

This register contains additional information about the DCP module implementation parameters.

This register provides capability information for the DCP block. It indicates the number of channels implemented as well as the number of key storage locations available for software use.

Address: HW_DCP_CAPABILITY0 – 8002_8000h base + 30h offset = 8002_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DISABLE_ DECRYPT	ENABLE_ TZONE	DISABLE_ UNIQUE_KEY	RSVD[28:16]												
W				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:12]				NUM_CHANNELS				NUM_KEYS							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

HW_DCP_CAPABILITY0 field descriptions

Field	Description
31 DISABLE_ DECRYPT	Write to a 1 to disable decryption. This bit can only be written by secure software and the value can only be cleared by a reset.
30 ENABLE_TZONE	Write to a 1 to enable trustzone support. Channel operations initiated by secure operations will be protected from non-secure operations and the secure channel interrupts will be routed to the secure DCP interrupt. This bit can only be written by secure software and the value can only be cleared by a reset.
29 DISABLE_ UNIQUE_KEY	Write to a 1 to disable the per-device unique key. The device-specific hardware key may be selected by using a value of 0xFE in the key select field.
28–12 RSVD	Reserved, always set to zero.
11–8 NUM_ CHANNELS	Encoded value indicating the number of channels implemented in the design.
7–0 NUM_KEYS	Encoded value indicating the number of key storage locations implemented in the design.

13.3.5 DCP Capability 1 Register (HW_DCP_CAPABILITY1)

This register contains information about the algorithms available on the implementation.

This register provides capability information for the DCP block. It contains two fields indicating which encryption and hashing algorithms are present in the design. Each bit set indicates that support for the associated function is present.

Address: HW_DCP_CAPABILITY1 – 8002_8000h base + 40h offset = 8002_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HASH_ALGORITHMS																CIPHER_ALGORITHMS															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_DCP_CAPABILITY1 field descriptions

Field	Description
31–16 HASH_ ALGORITHMS	One-hot field indicating which hashing features are implemented in HW. 0x0001 **SHA1** — 0x0002 **CRC32** — 0x0004 **SHA256** —
15–0 CIPHER_ ALGORITHMS	One-hot field indicating which cipher algorithms are available. 0x0001 **AES128** —

13.3.6 DCP Context Buffer Pointer (HW_DCP_CONTEXT)

This register contains a pointer to the memory region to be used for DCP context swap operations.

This register contains a pointer to the start of the context pointer memory in on-chip SRAM or off-chip SDRAM. This buffer will be used to store state information when the DCP module changes from one channel to another.

Address: HW_DCP_CONTEXT – 8002_8000h base + 50h offset = 8002_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CONTEXT field descriptions

Field	Description
31–0 ADDR	Context pointer address. Address should be located in system RAM and should be word-aligned for optimal performance.

13.3.7 DCP Key Index (HW_DCP_KEY)

This register contains a pointer to the key location to be written.

The DCP module maintains a set of write-only keys that may be used by software. To write a key, software must first write the desired key index/subword to this register and then write the key values to the key registers (below). After each write to the key data register, the SUBWORD field will increment to allow software to write the subsequent key to be written without having to rewrite the key index.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
HW_DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
 0
HW_DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
HW_DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
HW_DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
HW_DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: HW_DCP_KEY – 8002_8000h base + 60h offset = 8002_8060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:8]								RSVD_INDEX		INDEX		RSVD_ SUBWORD		SUBWORD	
W	▨								▨				▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_KEY field descriptions

Field	Description
31–8 RSVD	Reserved, always set to zero.

HW_DCP_KEY field descriptions (continued)

Field	Description
7–6 RSVD_INDEX	Reserved, always set to zero.
5–4 INDEX	Key index pointer. Valid indices are 0-[number_keys].
3–2 RSVD_ SUBWORD	Reserved, always set to zero.
1–0 SUBWORD	Key subword pointer. Valid indices are 0-3. After each write to the key data register, this field will increment.

13.3.8 DCP Key Data (HW_DCP_KEYDATA)

This register provides write access to the key/key subword specified by the Key Index Register.

Writing this location updates the selected subword for the key located at the index specified by the Key Index Register. A write also triggers the SUBWORD field of the KEY register to increment to the next higher word in the key.

EXAMPLE

```
// write key 0 to 0x00112233_44556677_8899aabb_ccddeeff
HW_DCP_KEY_WR(BF_DCP_KEY_INDEX(0) | BF_DCP_KEY_SUBWORD(0));  // set key index to key 0, subword
 0
HW_DCP_KEYDATA_WR(0xccddeeff);  // write key values (subword 0)
HW_DCP_KEYDATA_WR(0x8899aabb);  // write key values (subword 1)
HW_DCP_KEYDATA_WR(0x44556677);  // write key values (subword 2)
HW_DCP_KEYDATA_WR(0x00112233);  // write key values (subword 3)
```

Address: HW_DCP_KEYDATA – 8002_8000h base + 70h offset = 8002_8070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_KEYDATA field descriptions

Field	Description
31–0 DATA	Word 0 data for key. This is the least-significant word.

13.3.9 DCP Work Packet 0 Status Register (HW_DCP_PACKET0)

This register displays the values for the current work packet offset 0x00 (Next Command) field.

The Work Packet Status Registers show the contents of the currently executing packet. When the channels are inactive, the packet status register return 0. The register bits are fully documented here to document the packet structure in memory.

Address: HW_DCP_PACKET0 – 8002_8000h base + 80h offset = 8002_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET0 field descriptions

Field	Description
31–0 ADDR	Next Pointer Register,

13.3.10 DCP Work Packet 1 Status Register (HW_DCP_PACKET1)

This register displays the values for the current work packet offset 0x04 (control) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: HW_DCP_PACKET1 – 8002_8000h base + 90h offset = 8002_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAG								OUTPUT_ WORDSWAP	OUTPUT_ BYTESWAP	INPUT_ WORDSWAP	INPUT_ BYTESWAP	KEY_ WORDSWAP	KEY_ BYTESWAP	TEST_SEMA_ IRQ	CONSTANT_FILL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HASH_ OUTPUT	CHECK_HASH	HASH_TERM	HASH_INIT	PAYLOAD_KEY	OTP_ KEY	CIPHER_INIT	CIPHER_ ENCRYPT	ENABLE_BLIT	ENABLE_HASH	ENABLE_ CIPHER	ENABLE_ MEMCOPY	CHAIN_ CONTIGUOUS	CHAIN	DECR_ SEMAPHORE	INTERRUPT
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET1 field descriptions

Field	Description
31–24 TAG	Packet Tag
23 OUTPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap output data (big-endian data).
22 OUTPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap output data (big-endian data).
21 INPUT_ WORDSWAP	Reflects whether the DCP engine will wordswap input data (big-endian data).
20 INPUT_ BYTESWAP	Reflects whether the DCP engine will byteswap input data (big-endian data).
19 KEY_ WORDSWAP	Reflects whether the DCP engine will swap key words (big-endian key).
18 KEY_ BYTESWAP	Reflects whether the DCP engine will swap key bytes (big-endian key).
17 TEST_SEMA_ IRQ	This bit is used to test the channel semaphore transition to 0. FOR TEST USE ONLY!
16 CONSTANT_FILL	When this bit is set (MEMCOPY and BLIT modes only), the DCP will simply fill the destination buffer with the value found in the Source Address field.
15 HASH_OUTPUT	When hashing is enabled, this bit controls whether the input or output data is hashed. 0x00 **INPUT** — 0x01 **OUTPUT** —
14 CHECK_HASH	Reflects whether the calculated hash value should be compared against the hash provided in the payload.

HW_DCP_PACKET1 field descriptions (continued)

Field	Description
13 HASH_TERM	Reflects whether the current hashing block is the final block in the hashing operation, so the hash padding should be applied by hardware.
12 HASH_INIT	Reflects whether the current hashing block is the initial block in the hashing operation, so the hash registers should be initialized before the operation.
11 PAYLOAD_KEY	When set, indicates the payload contains the key. This bit takes precedence over the OTP_KEY control
10 OTP_KEY	Reflects whether a hardware-based key should be used. The KEY_SELECT field from the Control1 field is used to select from multiple hardware keys. The PAYLOAD_KEY bit takes precedence over the OTP_KEY bit.
9 CIPHER_INIT	Reflects whether the cipher block should load the initialization vector from the payload for this operation.
8 CIPHER_ ENCRYPT	When the cipher block is enabled, this bit indicates whether the operation is encryption or decryption. 0x01 **ENCRYPT** — 0x00 **DECRYPT** —
7 ENABLE_BLIT	Reflects whether the DCP should perform a blit operation. Source data is always continuous and the destination buffer is written in run/stride format. When set, the BUFFER_SIZE field is treated as two 16-bit values for the X-Y extents of the blit operation.
6 ENABLE_HASH	Reflects whether the selected hashing function should be enabled for this operation.
5 ENABLE_ CIPHER	Reflects whether the selected cipher function should be enabled for this operation.
4 ENABLE_ MEMCOPY	Reflects whether the selected hashing function should be enabled for this operation.
3 CHAIN_ CONTIGUOUS	Reflects whether the next packet's address is located following this packet's payload.
2 CHAIN	Reflects whether the next command pointer register should be loaded into the channel's current descriptor pointer.
1 DECR_ SEMAPHORE	Reflects whether the channel's semaphore should be decremented at the end of the current operation. When the semaphore reaches a value of zero, no more operations will be issued from the channel.
0 INTERRUPT	Reflects whether the channel should issue an interrupt upon completion of the packet.

13.3.11 DCP Work Packet 2 Status Register (HW_DCP_PACKET2)

This register displays the values for the current work packet offset 0x08 (Control1) field.

This register shows the contents of the Control0 register from the packet being processed.

Address: HW_DCP_PACKET2 – 8002_8000h base + A0h offset = 8002_80A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CIPHER_CFG								RSVD				HASH_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KEY_SELECT								CIPHER_MODE				CIPHER_SELECT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET2 field descriptions

Field	Description
31–24 CIPHER_CFG	Cipher configuration bits. Optional configuration bits required for ciphers
23–20 RSVD	Reserved, always set to zero.
19–16 HASH_SELECT	Hash Selection Field 0x00 **SHA1** — 0x01 **CRC32** — 0x02 **SHA256** —
15–8 KEY_SELECT	Key Selection Field. The value here reflects the key index for the cipher operation. Values 0-3 refer to the software keys that can be written to the key RAM. The OTP key or the unique device-specific key may also be selected with a value of 0xFF (OTP key) or 0xFE (unique key). 0x00 **KEY0** — 0x01 **KEY1** — 0x02 **KEY2** — 0x03 **KEY3** — 0xFE **UNIQUE_KEY** — 0xFF **OTP_KEY** —
7–4 CIPHER_MODE	Cipher Mode Selection Field. Reflects the mode of operation for cipher operations. 0x00 **ECB** — 0x01 **CBC** —
3–0 CIPHER_ SELECT	Cipher Selection Field 0x00 **AES128** —

13.3.12 DCP Work Packet 3 Status Register (HW_DCP_PACKET3)

This register displays the values for the current work packet offset 0x0C (Source Address) field.

This register shows the contents of the Source Address register from the packet being processed. When the CONSTANT_FILL bit in the Control 0 field is set, this field contains the data written to the destination buffer.

Address: HW_DCP_PACKET3 – 8002_8000h base + B0h offset = 8002_80B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET3 field descriptions

Field	Description
31–0 ADDR	Source Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

13.3.13 DCP Work Packet 4 Status Register (HW_DCP_PACKET4)

This register displays the values for the current work packet offset 0x10 (Destination Address) field.

This register shows the contents of the Destination Address register from the packet being processed.

Address: HW_DCP_PACKET4 – 8002_8000h base + C0h offset = 8002_80C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET4 field descriptions

Field	Description
31–0 ADDR	Destination Buffer Address Pointer. This value is the working value and will update as the operation proceeds.

13.3.14 DCP Work Packet 5 Status Register (HW_DCP_PACKET5)

This register displays the values for the current work packet offset 0x14 (Buffer Size) field.

This register shows the contents of the bytecount register from the packet being processed. The field can be considered either a byte count or a buffer size. The logic treats this as a decrmenting count of bytes from the buffer size programmed into the field. As the transaction proceeds, the logic will decrement the bytecount as data is written to the destination buffer. For blit operations, the top 16-bits of this field represents the number of lines (y size) in the blit and the lower 16-bits represent the number of bytes in a line (x size).

Address: HW_DCP_PACKET5 – 8002_8000h base + D0h offset = 8002_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET5 field descriptions

Field	Description
31–0 COUNT	Byte Count register. This value is the working value and will update as the operation proceeds.

13.3.15 DCP Work Packet 6 Status Register (HW_DCP_PACKET6)

This register displays the values for the current work packet offset 0x1C (Payload Pointer) field.

This register shows the contents of the payload pointer fieldr from the packet being processed.

Address: HW_DCP_PACKET6 – 8002_8000h base + E0h offset = 8002_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PACKET6 field descriptions

Field	Description
31–0 ADDR	This regiser reflects the payload pointer for the current control packet.

13.3.16 DCP Channel 0 Command Pointer Address Register (HW_DCP_CH0CMDPTR)

The DCP channel 0 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 0 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
          HW_DCP_CHnCMDPTR_WR(0, v);   // Write channel 0 command pointer
          pCurptr = (hw_DCP_chncmdptr_t *) HW_DCP_CHnCMDPTR_RD(0);  // Read current command
pointer
```

Address: HW_DCP_CH0CMDPTR – 8002_8000h base + 100h offset = 8002_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH0CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 0.

13.3.17 DCP Channel 0 Semaphore Register (HW_DCP_CH0SEMA)

The DCP Channel 0 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. After processing each control packet, the DCP decrements the semaphore if it is non-zero. The channel will continue processing packets as long as the semaphore contains a non-zero value and the CHAIN or CHAIN_CONTIGOUS control bits in the Control0 field are set.

Address: HW_DCP_CH0SEMA – 8002_8000h base + 110h offset = 8002_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								VALUE								RSVD1								INCREMENT							
W	▨								▨								▨															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH0SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

13.3.18 DCP Channel 0 Status Register (HW_DCP_CH0STAT)

The DCP Channel 0 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH0STAT: 0x120

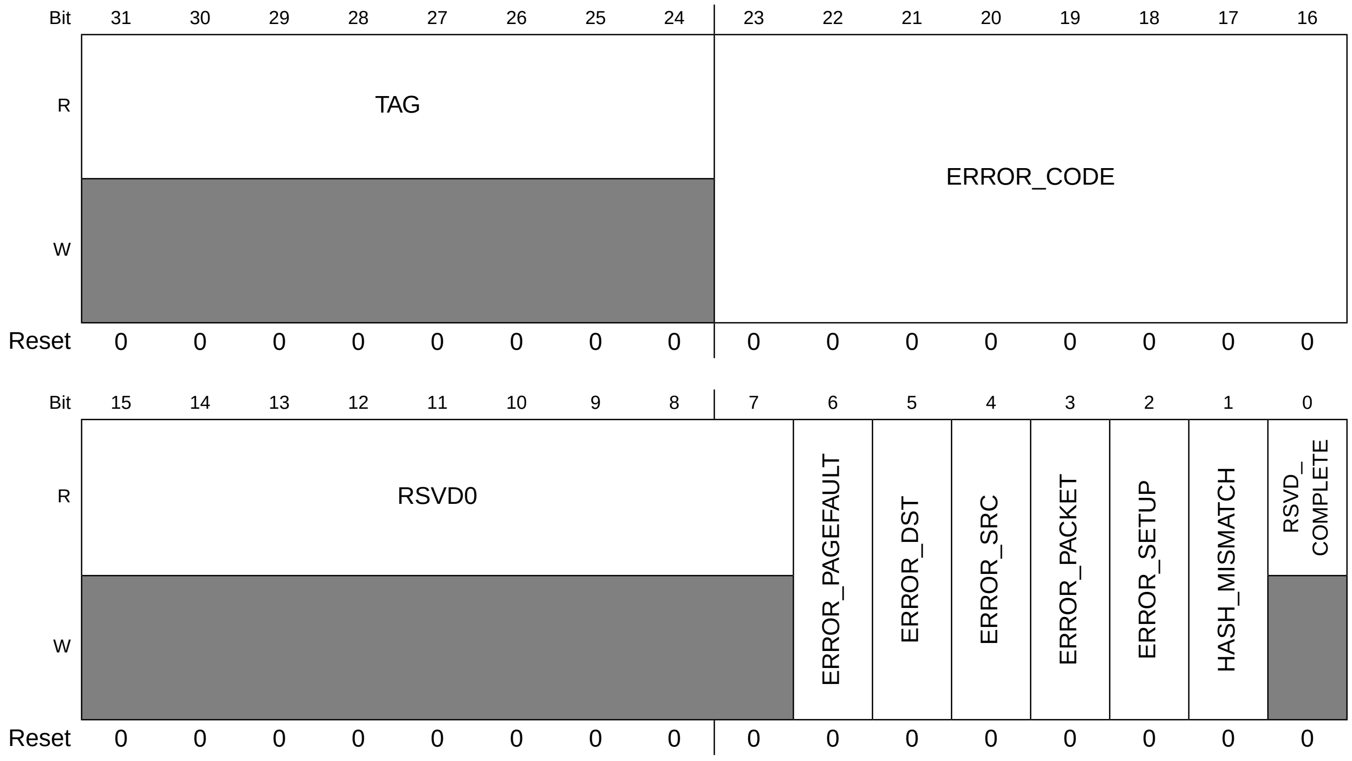
HW_DCP_CH0STAT_SET: 0x124

HW_DCP_CH0STAT_CLR: 0x128

HW_DCP_CH0STAT_TOG: 0x12C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: HW_DCP_CH0STAT – 8002_8000h base + 120h offset = 8002_8120h



HW_DCP_CH0STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)

HW_DCP_CH0STAT field descriptions (continued)

Field	Description
15–7 RSVD0	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a a bus error occurred when reading the packet or payload or when writing status back to the packet payload. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware has detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

13.3.19 DCP Channel 0 Options Register (HW_DCP_CH0OPTS)

The DCP Channel 0 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH0OPTS: 0x130

HW_DCP_CH0OPTS_SET: 0x134

HW_DCP_CH0OPTS_CLR: 0x138

HW_DCP_CH0OPTS_TOG: 0x13C

The options register can be used to control optional features of the channels.

Address: HW_DCP_CH0OPTS – 8002_8000h base + 130h offset = 8002_8130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH0OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate another operation for the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

13.3.20 DCP Channel 1 Command Pointer Address Register (HW_DCP_CH1CMDPTR)

The DCP channel 1 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 1 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
          HW_DCP_CHn_CMDPTR_WR(1, v);   // Write channel 1 command pointer
          pCurptr = (hw_DCP_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(1);  // Read current command
 pointer
```

Address: HW_DCP_CH1CMDPTR – 8002_8000h base + 140h offset = 8002_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH1CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 1.

13.3.21 DCP Channel 1 Semaphore Register (HW_DCP_CH1SEMA)

The DCP Channel 1 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: HW_DCP_CH1SEMA – 8002_8000h base + 150h offset = 8002_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								VALUE								RSVD1								INCREMENT							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH1SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.

HW_DCP_CH1SEMA field descriptions (continued)

Field	Description
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

13.3.22 DCP Channel 1 Status Register (HW_DCP_CH1STAT)

The DCP Channel 1 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH1STAT: 0x160

HW_DCP_CH1STAT_SET: 0x164

HW_DCP_CH1STAT_CLR: 0x168

HW_DCP_CH1STAT_TOG: 0x16C

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: HW_DCP_CH1STAT – 8002_8000h base + 160h offset = 8002_8160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAG								ERROR_CODE							
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0									ERROR_PAGEFAULT	ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	RSVD_ COMPLETE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH1STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 RSVD0	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurs when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

13.3.23 DCP Channel 1 Options Register (HW_DCP_CH1OPTS)

The DCP Channel 1 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH1OPTS: 0x170

HW_DCP_CH1OPTS_SET: 0x174

HW_DCP_CH1OPTS_CLR: 0x178

HW_DCP_CH1OPTS_TOG: 0x17C

The options register can be used to control optional features of the channels.

Address: HW_DCP_CH1OPTS – 8002_8000h base + 170h offset = 8002_8170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH1OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

13.3.24 DCP Channel 2 Command Pointer Address Register (HW_DCP_CH2CMDPTR)

The DCP channel 2 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 2 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
           HW_DCP_CHn_CMDPTR_WR(2, v);   // Write channel 2 command pointer
          pCurptr = (hw_dcp_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(2);  // Read current command
 pointer
```

Address: HW_DCP_CH2CMDPTR – 8002_8000h base + 180h offset = 8002_8180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH2CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 2.

13.3.25 DCP Channel 2 Semaphore Register (HW_DCP_CH2SEMA)

The DCP Channel 2 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: HW_DCP_CH2SEMA – 8002_8000h base + 190h offset = 8002_8190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								VALUE								RSVD1								INCREMENT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH2SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

13.3.26 DCP Channel 2 Status Register (HW_DCP_CH2STAT)

The DCP Channel 2 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH2STAT: 0x1A0

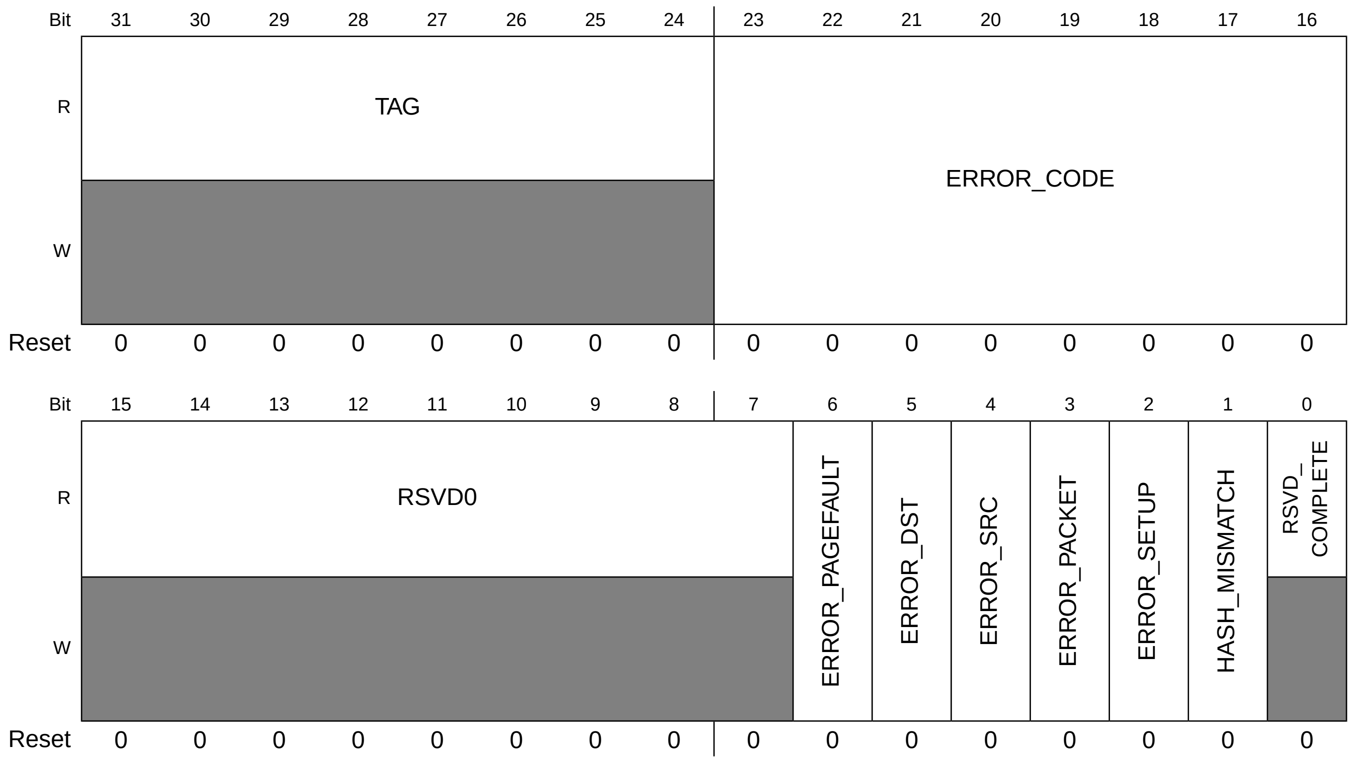
HW_DCP_CH2STAT_SET: 0x1A4

HW_DCP_CH2STAT_CLR: 0x1A8

HW_DCP_CH2STAT_TOG: 0x1AC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: HW_DCP_CH2STAT – 8002_8000h base + 1A0h offset = 8002_81A0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
TAG
ERROR_CODE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
RSVD0
ERROR_PAGEFAULT
ERROR_DST
ERROR_SRC
ERROR_PACKET
ERROR_SETUP
HASH_MISMATCH
RSVD_COMPLETE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_DCP_CH2STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 RSVD0	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.

HW_DCP_CH2STAT field descriptions (continued)

Field	Description
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

13.3.27 DCP Channel 2 Options Register (HW_DCP_CH2OPTS)

The DCP Channel 2 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH2OPTS: 0x1B0

HW_DCP_CH2OPTS_SET: 0x1B4

HW_DCP_CH2OPTS_CLR: 0x1B8

HW_DCP_CH2OPTS_TOG: 0x1BC

The options register can be used to control optional features of the channels.

Address: HW_DCP_CH2OPTS – 8002_8000h base + 1B0h offset = 8002_81B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH2OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

13.3.28 DCP Channel 3 Command Pointer Address Register (HW_DCP_CH3CMDPTR)

The DCP channel 3 current command address register points to the multiword descriptor that is to be executed (or currently being executed). The channel may be activated by writing the command pointer address to a valid descriptor in memory and then updating the semaphore to a non-zero value. After the engine completes processing of a descriptor, the next_ptr field from the descriptor is moved into this register to enable processing of the next descriptor. All channels with a non-zero semaphore value will arbitrate for access to the engine for the subsequent operation.

DCP Channel 3 is controlled by a variable sized command structure. This register points to the command structure to be executed.

EXAMPLE

```
         HW_DCP_CHn_CMDPTR_WR(3, v);   // Write channel 3 command pointer
         pCurptr = (hw_DCP_chn_cmdptr_t *) HW_DCP_CHn_CMDPTR_RD(3);  // Read current command
pointer
```

Address: HW_DCP_CH3CMDPTR – 8002_8000h base + 1C0h offset = 8002_81C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH3CMDPTR field descriptions

Field	Description
31–0 ADDR	Pointer to descriptor structure to be processed for channel 3.

13.3.29 DCP Channel 3 Semaphore Register (HW_DCP_CH3SEMA)

The DCP Channel 3 semaphore register is used to synchronize the CPU instruction stream and the DMA chain processing state. After a command chain has been generated in memory, software should write the address of the first command descriptor to the CMDPTR register and then write a non-zero value to the semaphore register to indicate that the channel is active. Each command packet has a chaining bit which indicates that another descriptor should be loaded into the channel upon completion of the current descriptor. If the chaining bit is not set, the next address will not be loaded into the CMDPTR register. Each packet also contains a decrement semaphore bit, which indicates that the counting semaphore should be decremented after the operation. A channel is considered active when the semaphore is a non-zero value. When programming a series operations, software must properly program the semaphore values in conjunction with the decrement_semaphore bits in the control packets to ensure that the proper number of descriptors are activated. A semaphore may be cleared by software by writing 0xFF to the HW_DCP_CHnSEMA_CLR register. The logic will also clear the semaphore if an error has occurred.

Each DCP channel has an 8 bit counting semaphore that is used to synchronize between the program stream and and the DCP chain processing. DCP processing continues until the engine attempts to decrement a semaphore that has already reached a value of zero. When the attempt is made, the DCP channel is stalled until software increments the semaphore count.

Address: HW_DCP_CH3SEMA – 8002_8000h base + 1D0h offset = 8002_81D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								VALUE								RSVD1								INCREMENT							
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH3SEMA field descriptions

Field	Description
31–24 RSVD2	Reserved, always set to zero.
23–16 VALUE	This read-only field shows the current (instantaneous) value of the semaphore counter.
15–8 RSVD1	Reserved, always set to zero.
7–0 INCREMENT	The value written to this field is added to the semaphore count in an atomic way such that simultaneous software adds and DCP hardware substracts happening on the same clock are protected. This bit field reads back a value of 0x00. Writing a value of 0x02 increments the semaphore count by two, unless the DCP channel decrements the count on the same clock, then the count is incremented by a net one. The semaphore may be cleared by writing 0xFF to the HW_DCP_CHnSEMA_CLR register.

13.3.30 DCP Channel 3 Status Register (HW_DCP_CH3STAT)

The DCP Channel 3 Interrupt Status register contains the interrupt status bit and the tag of the last completed operation from the command chain. If an error occurs during processing, the ERROR bit is set and an interrupt is generated.

HW_DCP_CH3STAT: 0x1E0

HW_DCP_CH3STAT_SET: 0x1E4

HW_DCP_CH3STAT_CLR: 0x1E8

HW_DCP_CH3STAT_TOG: 0x1EC

The interrupt status register is updated at the end of each work packet. If the interrupt bit is set in the command packet's command field, an interrupt will be generated once the packet has completed. In addition, the tag value from the command is stored in the TAG field so that software can identify which command structure was the last to complete. If an error occurs, the ERROR bit is set and processing of the command chain is halted.

Address: HW_DCP_CH3STAT – 8002_8000h base + 1E0h offset = 8002_81E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TAG								ERROR_CODE							
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0									ERROR_PAGEFAULT	ERROR_DST	ERROR_SRC	ERROR_PACKET	ERROR_SETUP	HASH_MISMATCH	RSVD_ COMPLETE
W	▨															▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH3STAT field descriptions

Field	Description
31–24 TAG	Indicates the tag from the last completed packet in the command structure
23–16 ERROR_CODE	Indicates additional error codes for some error conditions. 0x01 **NEXT_CHAIN_IS_0** — Error signalled because the next pointer is 0x00000000 0x02 **NO_CHAIN** — Error signalled because the semaphore is nonzero and neither chain bit is set 0x03 **CONTEXT_ERROR** — Error signalled because an error was reported reading/writing the context buffer 0x04 **PAYLOAD_ERROR** — Error signalled because an error was reported reading/writing the payload 0x05 **INVALID_MODE** — Error signalled because the control packet specifies an invalid mode select (for instance, blit + hash)
15–7 RSVD0	Reserved, always set to zero.
6 ERROR_ PAGEFAULT	This bit indicates a page fault occurred while converting a virtual address to a physical address.. When an error is detected, the channel's processing will stop until the error handled by software.
5 ERROR_DST	This bit indicates a bus error occurred when storing to the destination buffer. When an error is detected, the channel's processing will stop until the error handled by software.
4 ERROR_SRC	This bit indicates a bus error occurred when reading from the source buffer. When an error is detected, the channel's processing will stop until the error handled by software.
3 ERROR_ PACKET	This bit indicates that a bus error occurred when reading the packet or payload or when writing status back to the packet paylaod. When an error is detected, the channel's processing will stop until the error is handled by software.
2 ERROR_SETUP	This bit indicates that the hardware detected an invalid programming configuration such as a buffer length that is not a multiple of the natural data size for the operation. When an error is detected, the channel's processing will stop until the error is handled by software.
1 HASH_ MISMATCH	The bit indicates that a hashing check operation mismatched for control packets that enable the HASH_CHECK bit. When an error is detected, the channel's processing will stop until the error is handled by software.
0 RSVD_ COMPLETE	This bit will always read 0 in the status register, but will be set to 1 in the packet status field after processing of the packet has completed. This was done so that software can verify that each packet completed properly in a chain of commands for cases when an interrupt is issued only for the last item in a packet. The completion bit for the channel is effectively the channel interrupt status bit.

13.3.31 DCP Channel 3 Options Register (HW_DCP_CH3OPTS)

The DCP Channel 3 Options Status register contains optional control information that may be used to further tune the behavior of the channel.

HW_DCP_CH3OPTS: 0x1F0

HW_DCP_CH3OPTS_SET: 0x1F4

HW_DCP_CH3OPTS_CLR: 0x1F8

HW_DCP_CH3OPTS_TOG: 0x1FC

The options register can be used to control optional features of the channels.

Address: HW_DCP_CH3OPTS – 8002_8000h base + 1F0h offset = 8002_81F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RECOVERY_TIMER															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_CH3OPTS field descriptions

Field	Description
31–16 RSVD	Reserved, always set to zero.
15–0 RECOVERY_ TIMER	This field indicates the recovery time for the channel. After each operation, the recover timer for the channel is initiallized with this value and then decremented until the timer reaches zero. The channel will not initiate operation on the next packet in the chain until the recovery time has been satisfied. The timebase for the recovery timer is 16 HCLK clock cycles, providing a range of 0ns to 8.3ms at 133 MHz operation.

13.3.32 DCP Debug Select Register (HW_DCP_DBGSELECT)

This register selects a debug register to view.

This register selects debug information to return in the debug data register.

Address: HW_DCP_DBGSELECT – 8002_8000h base + 400h offset = 8002_8400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																								INDEX							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_DBGSELECT field descriptions

Field	Description
31–8 RSVD	Reserved, always set to zero.
7–0 INDEX	Selects a value to read through the debug data register. 0x01 **CONTROL** — 0x10 **OTPKEY0** — 0x11 **OTPKEY1** — 0x12 **OTPKEY2** —

HW_DCP_DBGSELECT field descriptions (continued)

Field	Description
	0x13 **OTPKEY3** —

13.3.33 DCP Debug Data Register (HW_DCP_DBGDATA)

Reading this register returns the debug data value from the selected index.

This register returns the debug data from the selected debug index source.

Address: HW_DCP_DBGDATA – 8002_8000h base + 410h offset = 8002_8410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_DBGDATA field descriptions

Field	Description
31–0 DATA	Debug Data

13.3.34 DCP Page Table Register (HW_DCP_PAGETABLE)

The DCP Page Table register controls the virtual memory functionality of the DCP. It provides a base address for the page table as well as an enable/disable bit and the ability to flush the cached page table entries.

This register returns the debug data from the selected debug index source.

Address: HW_DCP_PAGETABLE – 8002_8000h base + 420h offset = 8002_8420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BASE[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BASE[15:2]														FLUSH	ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_PAGETABLE field descriptions

Field	Description
31–2 BASE	Page Table Base Address. The page table must be word aligned and the pointer should reference a page table in the standard ARM format.
1 FLUSH	Page Table Flush control. To flush the TLB, write this bit to a 1 then back to a 0.
0 ENABLE	Page Table Enable control. Virtual addressing will only be used when this bit is set to a 1. Disabling the page table will not flush any cached entries, so software should write the FLUSH high and enable LOW when updating page tables.

13.3.35 DCP Version Register (HW_DCP_VERSION)

Read-only register indicating implemented version of the DCP.

This register returns the debug data from the selected debug index source.

Address: HW_DCP_VERSION – 8002_8000h base + 430h offset = 8002_8430h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DCP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-onlyl value reflecting the MAJOR version of the design implementation.
23–16 MINOR	Fixed read-onlyl value reflecting the MINOR version of the design implementation.
15–0 STEP	Fixed read-onlyl value reflecting the stepping of version of the design implementation.

13.4 Disclaimer

The license for the AEC code is documented here for compliance:

Chapter 14
External Memory Interface (EMI)

14.1 Overview

The i.MX28 supports off-chip DRAM storage through the EMI controller, which is connected to the four internal AHB/AXI busses. The EMI supports multiple external memory types, including the following:

- 1.8-V mobile DDR1 (LP-DDR1)
- Standard 1.8 V DDR2
- Low-voltage 1.5 V DDR2 (LV-DDR2)

The EMI uses two primary clocks: the AHB bus, HCLK, and the DRAM source clock, EMI_CLK. The maximum specified frequencies for these two clocks are 200 MHz. The HCLK and EMI_CLK can be either synchronous or asynchronous, configurable.

The EMI consists of the following 3 major components:

- AXI/AHB bus interface and multi-port arbiter
- DRAM Control Logic
- DRAM PHY

14.1.1 Block Diagram

A high-level block diagram of the EMI is shown in the following figure.

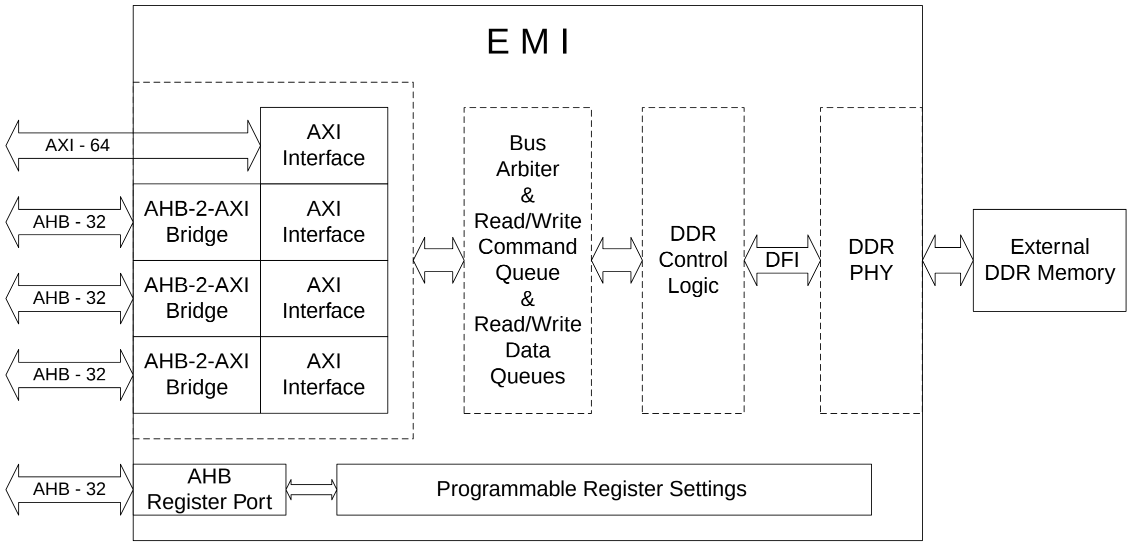


Figure 14-1. EMI Top-Level Block Diagram

EMI implements the following five bus interfaces:

- One 64-bit AXI bus interface
- Three 32-bit AHB bus interfaces to read or write data from or to external DDR memory
- One AHB interface dedicated to read/write PIO registers

14.2 EMI Address Mapping

The EMI automatically maps AXI and AHB bus addresses to the DRAM memory in a contiguous block. Addressing starts at user address 0 and ends at the highest available address according to the size and number of DRAM devices present. This mapping is dependent on how the memory controller was configured and how the parameters in the internal EMI registers are programmed. The exact number and values of these parameters depends on the configuration and the type of memory for which the memory controller was designed.

The mapping of the address space to the internal data storage structure of the DRAM devices is based on the actual size of the DRAM devices available. The size is stored in user-programmable parameters that must be initialized at power up. Certain DRAM devices allow for different mapping options to be chosen, while other DRAM devices depend on the burst length chosen.

14.2.1 DDR SDRAM Address Mapping Options

The address structure of DDR SDRAM devices contains five fields. Each of these fields can be individually addressed when accessing the DRAM. The address map for this EMI is ordered as follows:

Chip Select → Row → Bank → Column → Data Path

The maximum widths of the fields are based on the configuration settings. The actual widths of the fields may be smaller if the device address width parameters (addr_pins, eight_bank_mode, and column_size) are programmed differently.

- Chip Select—EMI supports 2 chip-select. The “Chip Select” field occupies 1-bit.
- Datapath—EMI supports 16-bit data width for external DDR memory. The “Datapath” field occupies 1-bit.
- Bank—The DDR memory device defines the bank number. The EMI supports 4 or 8 banks of DDR memory, which occupies 2 or 3-bits
- Row, Column—The DDR memory device defines the width of these fields

14.2.2 Maximum Address Space

The maximum user address range is determined by the width of the memory datapath, the number of chip select pins, and the address space of the DRAM device. The maximum amount of memory can be calculated by the following formula:

chip-selects x bank x 2^(row + column) x datapath (unit: byte)

For this DRAM controller, the maximum values for these fields are as follows:

- Chip Selects = 2
- Bank : 4 or 8, max 8
- Row : max 15-bits
- Column : max 12-bits
- Datapath width = 2 bytes (16-bits)

As a result, the maximum memory area supported by EMI module is 4 Gbytes. But in i.MX28, the system memory-map limits the maximum accessible area to 1 Gbytes.

14.2.3 Memory Mapping to Address Space

The following figures provide examples of address space mapping from a 32-bit bus address to DRAM address fields. The first example corresponds to a memory device with 8 banks, 13 row bits, and 10 column bits. The second example corresponds to a memory device with 8 banks, 14 row bits, and 10 column bits.

dont care [31:29]	chip-select [28]	row [27:14]	bank [13:11]	column [10:1]	datapath [0]

Figure 14-2. Memory Address Mapping Example 1

dont care [31:28]	chip-select [27]	row [26:14]	bank [13:11]	column [10:1]	datapath [0]

Figure 14-3. Memory Address Mapping Example 2

14.3 EMI Clocking

The EMI uses an external clock input from the ASIC device. Considering the design speed and complexity, this design may not be able to adhere to the skew and delay requirements of a PLL-based clocking scheme. As an alternative, this EMI uses a clocking system that propagates a clock from the memory controller to the memory devices. The largest challenge with this clock forwarding scheme is the management of the clock skew for the round-trip data transfer.

14.3.1 General System Behavior

In this system, the command and address for the transaction are sent from the memory controller coincident with the falling edge of the memory controller clock. Because the clock, command, and address signals all have roughly the same pad and flight delays to travel to the memory, the rising edge of the clock at the memory is centered with the command and address signals, allowing reliable capture. In addition, because all signals are sourced from the same location, this clocking system can tolerate a significant skew and provides more flexibility in timing.

14.3.2 Changing Input Clock Frequency

The operating frequency of the EMI is dependent on an ASIC-level input clock. There are situations in which the user may wish to modify the frequency of the clock without resetting the memory controller. To change the clock frequency at which the EMI operates, the memory controller must stop processing requests, the clock must be adjusted, the memory controller's timing parameters must be reprogrammed, and then the memory controller can be restarted. The procedure to follow for changing the clock frequency is as follows:

1. Ensure that the EMI is idle, which is when the controller_busy signal is low.
2. Put the memory devices into self-refresh mode by asserting the srefresh parameter to ‘b1. It is imperative that the memory devices be placed into self-refresh through this parameter and not through any other means. If the devices were placed into self-refresh mode through one of the low-power modes (programming of the lowpower_control parameter), then the user should first bring the devices out of that low power mode manually and then program the srefresh parameter with a ‘b1.
3. Wait until the memory devices have been placed into self-refresh mode, as indicated by the cke_status signal. This signal is the value of the control_cke signal inside the memory controller, delayed by the number of clocks specified in the cke_delay parameter.
4. Mask the DLL lock state change bit (bit 8) in the int_status parameter by setting bit 8 in the int_mask parameter to ‘b1.
5. Stop the memory controller by writing a ‘b0 to the start parameter.
6. Clear the DLL lock state change bit (bit 8) in the int_status parameter by writing bit 8 in the int_ack parameter to ‘b1. Also, clear any other interrupts in the int_status parameter by writing to the int_ack parameter.
7. If the user wishes to use a controller interrupt to signal when the DLL has relocked, then unmask the DLL lock state change bit (bit 8) in the int_status parameter by clearing bit 8 in the int_mask parameter to a ‘b0. If the user wishes to poll the dlllockreg parameter for this indication, then this step may be skipped.
8. The clock frequency may now be changed. Once the clock frequency has stabilized, the user should modify any parameters that may be affected by the frequency change such as caslat_lin, caslat_lin_gate, any of the timing parameters, and so on. The user should review the parameters carefully to ensure that all parameters that require modification are changed.
9. After all parameters have been updated, restart the EMI by writing a ‘b1 to the start parameter. This forces the DLL to lock to the new frequency.
10. Once the DLL has locked and the PHY has initialized, the memory controller input signal dfi_init_complete will be asserted. If the user is using a controller interrupt to monitor DLL re-lock, then wait for a controller interrupt or poll the int_status parameter for the DLL lock state change bit (bit 8) to be set. If the controller interrupt is not being used, poll the dlllockreg parameter for the bit to be set.
11. At this point, the user may bring the memory devices out of self-refresh by clearing the srefresh parameter to ‘b0. The user does not need to wait to send commands to the memory controller after clearing the srefresh parameter; the memory controller will adjust for self-refresh exit time before processing memory commands.

14.4 EMI AHB and AXI Interface

This section discusses how to cofingure and use the EMI AHB and AXI interface.

14.4.1 AHB-AXI Bridges

An incoming AHB transaction is translated to an AXI command in the AHB-AXI bridge. The AHB-AXI bridges support the AHB-lite protocol and are designed for multilayer AHB architectures. This implies that an AHB slave port never responds with a SPLIT or a RETRY response type. Early termination of AHB bursts is fully supported.

Note that an the AHB slave port can be used in a system with masters that support the full AHB protocol as long as the system is multilayer AHB. For simplicity, the AHB-AXI bridges do not support exclusive access or lock accesses.

14.4.2 AXI Interfaces

The AXI interfaces function as AXI slaves to the AHB-AXI bridges. Transfers are burst-based of variable byte counts. The transfer types INCR and WRAP are fully supported. FIXED burst types are not supported.

Each interface contains five separate channels of traffic to/from the memory: write response, read command, write command, read data, and write data.

14.4.2.1 Internal Command Handling

The EMI uses placement logic to fill the command queue with a command order that maximizes the throughput and efficiency of the core logic. Command reordering in the placement queue supports AXI and AHB restrictions.

Note: The AHB bus does not allow multiple threads on a single port; therefore, ports with an AHB-AXI bridge only use thread ID “0” for all commands.

Note: For simplicity, ports that connect to an AHB bus through an AHB-AXI bridge will be referenced as “AHB-bridged” ports. Ports that connect directly to an AXI bus will be referenced as “native AXI” ports.

If the placement logic is being used, the EMI will optimize the core by re-ordering read and write commands as necessary based on these rules:

- For the AHB-bridged ports, read commands will always execute in the same order as they were accepted into the port. Similarly, write commands will always execute in the same order as they were accepted into the port.
- For the native AXI ports, read commands from the same thread ID will always execute in the same order as they were accepted into the port. However, read commands from different thread IDs on the native AXI ports may be automatically re-arranged in the core logic to execute out-of-order. When commands from different thread IDs are re-ordered, read data returned to the AXI port interfaces will also be out-of-order and may be interleaved. To avoid re-ordering within a port, the AXI bus master should use one thread ID for all commands from any port.
- For the native AXI ports, write commands from the same or different thread IDs will always execute in the same order as they were accepted into the port.
- Read or Write commands from different AHB-bridged or native AXI ports may be automatically re-arranged in the EMI to execute out-of-order.
- Within an AHB-bridged port, a write command that is accepted into the port after a read will not be executed ahead of the read command. However, a read command that is accepted into the port after a write may be indeterminably re-arranged in the EMI ahead of the write command.
- Within a native AXI port, read and write commands from the same or different thread IDs may be re-arranged in the core logic. Read and write commands with different thread IDs will be automatically re-arranged to execute in an optimal order, as long as there are no collisions between the commands.

Command handling is summarized in the following table.

Table 14-1. Reordering/Interleaving Behavior

Commands	Thread ID	Can Commands be Rearranged and Data be Interleaved?
Two Read Commands from 1 port	Same	No
Two Read Commands from 1 port	Different	Yes[1]
Two Read Commands from Different ports	Same or Different	Yes
Two Write Commands from 1 port	Same	No
Two Write Commands from 1 port	Different	No 1
Two Write Commands from Different ports	Same or Different	Yes
Read/Write or Write/Read Command from 1 port	Same	Maybe[2]
Read/Write or Write/Read Command from 1 port	Different	Yes 1
Read/Write or Write/Read Command from Different ports	Same or Different	Yes

1. Since all commands from an AHB-bridged port use a single thread ID, this option is not applicable for AHB-bridged ports.
2. For a native AXI port, read and write commands may be interleaved. For an AHB-bridged port, a write command will not be placed ahead of a read command from the same port. However, a read command may be placed ahead of a write command.

An incoming AXI transaction is mapped into a core logic-level transaction, then synchronized from the AXI clock domain to the core logic clock domain and stored in the AXI port FIFOs. Each instruction consists of an address, size, length and thread ID. Since a port may utilize multiple thread IDs, the source ID that is used in the core logic is a combination of both the port and thread information. This concatenation occurs in the Arbiter and this source ID is used in the placement logic. From the AXI FIFOs, the transaction is presented to the Arbiter which arbitrates requests from all ports and forwards a single transaction to the core logic.

14.4.2.2 AHB Signal to AXI Signal Translation

Table 14-2. Burst Type Translation

AHB (ahb_hburst)		AXI Burst type (axi_awburst / axi_arburst)		AXI Burst length (axi_awlen / axi_arlen)	
SINGLE	‘b000	FIXED	‘b00	1 word	‘b0000
INCR	‘b001	INCR	‘b01	4 words	‘b0011
WRAP4	‘b010	WRAP	‘b10	4 words	‘b0011
INCR4	‘b011	INCR	‘b01	4 words	‘b0011
WRAP8	‘b100	WRAP	‘b10	8 words	‘b0111
INCR8	‘b101	INCR	‘b01	8 words	‘b0111
WRAP16	‘b110	WRAP	‘b10	16 words	‘b1111
INCR16	‘b111	INCR	‘b01	16 words	‘b1111

14.4.2.3 Configured Options

Each AXI port has been defined for the requirements of the intended system. The configured options are:

- Type of interface to the EMI core clock (emi_clk)

  All ports are clock domain programmable relative to the emi_clk. These ports initialize in asynchronous operation, but can be changed by programming the associated axiY_fifo_type_reg parameter. For more information on the programming of this parameter, refer to section Port Clocking. Asynchronous FIFOs handle the clock domain crossing when operating in any of the non-synchronous modes.

- Width of the ID

  Each port is configured with a thread ID of 10 bits.

- Priority Definition

  Command priority is defined based on the port and the command type. For each port Y, there is an axiY_r_priority parameter which defines priorities for all read commands and an axiY_w_priority parameter which defines priorities for all write commands. Supported priority values range from 0 to 7, with 0 as the highest priority.

- Register Port

  The 32-bit AHB register port is fixed in Asynchronous mode between the AHB bus clock domain and the emi_clk domain.

- Port Bridging

  The following table shows the front-end bridge settings.

Table 14-3. Front-End Bridge Settings

Port Number	Bus Connection	Internal Port Type
Port 0	AXI	Native AXI
Port 1	AHB	AHB-AXI Bridged
Port 2	AHB	AHB-AXI Bridged
Port 3	AHB	AHB-AXI Bridged

- Buffering

  Each AXI interface contains a command, a read and a write FIFO, and a response storage array. In addition, each programmable port contains an asynchronous response FIFO to synchronize the memory response to the port clock domain when operating asynchronously.

Table 14-4. AXI Port FIFOs

Port Number	Port Data Width	Clock Domain Type	Command FIFO Depth	Write FIFO Depth	Read FIFO Depth	Response FIFO Depth	Response Storage Array Depth
Port 0	64	Programmable	4	8	8	4	8
Port 1	32	Programmable	2	8	8	4	4
Port 2	32	Programmable	2	8	8	4	4

Port Number	Port Data Width	Clock Domain Type	Command FIFO Depth	Write FIFO Depth	Read FIFO Depth	Response FIFO Depth	Response Storage Array Depth
Port 3	32	Programmable	4	8	8	4	8

- Exclusive Access Buffer Depth

  Exclusive access is an optional AXI feature that is only supported for the native AXI ports. This type of access will only be used if exclusive access commands are issued to the memory controller by driving the axiY_ARLOCK signal to ‘b10 with a read command.

  Each native AXI port contains 1 exclusive buffer and therefore each port may monitor the exclusivity of up to 1 transaction at any time. Refer to section Exclusive Access for more information.

- Error Detection

  When an illegal operational condition is detected on a new AXI transaction entering the port, the port responds through an AXI error signal and the controller interrupt signal, and the error signature is recorded in the register space.

  The AXI error signal flagged is dependent on the type of transaction that caused the error (read or write). The controller interrupt and the signature information is dependent on type of error (command or data). Refer to section Error Responses for more information on error detection.

14.4.3 Port Clocking

There are four user-selectable modes of operation for each of the AXI port interfaces. The mode is set by programming the corresponding axiY_fifo_type_reg parameter. The four settings are:

- Synchronous (‘b11)

  The AXI port clock and the emi_clk must be aligned in frequency in phase. The AXI port interface block will not be required to perform any clock synchronization in any of the FIFOs.

- 1:2 Port:Core Pseudo-Synchronous (‘b10)

  Reserved. Do not select this mode. Select “Asynchronous” instead.

The port operates at half of the frequency of the emi_clk frequency, with clocks that are aligned in phase. One stage of the two-stage synchronization logic of the FIFOs will be utilized to synchronize commands, write data and read data to the appropriate clock domain.

- 2:1 Port:Core Pseudo-Synchronous (‘b01)

  Reserved. Do not select this mode. Select “Asynchronous” instead.

  The port frequency is twice the emi_clk frequency, although the clocks are aligned in phase. One stage of the two-stage synchronization logic of the FIFOs will be utilized to synchronize commands, write data, and read data to the appropriate clock domain.

- Asynchronous (‘b00)

  The AXI bus and the core logic operate on clocks that are mismatched in frequency and phase. The AXI port FIFOs use two stages of synchronization logic to synchronize commands, write data, and read data to the appropriate clock domain.

14.4.4 AXI Interface FIFOs

Each programmable port contains four FIFOs for commands, read data, write data and response synchronization. The response synchronization FIFOs are only used when operating in asynchronous mode. In addition to the FIFOs, each port contains a storage array to hold the read and write responses. The depths and clock domain relativity of the FIFOs and the array are shown in Table 14-4.

The five channels of traffic and their relationship to the port FIFOs are shown below.

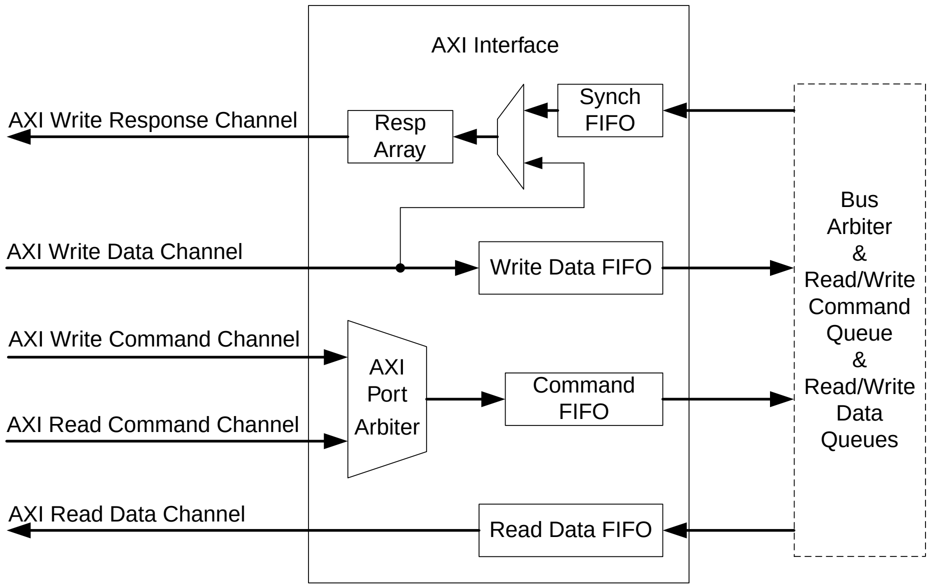


Figure 14-4. AXI Interface FIFOs

14.4.5 AXI Write Response Channel

When a write request is accepted into the AXI interface, an entry will be created in the Write-Response FIFO for that command. When a write response is ready, the response will be returned to the AXI master.

Different masters may require this response at different stages of the write command. A master that needs to quickly release the bus would optimally receive the completion response as soon as the port has accepted the write command and all of the corresponding data. Another master may wish to wait until the data has been accepted into the core logic, or successfully written to memory.

Each AXI interface is configured with two signals that work together to determine when an instruction is considered complete and the write completion response will be returned to the master. These signals are axi_AWCACHE [0] and axi_AWCOBUF. The following table details the relationship between these signals.

Table 14-5. Write Response Types

axi_AWCACHE[0]	axi_AWCOBUF	Response Information[1]
0	Irrelevant	Non-bufferable write command. Response is ready after the write data is committed to memory.
1	0	Standard bufferable write command. Response is ready when the command and all associated data have been received by the AXI data port. There is no guarantee of data coherency across all AXI ports.

axi_AWCACHE[0]	axi_AWCOBUF	Response Information[1]
1	1	Coherent bufferable write command. Response is ready when the command has been accepted by the command queue in the core logic. This guarantees data coherency across all ports but reduces the overall write response latency relative to the non-bufferable option.

1. The response will only be sent if all of the older write responses have been issued for any thread ID on that port.

axi_AWCACHE [0] comes from the standard AXI bus signal and the axi_AWCOBUF is configured by user-defined register “BRESP_TIMING” (Bit [0] of register HW_DRAM_CTL00).

Note that in the “AHB-AXI Bridge”, axi_AWCACHE [0] is tied-off to 1, and the axi_AWCOBUF is tied-off to 0. Therefore, the write reponse type of AXI Interface 1~3 is forced to “Standard Bufferable Write.”

14.4.6 AHB Register Port

The register port is an independent AHB port to the EMI. This port operates asynchronously. The register port only supports the AHB SINGLE burst type.

The register port only supports transfer types of NONSEQ or IDLE. There is no support for INCR or WRAP burst types. There is no support for SEQ or BUSY transfer types.

14.4.7 AXI Transactions

The AXI Interface supports burst tpye of INCR and WRAP.

The AXI Interface supports burst length of 1-16 beats.

14.4.8 Exclusive Access

Note: The exclusive access option is an AXI-specific feature that requires use of the axi_ARLOCK and axi_AWLOCK signals. Therefore, this feature is only relevant for the native AXI ports.

The exclusive access feature allows a master to monitor if a memory area has been altered since its last read. Exclusive access does not imply that the memory area is locked; other thread IDs of that port, or other ports, may access the area for reads or writes even though an exclusive access exists. If any writes occur to a memory area with a valid exclusive access request, the master will lose exclusivity and be informed of this status when it attempts to write to the area again. A loss of exclusivity does not trigger an interrupt or any error conditions; however, the AXI protocol requires that the write data is not written to memory if an exclusive write fails its exclusivity check. The master that has lost exclusivity must determine whether to restart the sequence by requesting another exclusive read or to write the data to the memory regardless through a non-exclusive write.

14.4.8.1 Initiating an Exclusive Access Request

Exclusivity is enabled by issuing a read command to memory with the axi_ARLOCK signal driven to ‘b01.

14.4.8.2 Verifying the Request

The AXI specification dictates restrictions for exclusive access requests. After being accepted in the port, an exclusive read request will be checked against these requirements. If any of these conditions are violated, the command will be passed to the core logic as a non-exclusive read. These restrictions are:

- The address must be aligned to the total byte count. (Byte Count = (axi_ARLEN + 1) x 2^axi_ARSIZE)
- The byte count must be a power-of-two, and less than 128 bytes.

In addition, the AXI specification states that exclusive accesses can not be cacheable. However, since the EMI does not support cacheable commands, this bit is ignored and any cacheable exclusive access requests will be processed as non-cacheable exclusive accesses.

14.4.8.3 Validity of an Exclusive Request

The exclusive access will be considered valid from the point that the entry is created in the buffer until an activity occurs to invalidate it. These activities cause the “valid” bit of the exclusive access buffer entry to be cleared:

- The same source ID issues a write command for the same starting addressing of the same length and beat size.
- Any other source ID, from that port or another port, writes to any memory address in that memory span.

Note: The EMI assumes that a single master will only communicate with a single port. Therefore, a command to another port, with the same AXI ID (axi_AWID or axi_AWID) will not violate the exclusivity of another port’s exclusive region simply based on ID match.

- The user is tracking enough exclusive access commands that the exclusive access buffer for this port is full. Newer commands will be stored, overriding a previous exclusive access entry.
- The same source ID issues another exclusive read request for another memory region. This will not clear the valid bit, but will actually overwrite the entire exclusive access buffer entry (and then set the valid bit). The EMI will only track one exclusive region for each source ID.

14.4.8.4 Read Commands to Exclusive Access Regions

A memory area is not locked while an exclusive access is valid and therefore other source IDs may issue read commands to a memory area identified as exclusive. Reads, even other exclusive reads, do not affect the exclusivity of a region. Normal read commands will be processed as usual and, if successful, respond to the master with an OKAY (‘b00) on the axi_RRESP signal. Exclusive read commands will result in additional entries being created in the exclusive access buffers and, if successful, an EXOKAY (‘b01) response will be sent on the axi_RRESP signal.

Note: An exclusive read request from any source ID, from the same port or another port, even to an overlapping address within an exclusive access region, will not affect the initial exclusive access. Another entry will be created in that port’s buffer for this request.

14.4.8.5 Non-Exclusive Write Commands to Exclusive Access Regions

Any master in the system may issue a write command to a memory area identified as exclusive. However, a standard write to any address within an active exclusive access region will invalidate the exclusivity of that region. In this case, the “valid” bit of any exclusive access buffer entry that spans the modified locations in memory will be cleared and future exclusive write requests to this region will fail their exclusivity check.

14.4.8.6 Exclusive Writes

An exclusive write command is only distinguished from a standard write command by the axi_AWLOCK signal being driven to ‘b01. When an exclusive write is received, the EMI first compares the source ID (axi_AWID and port number), transaction beat length (axi_AWLEN), transaction start address (axi_AWADDR), and transaction beat size (axi_AWSIZE) to the entries in the exclusive access buffer of that port.

14.4.8.7 Passing Exclusive Access Check

If any of the valid buffer entries exactly match the incoming command, the following activities occur:

- That buffer entry's “valid” bit is cleared.
- All other buffer entries, from this port or other ports, are checked to see if this write invalidates any other exclusive access regions. If so, the “valid” bits of these entries will be cleared.
- The transaction is processed.
- If successful, the write response channel (axi_BRESP) returns an EXOKAY response (‘b01) to the master.

14.4.8.8 Failing Exclusive Access Check

If an invalid buffer entry exactly matches the incoming command, or no buffer entries exactly match the incoming command, then the write will fail. The command will be processed internally as a flushed write command.

For a flushed write, the write data will be cleared out of the memory controller FIFOs but the data stored in external DRAM memory will NOT change.

The user will be informed of this condition through the write response channel (axi_BRESP). However, instead of an EXOKAY response, the memory controller will issue an OKAY response. This indicates to the master that the exclusive write did not occur. The master may re-issue the request as a non-exclusive write, or may restart the process by re-issuing the exclusive read.

14.4.9 Error Responses

This section discusses AXI error responses and AXI error reporting.

14.4.9.1 AXI Error Response

When an illegal operational condition is detected on a new AXI transaction entering the port, the port responds with an error condition. Instructions that generate AXI errors result in unpredictable behavior and may cause memory corruption and/or hang conditions.

14.4.9.2 AXI Error Reporting

If an AXI command error occurs, a bit will be set in the int_status parameter and the address and source ID of the command are saved in the port_cmd_error_addr and port_cmd_error_id parameters, respectively. In addition, the access type that relate to the error are stored in the port_cmd_error_type parameter.

Similarly, when a data error occurs, the source ID of the command is saved in the port_data_error_id parameter. The access type that relate to the error are stored in the port_data_error_type parameter.

The bits in the error type parameters are not exclusive. Multiple bits may be set to indicate the type of errors that occurred. Reading the int_ack parameter will allow future errors to be captured in these error parameters.

If multiple errors occur prior to an acknowledgment of the first error, the parameters will still represent the first error attributes. Other error signatures will be lost. If multiple errors occur simultaneously on different ports, the error information will represent the lowest numbered erring port.

14.4.10 Arbiter

From the port interface blocks, commands are presented to the Arbiter, which is responsible for arbitrating between the port requests and sending a single command to the core logic. Refer to the “EMI Multi-Port Arbiter” Chapter for details.

14.4.11 Write Data Queue

The write data queue is a write data storage array for transactions. The queue consists of multiple buffers holding write data for the write requests of a particular port. Write data is stored in these buffers for commands in the command queue until the command is processed in the placement logic and needed by the DRAM command arbitration logic. The buffers can accept data whenever any space is available.

14.4.12 DRAM Command Processing

The DRAM command processing logic is used to process the commands in the Command Queue. The logic organizes the commands to the memory devices in such a way that data throughput is maximized. DRAM-Bank opening and closing cycles are used for data transfers.

The logic uses a variety of factors to determine when to issue bank open and close commands. The logic reviews the entire Command Queue for look-ahead of which banks are to be accessed in the future. The timing is then set to meet the trc and tras_min timing parameters of the memory devices, values which were programmed into the memory controller on initialization. This flexibility allows the memory controller to be tuned to extract the maximum performance out of memory devices. The parameters that relate to DRAM device protocol are listed in the “EMI Parameter Descriptions” Chapter.

14.4.13 Latency

By using the placement logic of the command queue in the core logic, a new request through any port can be immediately placed at the top of the command queue or can interrupt an ongoing request. This scheme allows a high priority request to be serviced in the shortest possible time.

However, since there are many factors that determine the placement into the command queue, there are also many factors that affect the actual latency of the command. These factors include:

- The coherency status of the transactions already in the command queue: If there is a data coherency conflict with a transaction already in the command queue, the new transaction will be placed after the transaction that produced the conflict. The position of the conflicting transaction determines the latency of the high priority read or write command.
- The priority status of the transactions already in the command queue: If the new command has a higher priority than those already in the command queue, the new request will be serviced ahead of the lower priority command. As a result, the latency of the new command will be lower than the latency of the older command.
- The read, write, and bank information of the transactions already in the command queue: In general, reads will be placed ahead of writes when both are of the same priority level. Read commands are grouped with other read commands of similar priorities and write commands are grouped with other write commands of similar priorities. Among these groupings, transactions with similar bank and different row destinations are separated as much as possible.

If all of the placement conditions are met, then a new command would be placed at the top of the command queue. However, if the new command is of a higher priority than the transaction that is executing, the current command will be interrupted and the new command will execute first. The interruption will occur at a natural burst boundary of the DRAMs. The interrupted transaction will be placed at the top of the placement queue and it will resume after the new request is completed. The page status of the new transaction determines when the current transaction is interrupted. If the page for the new transaction is already open, then the current transaction will be interrupted at the next natural burst boundary of the DRAM device. If the page is not currently open, then the new request will be placed at the top of the command queue while its page is prepared.

There are a fixed number of latency cycles in the memory controller, based on the pipeline through the memory controller logic. These steps are:

1. Command passing through the port interface. (fixed)
2. Arbitration through the Arbiter. (fixed)
3. Placement into the Command Queue. (fixed)
4. Memory Command Generation. (variable)
5. Sending of control signals from the core logic to flip-flops near the I/O drivers. (fixed)
6. Flight time to the DRAM device. (variable)
7. Flight time from the DRAM device. (variable)
8. For reads, synchronization of read data from the data strobe domain. (fixed)
9. For reads, data pass through the port interface. (fixed)

For asynchronous AXI interfaces, an additional 4–5 cycles are included for the round-trip transaction to synchronize to the EMI_CLK.

14.5 EMI Multi-Port Arbiter

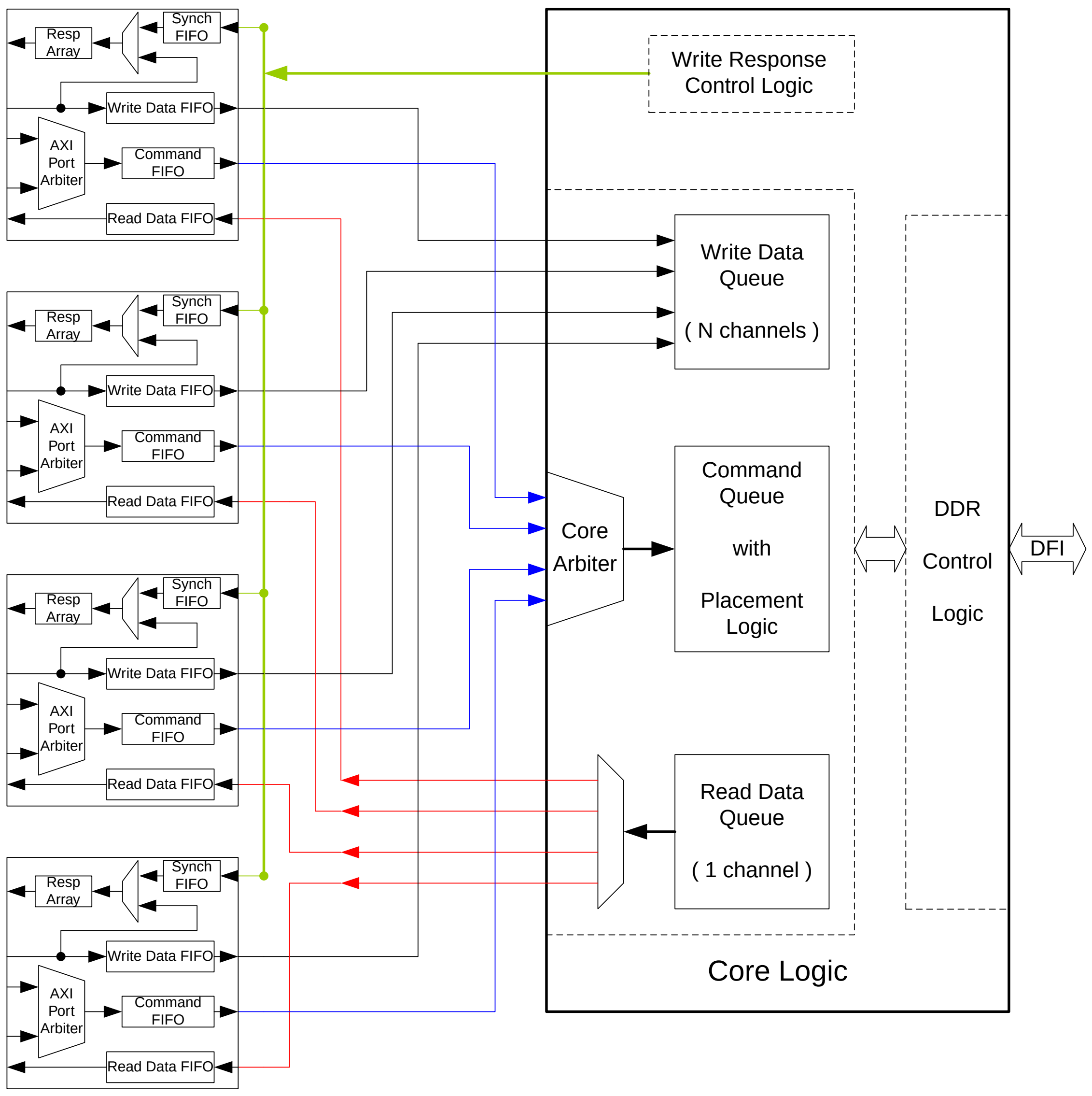


Figure 14-5. EMI Core Logic

The Arbiter is responsible for arbitrating requests from the ports and sending requests to the core logic. Each transaction received at the Arbiter logic has an associated priority, which works with each port's arbitration logic to determine how ports issue requests to the core logic. This memory controller supports the Bandwidth Allocation/Priority Round-Robin arbitration scheme.

The Arbiter logic routes read data from the core logic to the appropriate port. The requesting port is assumed able to receive the data. Write data from each port is connected directly to its own write data interface in the core logic, allowing the ports to independently pass write data to the core buffers.

14.5.1 Arbitration Overview

The bandwidth allocation scheme is an extension of simple round-robin arbitration. It is based on the priority of the requests and is influenced by the actual bandwidth consumed by the port inside the core logic.

Priority round-robin arbitration is a complex arbitration scheme. In order to understand its operation, each concept must be first understood individually. Section Understanding Round-Robin Arbitration through section Understanding Port Bandwidth Hold-Off describe the various components of priority round-robin arbitration.

14.5.2 Understanding Round-Robin Arbitration

Round-robin operation is a simple form of arbitration which offers each port an opportunity to issue a command. This scheme uses a counter that rotates through the port numbers, incrementing every time a port request is granted.

If the port that the counter is referencing has an active request, and the core command queue is not full, then this request will be sent to the core. If there is not an active request for that port, then the port will be skipped and the next port will be checked. The counter will increment by one whenever any request has been processed, regardless of which port's request was arbitrated.

Round-robin operation ensures that each port's requests can be successfully arbitrated into the core logic every N cycles, where N is the number of ports in the EMI. No port will ever be locked out, and any port can have its requests serviced on every cycle as long as all other ports are quiet and the command queue is not full.

An example of the round-robin scheme is shown in Table "Round-Robin Operation Example".

Cycles 0, 2 and 6 show the system behavior when the command queue is full. Cycle 8 shows the system behavior when the port addressed by the arbitration counter does not have an active request. All other cycles show normal behavior.

Table 14-6. Round-Robin Operation Example

CYCLE	PORT ADDRESSED BY THE ARBITRATION COUNTER	PORTS REQUESTING				COMMAND QUEUE FULL ?	WINNER OF ARBITRATION	VALUE OF COUNTER AT NEXT CYCLE
		PORT 0	PORT 1	PORT 2	PORT 3			
0	0	Y	Y	Y	Y	Yes	None	0
1	0	Y	Y	Y	Y	No	P0	1
2	1	-	Y	Y	Y	Yes	None	1
3	1	Y	Y	Y	Y	No	P1	2
4	2	Y	-	Y	Y	No	P2	3
5	3	Y	-	-	Y	No	P3	0
6	0	Y	-	Y	-	Yes	None	0
7	0	Y	-	Y	-	No	P0	1
8	1	-	-	Y	-	No	P2	2
9	2	-	-	Y	Y	No	P2	3
10	3	Y	-	-	Y	No	P3	0

14.5.3 Understanding Port Priority

For AXI ports, the priority is associated with a port and each port has separate priority parameter for reads and writes. These values are stored into the programmable parameters axiY_r_priority and axiY_w_priority (where Y represents the port number) at controller initialization.

Internally, the ports are organized into priority groups based on their priority settings. All ports within a priority group are treated equally for arbitration unless a port has exceeded its allocated bandwidth. The priority value is also used by the placement logic inside the core logic when filling the command queue.

A priority value of 0 is highest priority, and a priority value of (decimal) 7 is the lowest priority. The user may program at priority level 0; however, it is best to reserve this priority value so that the placement queue can elevate to this level through aging.

14.5.4 Understanding Port Bandwidth

Each port has an associated bandwidth limit that sets the maximum percentage of the core logic bandwidth that the port is allowed to use. Once this level is reached, the Arbiter will no longer accept requests from this port until the bandwidth usage drops below the threshold. This scheme allows for the bandwidth to be shared between the ports. If required, an overflow option allows the port to continue to receive requests after the bandwidth limit has been reached. See section Understanding Port Bandwidth Hold-Off for more information on this option.

The bandwidth limits are stored in the programmable parameters axiY_bdw for each port Y at EMI initialization. The axiY_current_bdw parameters are used to track the actual bandwidth utilized as computed by the bandwidth calculation module inside the Arbiter.

Port bandwidth is computed by counting the number of cycles that the core logic is busy actively processing that port’s request in each 100 cycle period, referred to as the statistics window.

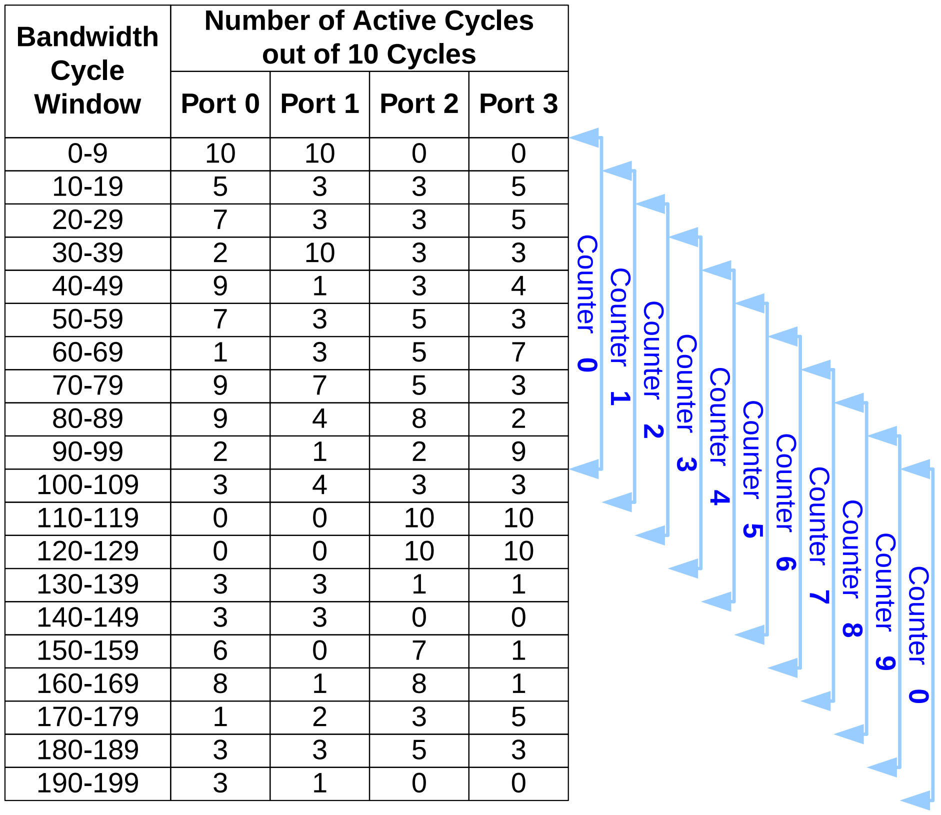
In the EMI, 10 counters are used for this computation. The counters track the number of active cycles in each statistics window, generating a moving average bandwidth value for each port. This is the actual bandwidth utilized value saved in the current bandwidth parameters (axiY_current_bdw). The values in the current bandwidth parameters are updated every 10 cycles with the actual bandwidth used in the last 100 cycles.

The core logic is defined as actively processing for a port if any of the following situations occur:

- The core logic is ready to transfer write data from the port to memory, but the data has not arrived from the port.
- The core logic is holding the port’s command and is ready to transfer to memory, but is waiting to open a bank, precharge a bank, or some other memory-related action.
- The core logic is actively transferring data from the port to memory.
- The core logic is ready to transfer read data from memory to the port but the port is busy and unable to accept the data.

If all ports are assigned 100% bandwidth, then bandwidth usage will not factor and arbitration will be purely based on priority.

The following figure shows bandwidth usage for a 4-port system with a 100-cycle calculation window and 10 counters.



Bandwidth Cycle Window	Number of Active Cycles out of 10 Cycles			
	Port 0	Port 1	Port 2	Port 3
0-9	10	10	0	0
10-19	5	3	3	5
20-29	7	3	3	5
30-39	2	10	3	3
40-49	9	1	3	4
50-59	7	3	5	3
60-69	1	3	5	7
70-79	9	7	5	3
80-89	9	4	8	2
90-99	2	1	2	9
100-109	3	4	3	3
110-119	0	0	10	10
120-129	0	0	10	10
130-139	3	3	1	1
140-149	3	3	0	0
150-159	6	0	7	1
160-169	8	1	8	1
170-179	1	2	3	5
180-189	3	3	5	3
190-199	3	1	0	0

Figure 14-6. System Bandwidth Example

For this system, the bandwidth will be monitored over each 100 cycles. The bandwidth calculation parameters will be updated every 10 cycles with the bandwidth usage of the last 100 cycles as shown graphically in the figure.

The bandwidth totals are shown in #system_bandwidth_usage_example. In the system, these values would be stored in the axiY_current_bdw parameters after each calculation.

Table 14-7. System Bandwidth Usage Example

COUNTER NUMBER	CYCLES COUNTING	CALCULATED USAGE			
		PORT 0	PORT 1	PORT 2	PORT 3
0	0 - 99	61 / 100 = 61%	45 / 100 = 45%	37 / 100 = 37%	41 / 100 = 41%
1	10 - 109	54 / 100 = 54%	39 / 100 = 39%	40 / 100 = 40%	44 / 100 = 44%
2	20 - 119	49 / 100 = 49%	36 / 100 = 36%	47 / 100 = 47%	49 / 100 = 49%
3	30 - 129	42 / 100 = 42%	33 / 100 = 33%	54 / 100 = 54%	54 / 100 = 54%
4	40 - 139	43 / 100 = 43%	26 / 100 = 26%	52 / 100 = 52%	52 / 100 = 52%
5	50 - 149	37 / 100 = 37%	28 / 100 = 28%	49 / 100 = 49%	48 / 100 = 48%
6	60 - 159	36 / 100 = 36%	25 / 100 = 25%	51 / 100 = 51%	46 / 100 = 46%

COUNTER NUMBER	CYCLES COUNTING	CALCULATED USAGE			
		PORT 0	PORT 1	PORT 2	PORT 3
7	70 - 169	43 / 100 = 43%	23 / 100 = 23%	54 / 100 = 54%	40 / 100 = 40%
8	80 - 179	35 / 100 = 35%	18 / 100 = 18%	52 / 100 = 52%	42 / 100 = 42%
9	90 - 189	29 / 100 = 29%	17 / 100 = 17%	49 / 100 = 49%	43 / 100 = 43%
0	100 - 199	30 / 100 = 30%	17 / 100 = 17%	47 / 100 = 47%	34 / 100 = 34%

14.5.5 Understanding Port Bandwidth Hold-Off

When the bandwidth used by a port exceeds its specified limit, that port is held off from subsequent arbitration decisions for a period of time known as the statistics reporting time. This causes a period of inactivity from that port, allowing the actual bandwidth used for that port to fall below the threshold. Since the bandwidth used is updated every ten cycles, the minimum hold off period for this system is ten cycles.

This scheme is designed to constrain individual ports (especially ports programmed at higher priority) from overtaking all available bandwidth and locking out other ports. However, this can have its drawbacks.

Consider a situation where only one port has been actively requesting and has therefore used all of its available bandwidth. The bandwidth hold-off will prevent additional requests from being accepted, even though no other ports are requesting. The core logic command queue will sit empty for several cycles while the hold-off is cleared. This is obviously wasted bandwidth and is detrimental to overall system performance. EMI has incorporated a bandwidth hold-off override function for such a situation in the axiY_bdw_ovflow parameters.

A port will be allowed to exceed its allocated bandwidth when all of these conditions are true:

- The bandwidth overflow parameter (axiY_bdw_ovflow) is set to ‘b1 for port Y.
- No other port, whose bandwidth has not been exceeded, is requesting at the same priority level.
- The command queue has less than the number of entries specified in the arb_cmd_q_threshold parameter.

This last condition is a preventative measure to maintain latency requirements for ports programmed at higher priority. When a port is allowed to exceed bandwidth, it may fill the command queue with transactions. If this occurs, and a higher priority port starts requesting, then there will be no room in the command queue for the new requests. This means that the higher priority port will actually be held off for potentially several cycles. In this situation, even though the core logic bandwidth is being utilized well, the latency requirements of the higher priority port are not being met. The arb_cmd_q_threshold parameter is used to limit the bandwidth overflow and prevent this condition. It ensures that a certain number of slots remain available in the command queue for other ports. As a result, bandwidth overflow will be allowed as long as there are less than arb_cmd_q_threshold number of entries in the command queue.

14.5.6 Priority Round-Robin Arbitration Summary

The MC priority round-robin arbitration system combines the concepts of round-robin operation, priority, bandwidth, and port bandwidth hold-off. The incoming commands are separated into priority groups based on the priority of the associated port for that type of command. Within each priority group, the Arbiter evaluates the requesting ports, the command queue, the priority of the requests, the bandwidth being used and the overflow option to determine the winner of the arbitration.

The order of steps is as follows:

1. Is the core logic command queue full?
   - Yes: No further action is taken.
   - No: Review the ports.
2. For all requests at each priority level, use round-robin arbitration to select a request for that priority level.
3. For the highest priority port that is selected, has the bandwidth allocation for this port been exceeded?
   - No: This request wins arbitration. Move to step 6.
   - Yes: Check the bandwidth overflow status.
4. Is bandwidth overflow enabled?
   - No: Repeat step 3 for the next highest priority request.
   - Yes: Check the conditions for overflow.
5. Are the overflow conditions met?
   - No: Repeat step 3 for the next highest priority request.
   - Yes: This request wins arbitration.
6. Once a request wins arbitration, it is processed into the command queue and the round-robin counter is updated to the next port in the circular queue.

14.5.7 Arbitration Examples

To demonstrate arbitration behavior, consider the following system:

- Four ports.
- Ports 0 and 1 request only at priority 1. Ports 2 and 3 request only at priority 2.
- All ports have bandwidth allocations of 100% and the bandwidth overflow bit is set to ‘b1. (Bandwidth will not affect arbitration.)

An example with these settings is shown in Table 14-8. Note that priority 2 requests will only win arbitration when there are no priority 1 requests. Also note that each arbitration counter only increments, and always increments, when a request of that priority is processed. Cycle 5 demonstrates the always condition of counter incrementing: even though the arbitration was won by the other port of highest priority instead of the one in the counter, the counter still increments.

Table 14-8. Priority Round-Robin without Bandwidth Consideration

Cycle	Current Arbitration Counter		Ports Requesting				Command Queue Full	Winner of Arbitration	Next Arbitration Counter	
	PG 1	PG 2	Port 0	Port 1	Port 2	Port 3			PG 1	PG 2
0	0	2	Y	Y	Y	Y	Yes	None	0	2
1	0	2	Y	Y	Y	Y	No	P0	1	2
2	1	2	—	Y	Y	Y	No	P1	0	2
3	9	2	Y	—	Y	Y	No	P0	1	2
4	1	2	—	—	Y	Y	No	P2	1	3
5	1	3	Y	—	—	Y	No	P0	1	3
6	0	3	—	—	—	Y	No	P3	0	2
7	0	2	—	—	—	—	No	None	0	2
8	0	2	—	—	Y	Y	Yes	None	0	2
9	0	2	Y	—	Y	Y	No	P0	1	2
10	1	2	—	—	Y	Y	No	P2	1	3
11	1	3	—	—	—	Y	No	P3	1	2
PG1 = Priority Group 1 and PG2 = Priority Group 2										

The example was a very simplified case without considering bandwidth. However, in most cases, bandwidth will factor into the arbitration. Therefore, consider the same system as used in the previous example. However, now the allocated bandwidth is less than 100%. This system is shown in Table 14-9. The “Bandwidth Held Off” column indicates if the bandwidth was held off for any port in that cycle. (Note cycles 3 and 11.) Even though the priority group 1 arbitration counter is pointing to port 0 in both cases, because the port is held off, port 0 does not win arbitration. A lower priority port, from priority group 2, wins arbitration instead.

Table 14-9. Priority Round-Robin with Bandwidth Consideration

Cycle	Current Arbitration Counter		Ports Requesting				Command Queue Full	Bandwidth Held Off	Winner of Arbitration	Next Arbitration Counter	
	PG 1	PG 2	Port 0	Port 1	Port 2	Port 3				PG 1	PG 2
0	0	2	Y	Y	Y	Y	Yes	No	None	0	2
1	0	2	Y	Y	Y	Y	No	No	P0	1	2
2	1	2	—	Y	Y	Y	No	No	P1	0	2
3	0	2	Y	—	Y	Y	No	Yes, port 0	P2	0	3
4	0	3	Y	—	—	Y	No	No	P0	1	3
5	1	3	Y	—	—	Y	No	No	P0	0	3
6	0	3	—	—	—	Y	No	No	P3	0	2
7	0	2	—	—	—	—	No	No	None	0	2
8	0	2	—	—	Y	Y	Yes	No	None	0	2
9	0	2	Y	—	Y	Y	No	No	P0	1	2
10	1	2	Y	—	Y	Y	No	No	P0	0	2
11	0	2	Y	—	Y	Y	No	Yes, port 0	P2	0	3
12	0	3	Y	—	—	Y	No	No	P0	1	3
13	1	3	—	—	—	Y	No	No	P3	1	2
PG1 = Priority Group 1 and PG2 = Priority Group 2											

14.5.8 Configuring and Programming for Priority Round-Robin Arbitration

The priority round-robin arbitration scheme requires the use of the following programmable parameters:

- axiY_r_priority, axiY_w_priority, axiY_bdw, axiY_current_bdw, axiY_bdw_ovflow
- arb_cmd_q_threshold

Since these parameters work together, there are trade-offs for the settings.

Note: All of the arbitration parameters must be programmed prior to setting the start parameter to ‘b1. Any subsequent changes to the arbitration parameters may result in unpredictable system operation.

14.5.8.1 Definition of the Statistics Window and the Number of Counters

The statistics window sets the length of time used to measure port bandwidth and the number of counters determines the granularity of the measurement. The values set for both factors affect bandwidth measurement accuracy, gate area and arbitration latency. Accuracy can be added to the bandwidth measurements by increasing the number of counters and the length of time for a calculation. However, this results in a large increase in the gate area.

Arbitration latency for a port is affected by how often bandwidth is updated. This is based on the amount of time over which each counter tabulates activity: the statistics window divided by the number of counters. As a general guideline, the statistics window should be just large enough that the bandwidth used for one complete transaction does not exceed the allocated bandwidth.

In the EMI, the statistics window is defined as 100 cycles, and there are ten counters per port.

14.5.8.2 Command Priority, Port Bandwidth, and Bandwidth Overflow

Command priority, bandwidth and bandwidth overflow play an important role in arbitration. Priority is the most important factor since commands are first sorted into priority groups based on their priority settings. In a multi-port system, all ports with tight latency requirements should be assigned higher priority values (lower numbers). Note that the priority of the command affects both arbitration and placement into the core logic command queue. As a result, it is more complicated to meet the latency requirements of all ports. For high priority commands or ports with low allocated bandwidths, bandwidth overflow may be useful. To mimic simple round-robin arbitration, program all ports to have the same priority and full bandwidth allocations.

Note: The bandwidth parameter (axiY_bdw) is a seven-bit parameter. However, any programmed value greater than 0x64 is interpreted as 100%.

14.5.9 Command Queue with Placement Logic

From the Arbiter, commands are routed to the command queue of the core logic. The command queue is fed using a placement algorithm. For more information on this algorithm, refer to section Core Command Queue with Placement Logic.

14.6 Core Command Queue with Placement Logic

The core logic contains a command queue that accepts commands from the Arbiter. This command queue uses a placement algorithm to determine the order that commands will execute in the core logic. The placement logic follows many rules to determine where new commands should be inserted into the queue, relative to the contents of the command queue at the time. Placement is determined by considering address collisions, source collisions, data collisions, command types and priorities. In addition, the placement logic attempts to maximize efficiency of the core logic through command grouping and bank splitting. Once placed into the command queue, the relative order of commands is constant.

Many of the rules used in placement may be individually enabled/disabled. In addition, the queue may be disabled by clearing the placement_en parameter, resulting in an in-line queue that services requests in the order they are received. If the placement_en parameter is cleared to ‘b0, the placement algorithm will be ignored.

14.6.1 Rules of the Placement Algorithm

The factors affecting command placement all work together to identify where a new command fits into the execution order. They are listed in order of importance, as follows:

- Address collision/data coherency violation
- Source ID collision
- Write buffer collision
- Priority
- Bank splitting
- Read/write grouping

14.6.1.1 Address Collision/Data Coherency Violation

The order in which read and write commands are processed in the memory controller is critical for proper system behavior. While reads and writes to different addresses are independent and may be re-ordered without affecting system performance, reads and writes that access the same address are significantly related. If the port requests a read after a write to the same address, then repositioning the read before the write would return the original data, not the changed data. Similarly, if the read was requested ahead of the write but accidentally positioned after the write, then the read would return the new data, not the original data prior to being overwritten. These are significant data coherency mistakes.

To avoid address collisions, reads or writes that access the same chip select, bank and row as a command already in the command queue will be inserted into the command queue after the original command, even if the new command is of a higher priority.

This factor may be enabled/disabled through the addr_cmp_en parameter and should only be disabled if the system can guarantee coherency of reads and writes.

14.6.1.2 Source ID Collision

Each port is assigned a specific source ID that is a combination of the port and thread ID information, and identifies the source uniquely. This allows the memory controller to map data from/to the correct source/destination.

Note that a source ID does contain port identification information, which means that the rules for placement are dependent on the requesting port. There will not be source ID collisions between ports.

In general, read commands from the same source ID will be placed in the command queue in order. Therefore, a read command with the same source ID as a read command already in the command queue will be processed after the original read command. All write commands from a port, even with different source IDs, will be executed in order.

The behavior of commands of different types from the same source ID is dependent on the user configuration. For this Memory Controller, the placement of new read/write commands that collide in terms of source ID with existing entries in the command queue will only depend on other commands of the same type, not on different types. This means that, if there are no address conflicts, a read command could be executed ahead of a write command with the same source ID, and likewise a write command could be executed ahead of a read command with the same source ID.

This feature will always be enabled.

14.6.1.3 Write Buffer Collision

Incoming write requests in the command queue are allocated to one of the four write buffers of the core logic automatically based on availability. New write commands will be designated to any of the available buffers. However, back-to-back write requests from a particular source ID will be allocated to the same write buffer as the previous command.

Since the core logic must pull data out of the buffers in the order it was stored, if a write command is linked to a buffer that is associated with another command in the queue, then the new command will be placed in the command queue after that command, regardless of priority. This feature will always be enabled.

14.6.1.4 Priority

Priorities are used to distinguish important commands from less important commands. Each command is given a priority based on the command type through the programmable parameters axiY_r_priority and axiY_w_priority (where Y represents the port number). A priority value of 0 is the highest priority, and a priority value of 7 is the lowest priority.

The placement algorithm will attempt to place higher priority commands ahead of lower priority commands, as long as they have no source ID, write buffer or address collisions. Higher priority commands will be placed lower in the command queue if they access the same address, are from the same requestor or use the same buffer as lower priority commands already in the command queue.

14.6.1.5 Bank Splitting

Before accesses can be made to two different rows within the same bank, the first active row must be closed (pre-charged) and the new row must be opened (activated). Both activities require some timing overhead; therefore, for optimization, the placement queue will attempt to insert the new command into the command queue such that commands to other banks may execute during this timing overhead. The placement of the new commands will still follow priority, source ID, write buffer and address collision rules.

The placement logic will also attempt to optimize the core logic by inserting a command to the same bank as an existing command in the command queue immediately after the original command. This reduces the overall timing overhead by potentially eliminating one pre-charging/activating cycle. This placement will only be possible if there are no priority, source ID, write buffer or address collisions or conflicts with other commands in the command queue.

All bank splitting features are enabled through the bank_split_en parameter.

14.6.1.6 Read/Write Grouping

The memory suffers a small timing overhead when switching from read to write mode. For efficiency, the placement queue will attempt to place a new read command sequentially with other read commands in the command queue, or a new write command sequentially with other write commands in the command queue. Grouping will only be possible if no priority, source ID, write buffer or address collision rules are violated.

This feature is enabled through the rw_same_en parameter.

14.6.2 Command Execution Order After Placement

Once a command has been placed in the command queue, its order relative to the other commands in the queue at that time is fixed. While this provides simplicity in the algorithm, there are drawbacks. For this reason, the Memory Controller offers two options that affect commands once they have been placed in the command queue.

14.6.2.1 Command Aging

Since commands can be inserted ahead of existing commands in the command queue, the situation could occur where a low priority command remains at the bottom of the queue indefinitely. To avoid such a lockout condition, aging counters have been included in the placement logic that measure the number of cycles that each command has been waiting. If command aging is enabled through the active_aging parameter, then if an aging counter hits its maximum, the priority of the associated command will be decremented by one (lower priority commands are executed first). This increases the likelihood that this command will move to the top of the command queue and be executed. Note that this command does not move relative positions in the command queue when it ages; the new priority will be considered when placing new commands into the command queue.

Aging is controlled through a master aging counter and command aging counters associated with each command in the command queue. The age_count and command_age_count parameters hold the initial values for each of these counters, respectively. When the master counter counts down the age_count value, a signal is sent to the command aging counters to decrement. When the command aging counters have completely decremented, then the priority of the associated command is decremented by one number and the counter is reset. Therefore, a command does not age by a priority level until the total elapsed cycles has reached the product of the age_count and command_age_count values. The maximum number of cycles that any command can wait in the command queue until reaching the top priority level is the product of the age_count value, the command_age_count value, and the number of priority levels in the system.

14.6.2.2 High-Priority Command Swapping

Commands are assigned priority values to ensure that critical commands are executed more quickly in the memory controller than less important commands. Therefore, it is desirable that high-priority commands pass into the core logic as soon as possible. The placement algorithm takes priority into account when determining the order of commands, but still allows a scenario in which a high-priority command sits waiting at the top of the command queue while another command, perhaps of a lower priority, is in process.

The high-priority command swapping feature allows this new high-priority command to be executed more quickly. If the user has enabled the swapping function through the swap_en parameter, then the entry at the top of the command queue will be compared with the current command in progress. If the command queue's top entry is of a higher priority (not the same priority), and it does not have an address, source ID or write buffer conflict with the current command being executed, then the original command will be interrupted.

For this memory controller, an additional check is performed before a read command is interrupted. If the read command in progress and the read command at the top of the command queue is from the same port, then the executing command will only be interrupted if the swap_port_rw_same_en parameter is set to ‘b1. If this parameter is cleared to ‘b0, a read command from the same port as a read command in progress, even with a higher priority and without any conflicts, would remain at the top of the command queue while the current command completes.

Note: All write commands from a single port, even with different source IDs, will be executed in order. Therefore, two write commands from the same port will never be swapped regardless of the settings of the swap_en and swap_port_rw_same_en parameters.

Note: Priorities are assigned to read commands based on the settings in the axiY_r_priority parameters. While all read commands from a port are assigned the same priority when placed in the command queue, their priorities may change over time through command aging. While uncommon, it is possible that a higher-priority read command may be at the top of the command queue while a lower-priority read command is executing. The behavior of the system in this scenario is based on the value of the swap_en and swap_port_rw_same_en parameters.

Refer to the following table for details:

Table 14-10. Swapping Behavior

Active Command Priority	New Command Priority	Originating Port for Commands	Conflicts?	Action
Higher	Lower	Same or Different	Yes or No	Current command continues

Active Command Priority	New Command Priority	Originating Port for Commands	Conflicts?	Action
Lower	Higher	Same	Yes	Current command continues
Lower	Higher	Same	No	Will swap IF swap_en = 1 and swap_port_rw_same_en = 1
Lower	Higher	Different	No	Will swap IF swap_en = 1

If the command is to be interrupted, it will be halted after completing the current burst, stored and placed at the top of the queue, and the new command will be executed. As long as the command queue is not full, new commands may continue to be inserted into the command queue based on the placement rules, even at the head of the queue ahead of the interrupted command. The top entry in the command queue will be executed next. Whenever the interrupted command is resumed, it will start from the point at which it was interrupted.

Note that priority 0 commands will never be interrupted, so the user should set any commands that should not be interrupted to priority 0. If supported by the port interfaces, setting the swap_port_rw_same_en parameter will enable interleaving.

14.7 DDR PHY

The DDR PHY encapsulates all functionality required to interface to external DDR DRAM devices into a single module. This module is used to control the off-chip data capture and synchronization logic for the read data. This module performs the following functions:

- Contains all data registers used to launch data, address and control signals to the DDR memory and the memory controller.
- Controls the off-chip data capture and synchronization logic for the read data.
- Includes a DLL for timing.

14.7.1 High Level Block Diagram

EMI uses a slice-based approach for the DDR PHY. Each slice manages a byte (8-bit) of data and it’s corresponding dqs and dm signals. A high level block diagram of the PHY is provided below.

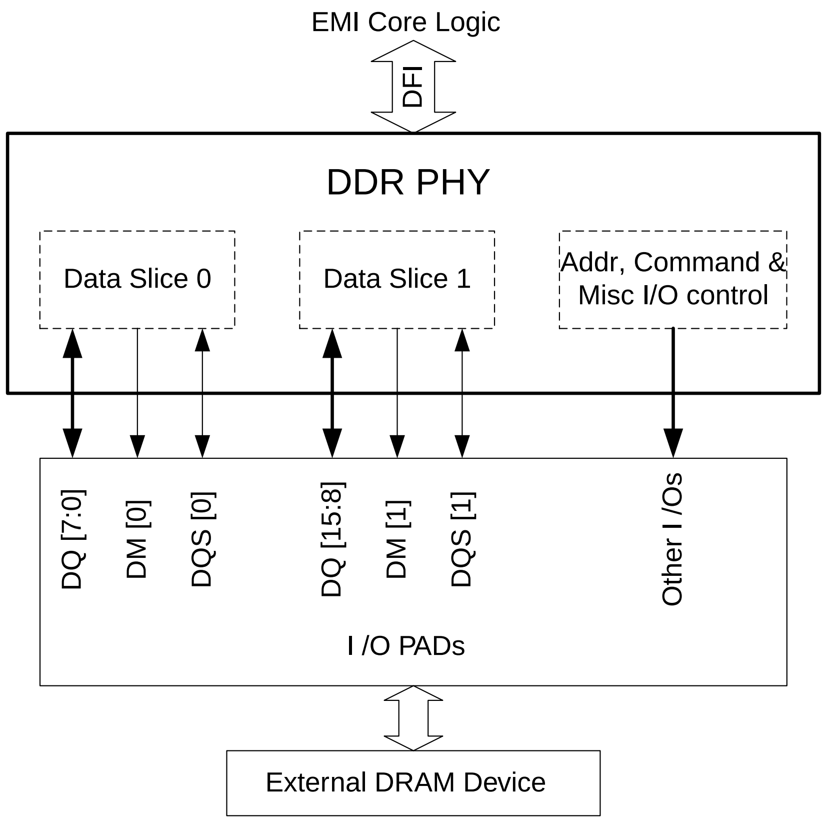


Figure 14-7. DDR PHY High-Level Block Diagram

14.7.2 DFI

The interface between the EMI core logic and the DDR_PHY is named as "DFI".

"D" stands for "DRAM Controller" while "FI" is alias to "PHY".

14.7.3 I/O Timing of Address and Command

The figure below illustrates the I/O timing of Address and Command.

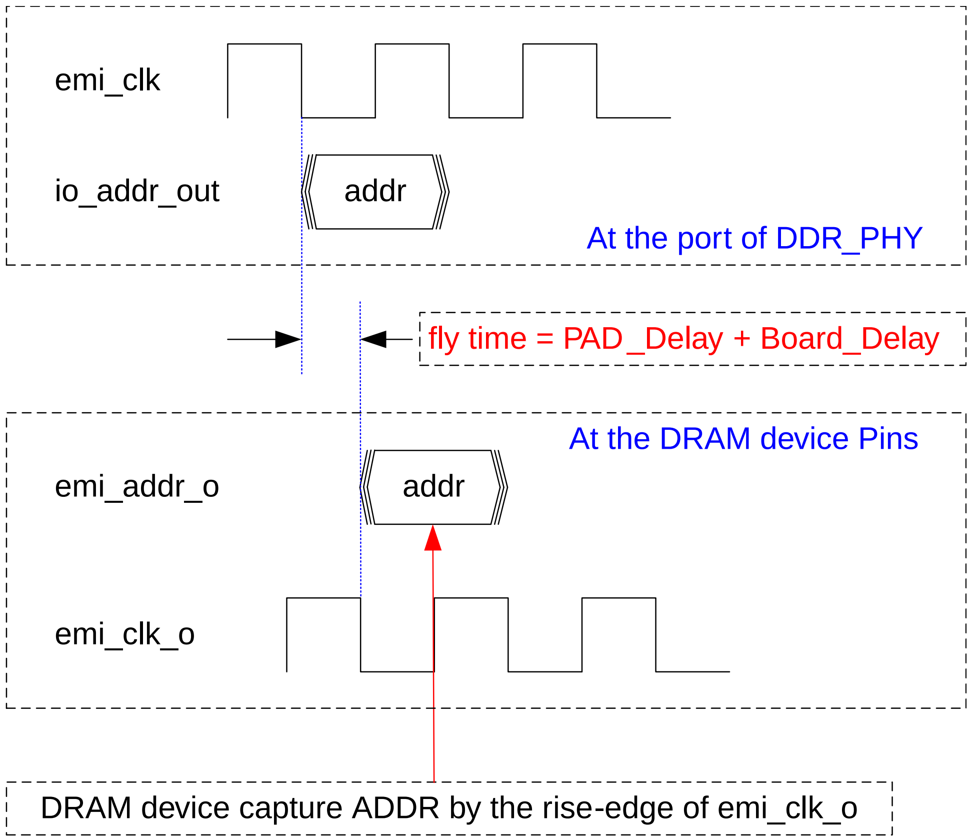


Figure 14-8. I/O Address Timing

The address and command signals including all DRAM interface signals except the DQ and DQS: clock-enable (CKE), chip-select, bank-address, row/cloumn_address, CASn, RASn, WEn, and so on.

- The Address and command signals are latched out by the falling-edge of emi_clk; In another words, the rise-edge of emi_clk is center-aligned with the output Address and Command signals.
- The fly-time is the signal's propagation time start from the DDR_PHY logic, go across the output I/O (PAD) of i.MX28, go through the signal trace on the board, then arrive the PIN of the DRAM device.
- The DRAM device captures the Address and Command signals with the rising edge of emi_clk_o.

14.7.4 Data Slice Overview

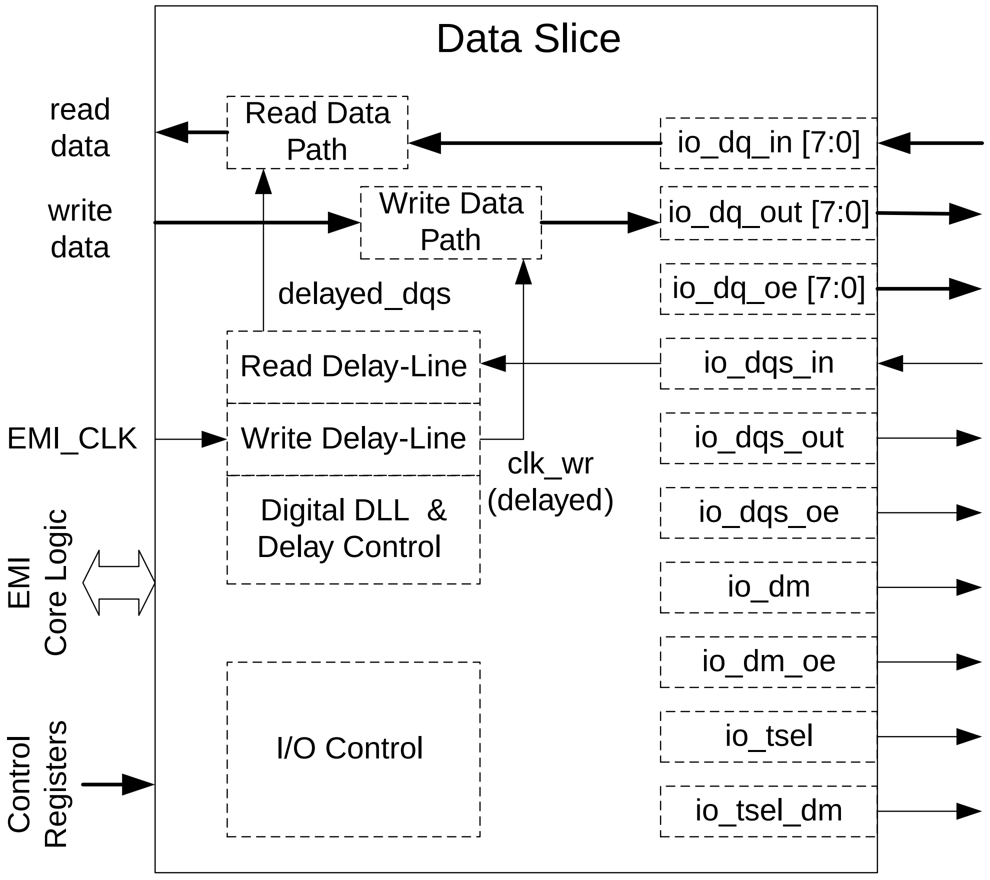


Figure 14-9. DDR PHY Data Slice

Each data-slice manages a byte (8-bit) of data and it’s corresponding signals.

The Read Data Path captures read data by latching it into read data buffers by both the posedge and negedge of delayed_dqs. Then, the read data is synchronized from delayed_dqs clock domain to emi_clk domain.

The Write Data Path synchronizes write data and write data mask (the dm) from emi_clk domain to the dqs_out domain.

The Digital DLL controls the delay values of read delay-line and write delay-line. It can be bypassed and switched to manual delay control mode. In manual delay control mode, the delay values of read/write delay-line can be programmed separately into the control registers.

14.7.5 Read Data Capture

When reading, DDR (dual data rate) devices send a data strobe (DQS) signal coincident with the read data. The edges of this DQS strobe are aligned (edge-aligned) with the data output by the DDR devices.

To latch read data into the EMI read data buffer, it is required that the latch clock edge is center-aligned with the data. Thus, a delayed version of the DQS strobe signal must be used to capture the data. Because the frequency of the DQS strobe signal is matched to the emi_clk, the delay is a relative number based on the period of the emi_clk. In the example shown below, the delay is set to approximately 25% of the emi_clk cycle.

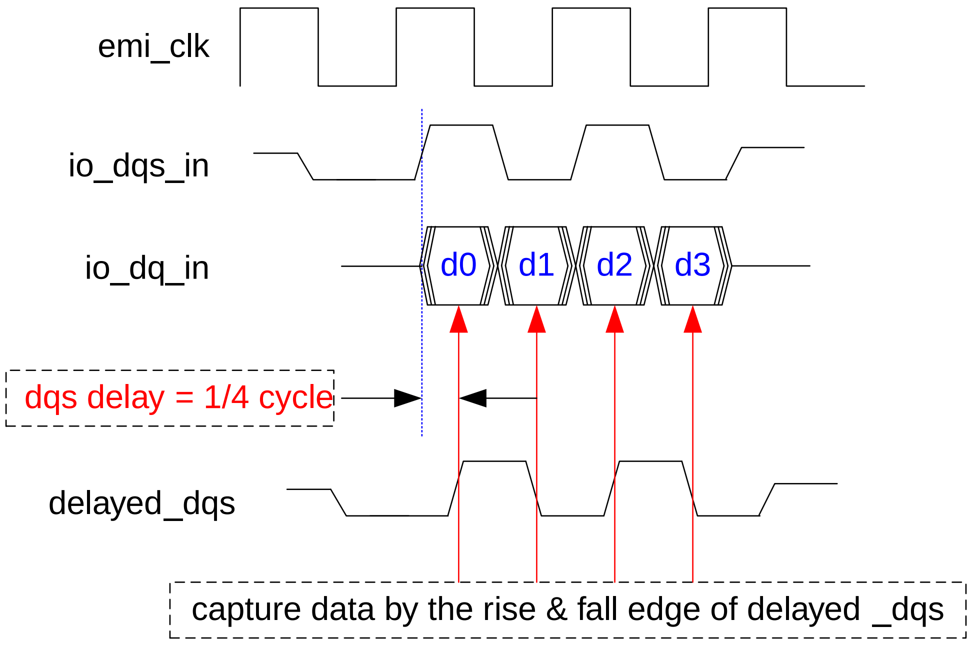


Figure 14-10. Read Data Capture

14.7.6 Synchronize Read Data From delayed_dqs to emi_clk Domain

Read data is captured into data buffers by the delayed_dqs. It need to be synchronized to the emi_clk domain, then returned to EMI core logic.

The following figure illustrates how the read data from an external DRAM device was captured and synchronized to EMI core logic.

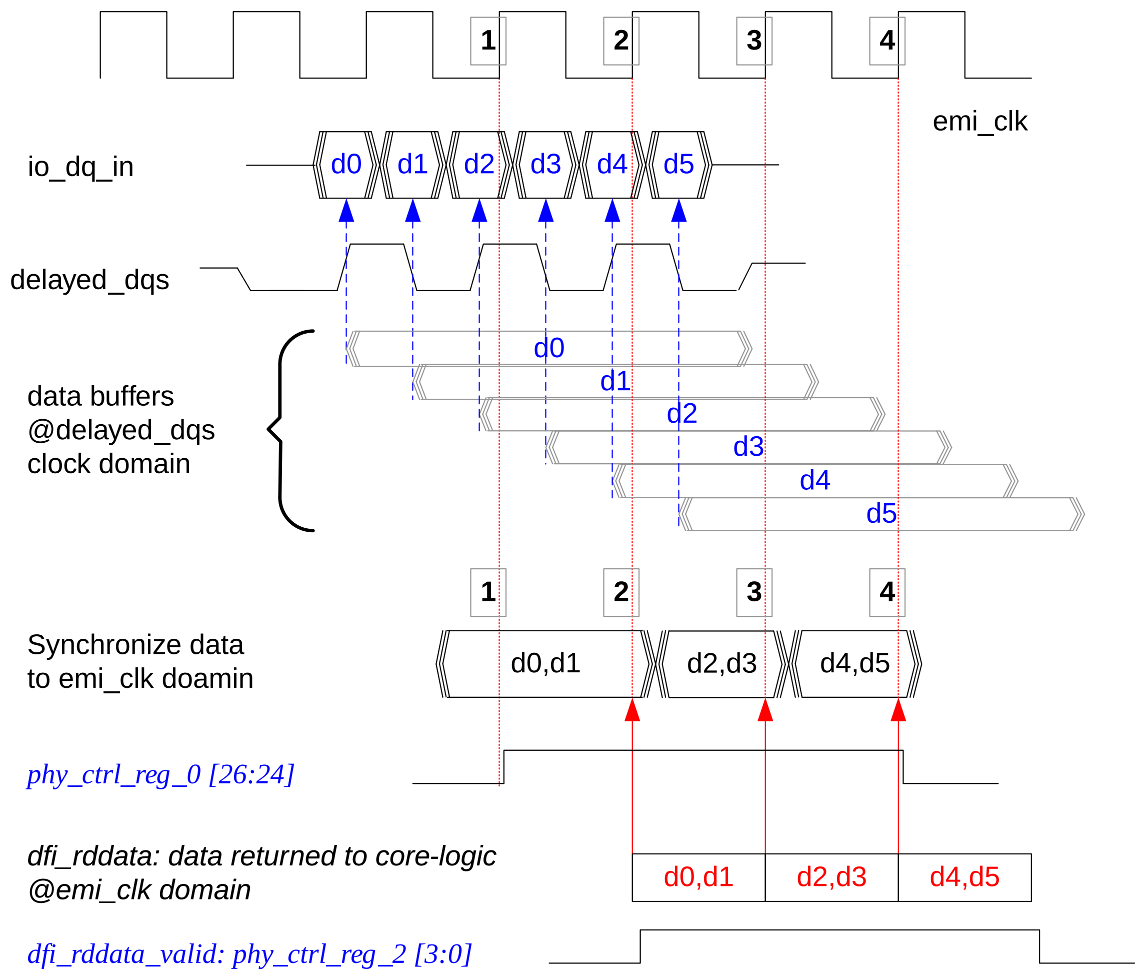


Figure 14-11. Synchronized Read Data

io_dq_in and io_dqs_in are not aligned with emi_clk in phase. Many factors affect the phase of io_dq/dqs_in, such as voltage, temperature, board layout, manufacturing process, and so on.

1. Must capture the io_dq_in by delayed_dqs to meet the critical timing requirement in high frequency.
2. The data path width @emi_clk domain is twice the data path width @delayed_dqs domain.
3. The EMI core logic fetch read data by the rise-edge of emi_clk.
4. The EMI core logic fetch read data by two important signals: dfi_rddata and dfi_rddata_valid.
5. The user can program the timing position of dfi_rddata and dfi_rddata_valid by programmable registers phy_ctrl_reg_0[26:24] and phy_ctrl_reg_2[3:0]. The unit is one cycle of emi_clk.

6. In this example, read data d0,d1 getting valid before edge number 1 of emi_clk and is synchronized at edge 2 of emi_clk. Both the setup and hold time requirement are met which means that the read data could be fetched by core logic safely.

Alternatively, read data can also be returned to core logic at edge number 1 of emi_clk, a cycle prior to the example. In this scenario, you need to set the value of register phy_ctrl_reg_0[26:24] and phy_ctrl_reg_2[3:0] to one number less. Then, the dfi_rddata and dfi_rddata_valid would become valid one cycle earlier.

NOTE

The benefit is that the read latency is shortened by one cycle, which helps increase the system performance. But, there's a timing risk on the setup time. If the io_dq/dqs_in comes a little late, a setup time violation can occur and the unexpected data is returned to core logic, which can cause a system crash.

14.7.7 Write Data Path

The following figure illustrates the write data path.

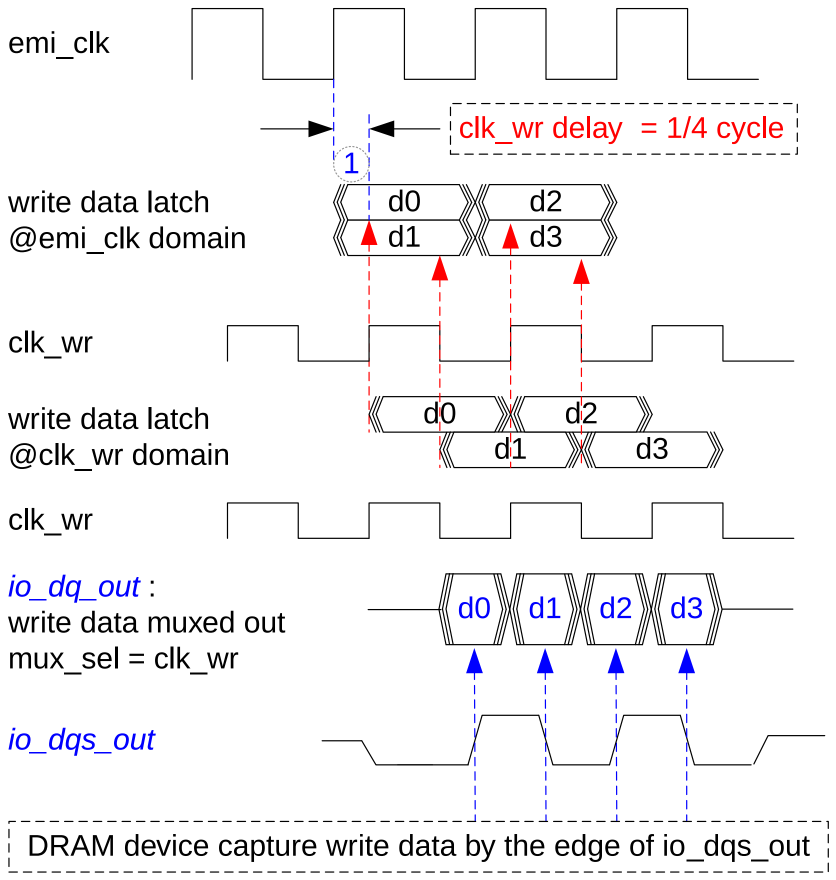


Figure 14-12. Write Data Path

NOTE

The marker 1 emphasizes that the setup time here for data d0 is only 1/4 cycle, that would be critical in timing. Write Data Path Low-Latency Option provides more details.

The io_dqs_out and the emi_clk are edge-aligned in this EMI design. This help to meet the tDQSS timing requirement defined by DRAM device.

The io_dqs_out and the io_dq_out are center-aligned which is required by the DRAM device.

14.7.8 Write Data Path Low-Latency Option

In the example of Figure 14-12, note the marker 1. It emphasizes that the setup time is only 1/4 cycle for data d0, which is critical in timing.

This DDR_PHY provides two write data path options for the user :

- The Standard Latency option.

  An extra latch at emi_clk domain is asserted into the write data path, right ahead of the latch at clk_wr domain (at the place of marker 1 in Figure 14-12). By this means, the asserted-latch and the latch at clk_wr are put back-to-back. The total delay in write data path is 1 + 1/4 cycle.

  It is safe in timing, but decreases performance by adding one cycle of latency.

- The Low-Latency option.

  The asserted latch for standard-latency is bypassed. The total delay in write data path is 1/4 cycle.

  There is risk in the timing of the write data path, but it has higher performance than the standard-latency case.

14.7.9 Digital DLL and the Delay-Line

Due to the asynchronous nature of the DRAM devices, the timing requirements for capturing and receiving data between the i.MX28 and the DRAM devices must be addressed. This EMI contains a circuit that, in conjunction with I/O cell circuitry, can be used to meet the timing requirements for DRAM devices. The delay compensation circuit was designed with the following features:

- Programmable read strobe delay specified as a percentage of a clock cycle.

- Programmable write data delays specified as percentages of a clock cycle.
- Delay compensation circuit re-sync circuitry activated during refresh cycles to compensate for temperature and voltage drift.
- Separate delay chains for each read DQS signal from the DRAM devices.

The delay compensation circuitry relies on a master/slave approach. There is a master delay line which is used to determine how many delay elements constitute a complete cycle. This count is used, along with the programmable fractional delay settings, to determine the actual number of delay elements to program into the slave delay lines. The master and slave delay lines are identical. This approach allows the memory controller to observe a clock and then delay other signals a fixed percentage of that clock. The DLL logic does not actively generate clock signals.

The actual delay element for the delay lines is user-selectable.

The following figure shows the block diagram for a digital DLL.

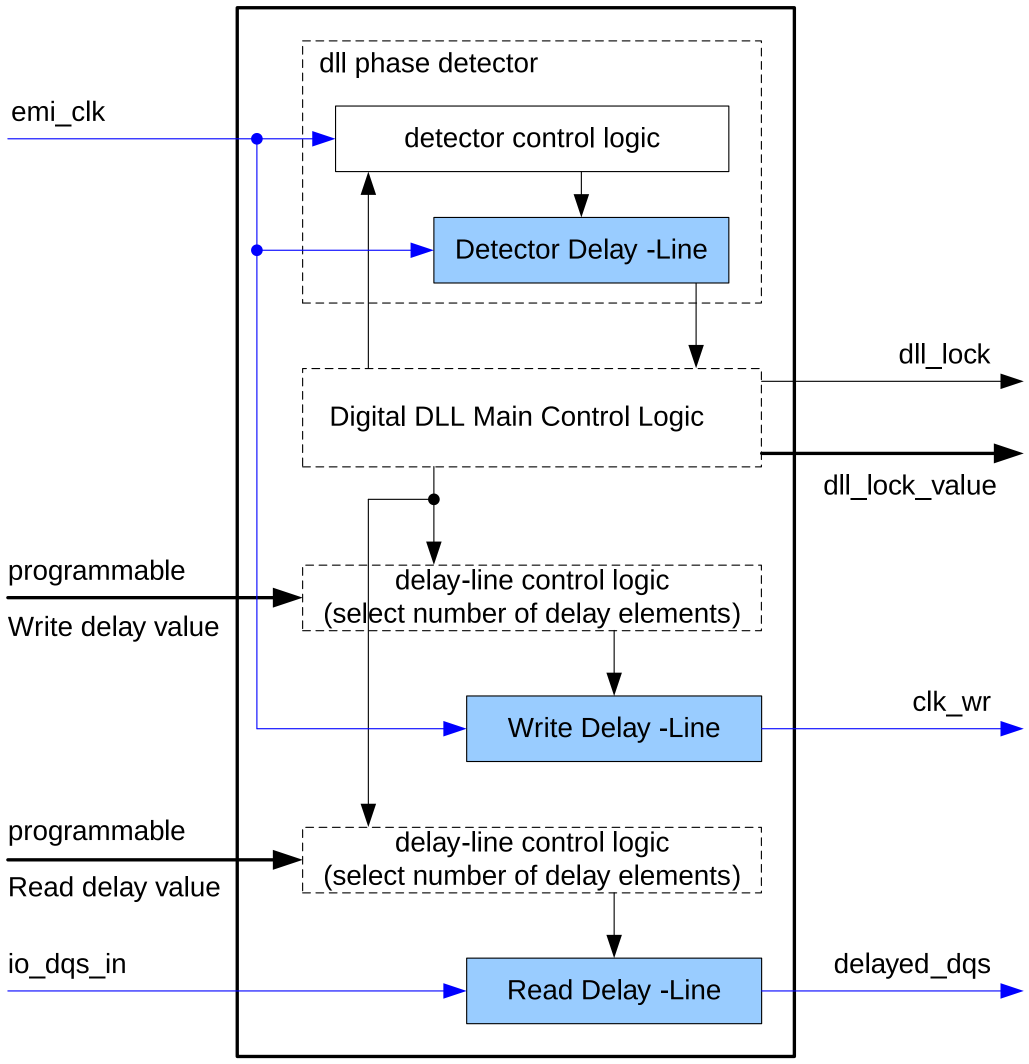


Figure 14-13. Digital DLL

The delay-line is comprised of 137 tiny delay-elements. The delay value of each delay element is almost the same. And the number of delay elements which is used to form a delay result is configurable.

There are two modes for the delay-line control :

- Auto configure mode

  This process begins by the “phase-detector”. Once the phase-detector successfully complete the detection, it will raise the “dll_lock” signal and output “dll_lock_value” which indicates the delay of cone cycle.

  The number of elements that are needed to capture an entire clock cycle is then converted into an unsigned integer named encoder [7:0]. This integer is used as the dividend for the read and write delay parameters. The actual delay setting for the delay lines is calculated by multiplying the encoder [7:0] integer by the parameter settings for each delay line and then dividing by 128 and rounding. These values are then encoded into a one-hot counter and updated at initialization and at every refresh interval.

- Bypass mode, or manual configure mode

  The phase-detector can be bypassed. The number of delay elements for read and write delay-line is programmed manually.

NOTE

In case the number of delay elements is fixed, the actual delay value (in nano-second) of a delay-line would change according to the environmental conditions, such as voltage, temperature, and so on.

Using the “auto configure mode” help to eliminate the change of delay value because the reference delay itself also changes along with environmental conditions.

14.7.10 Configure output enable of I/O Control

Some EMI I/Os require a control signal named "output enable". The I/O can drive signals out only when the "output enable" signal is active. Refer to the figure below.

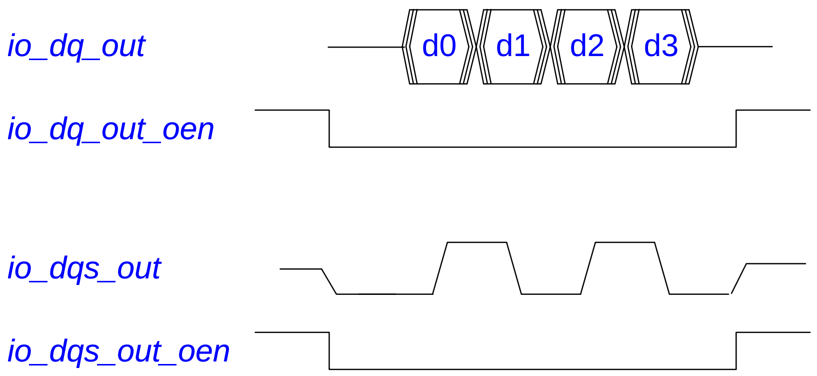


Figure 14-14. EMI I/O Output Enable

There are three kinds of output enable signals in this EMI design:

- io_dq_out_oen

  output enable for write data.

- io_dqm_oen

  output enable for write data mask. Please configure it exactly the same as io_dq_out_oen.

- io_dqs_oen

output enable for write data strobe.

The suffix "_oen" means the output enable signal is active low.

The start time and the end time of a output enable signal can be configured separately.

The start time and the end time can be adjusted by 1/4 clock cycle in unit.

To meet the timing requirement of the I/O circuits, the start time of each output enable must be prior to it's corresponding output signal, and, the end time must be later than it's corresponding signal. In another word, the output enable signal must be "wider" than the output signal. The margin at start time is named "pre-amble" and the margin at end time is named "post-amble". In the example of Figure 14-14, pre-amble = 1/2 cycle while post-amble = 1/4 cycle.

14.8 Programmable Registers

EMI Register Format Summary

HW_DRAM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_0000	DRAM Control Register 00 (HW_DRAM_CTL00)	32	R/W	0000_0000h	14.8.1/1111
800E_0004	AXI Monitor Control (HW_DRAM_CTL01)	32	R/W	0000_0000h	14.8.2/1112
800E_0008	DRAM Control Register 02 (HW_DRAM_CTL02)	32	R/W	0000_0000h	14.8.3/1113
800E_000C	DRAM Control Register 03 (HW_DRAM_CTL03)	32	R/W	0000_0000h	14.8.4/1114
800E_0010	DRAM Control Register 04 (HW_DRAM_CTL04)	32	R/W	0000_0000h	14.8.5/1114
800E_0014	DRAM Control Register 05 (HW_DRAM_CTL05)	32	R/W	0000_0000h	14.8.6/1115
800E_0018	DRAM Control Register 06 (HW_DRAM_CTL06)	32	R/W	0000_0000h	14.8.7/1115
800E_001C	DRAM Control Register 07 (HW_DRAM_CTL07)	32	R/W	0000_0000h	14.8.8/1115
800E_0020	DRAM Control Register 08 (HW_DRAM_CTL08)	32	R	0000_0010h	14.8.9/1116
800E_0024	DRAM Control Register 09 (HW_DRAM_CTL09)	32	R	0000_0000h	14.8.10/1117
800E_0028	AXI0 Debug 0 (HW_DRAM_CTL10)	32	R	0000_0000h	14.8.11/1118
800E_002C	AXI0 Debug 1 (HW_DRAM_CTL11)	32	R	0000_0000h	14.8.12/1118
800E_0030	AXI1 Debug 0 (HW_DRAM_CTL12)	32	R	0000_0000h	14.8.13/1119
800E_0034	AXI1 Debug 1 (HW_DRAM_CTL13)	32	R	0000_0000h	14.8.14/1119
800E_0038	AXI2 Debug 0 (HW_DRAM_CTL14)	32	R	0000_0000h	14.8.15/1120
800E_003C	AXI2 Debug 1 (HW_DRAM_CTL15)	32	R	0000_0000h	14.8.16/1120
800E_0040	DRAM Control Register 16 (HW_DRAM_CTL16)	32	R/W	0000_0000h	14.8.17/1121
800E_0044	DRAM Control Register 17 (HW_DRAM_CTL17)	32	R/W	0000_0000h	14.8.18/1122

HW_DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_0054	DRAM Control Register 21 (HW_DRAM_CTL21)	32	R/W	0000_0000h	14.8.19/1124
800E_0058	DRAM Control Register 22 (HW_DRAM_CTL22)	32	R/W	0000_0000h	14.8.20/1125
800E_005C	DRAM Control Register 23 (HW_DRAM_CTL23)	32	R/W	0000_0000h	14.8.21/1127
800E_0060	DRAM Control Register 24 (HW_DRAM_CTL24)	32	R/W	0000_0000h	14.8.22/1127
800E_0064	DRAM Control Register 25 (HW_DRAM_CTL25)	32	R/W	0000_0000h	14.8.23/1128
800E_0068	DRAM Control Register 26 (HW_DRAM_CTL26)	32	R/W	0000_0000h	14.8.24/1128
800E_006C	DRAM Control Register 27 (HW_DRAM_CTL27)	32	R/W	0000_0000h	14.8.25/1130
800E_0070	DRAM Control Register 28 (HW_DRAM_CTL28)	32	R/W	0000_0000h	14.8.26/1131
800E_0074	DRAM Control Register 29 (HW_DRAM_CTL29)	32	R/W	0000_0000h	14.8.27/1132
800E_0078	DRAM Control Register 30 (HW_DRAM_CTL30)	32	R	0004_0F0Ch	14.8.28/1133
800E_007C	DRAM Control Register 31 (HW_DRAM_CTL31)	32	R/W	0000_0000h	14.8.29/1134
800E_0080	DRAM Control Register 32 (HW_DRAM_CTL32)	32	R/W	0000_0000h	14.8.30/1135
800E_0084	DRAM Control Register 33 (HW_DRAM_CTL33)	32	R/W	0000_0000h	14.8.31/1136
800E_0088	DRAM Control Register 34 (HW_DRAM_CTL34)	32	R/W	0000_0000h	14.8.32/1137
800E_008C	DRAM Control Register 35 (HW_DRAM_CTL35)	32	R/W	0000_0000h	14.8.33/1139
800E_0090	DRAM Control Register 36 (HW_DRAM_CTL36)	32	R/W	0000_0000h	14.8.34/1140
800E_0094	DRAM Control Register 37 (HW_DRAM_CTL37)	32	R/W	0000_0000h	14.8.35/1141
800E_0098	DRAM Control Register 38 (HW_DRAM_CTL38)	32	R/W	0000_0000h	14.8.36/1143
800E_009C	DRAM Control Register 39 (HW_DRAM_CTL39)	32	R/W	0000_0000h	14.8.37/1144
800E_00A0	DRAM Control Register 40 (HW_DRAM_CTL40)	32	R/W	0000_0000h	14.8.38/1144
800E_00A4	DRAM Control Register 41 (HW_DRAM_CTL41)	32	R/W	0000_0000h	14.8.39/1145
800E_00A8	DRAM Control Register 42 (HW_DRAM_CTL42)	32	R/W	0000_0000h	14.8.40/1146
800E_00AC	DRAM Control Register 43 (HW_DRAM_CTL43)	32	R/W	0000_0000h	14.8.41/1146
800E_00B0	DRAM Control Register 44 (HW_DRAM_CTL44)	32	R/W	0000_0000h	14.8.42/1147
800E_00B4	DRAM Control Register 45 (HW_DRAM_CTL45)	32	R/W	0000_0000h	14.8.43/1148
800E_00C0	DRAM Control Register 48 (HW_DRAM_CTL48)	32	R/W	0000_0000h	14.8.44/1148
800E_00C4	DRAM Control Register 49 (HW_DRAM_CTL49)	32	R/W	0000_0000h	14.8.45/1149
800E_00C8	DRAM Control Register 50 (HW_DRAM_CTL50)	32	R/W	0000_0000h	14.8.46/1150
800E_00CC	DRAM Control Register 51 (HW_DRAM_CTL51)	32	R/W	0000_0000h	14.8.47/1151
800E_00D0	DRAM Control Register 52 (HW_DRAM_CTL52)	32	R/W	0000_0000h	14.8.48/1151
800E_00D4	DRAM Control Register 53 (HW_DRAM_CTL53)	32	R/W	0000_0000h	14.8.49/1152
800E_00D8	DRAM Control Register 54 (HW_DRAM_CTL54)	32	R/W	0000_0000h	14.8.50/1153
800E_00DC	DRAM Control Register 55 (HW_DRAM_CTL55)	32	R/W	0000_0000h	14.8.51/1154
800E_00E0	DRAM Control Register 56 (HW_DRAM_CTL56)	32	R/W	0000_0000h	14.8.52/1155

HW_DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_00E8	DRAM Control Register 58 (HW_DRAM_CTL58)	32	R/W	0000_0000h	14.8.53/1155
800E_00EC	DRAM Control Register 59 (HW_DRAM_CTL59)	32	R	0000_0000h	14.8.54/1156
800E_00F0	DRAM Control Register 60 (HW_DRAM_CTL60)	32	R	0000_0000h	14.8.55/1157
800E_00F4	DRAM Control Register 61 (HW_DRAM_CTL61)	32	R	0000_0000h	14.8.56/1158
800E_00F8	DRAM Control Register 62 (HW_DRAM_CTL62)	32	R	0000_0000h	14.8.57/1158
800E_00FC	DRAM Control Register 63 (HW_DRAM_CTL63)	32	R	0000_0000h	14.8.58/1159
800E_0100	DRAM Control Register 64 (HW_DRAM_CTL64)	32	R	0000_0000h	14.8.59/1160
800E_0104	DRAM Control Register 65 (HW_DRAM_CTL65)	32	R	0000_0000h	14.8.60/1161
800E_0108	DRAM Control Register 66 (HW_DRAM_CTL66)	32	R/W	0004_0000h	14.8.61/1162
800E_010C	DRAM Control Register 67 (HW_DRAM_CTL67)	32	R/W	0000_0000h	14.8.62/1162
800E_0110	DRAM Control Register 68 (HW_DRAM_CTL68)	32	R/W	0000_0000h	14.8.63/1163
800E_0114	DRAM Control Register 69 (HW_DRAM_CTL69)	32	R/W	0000_0000h	14.8.64/1164
800E_0118	DRAM Control Register 70 (HW_DRAM_CTL70)	32	R/W	0000_0000h	14.8.65/1165
800E_011C	DRAM Control Register 71 (HW_DRAM_CTL71)	32	R/W	0000_0000h	14.8.66/1167
800E_0120	DRAM Control Register 72 (HW_DRAM_CTL72)	32	R/W	0000_0000h	14.8.67/1168
800E_0124	DRAM Control Register 73 (HW_DRAM_CTL73)	32	R/W	0000_0000h	14.8.68/1169
800E_0128	DRAM Control Register 74 (HW_DRAM_CTL74)	32	R/W	0000_0000h	14.8.69/1169
800E_012C	DRAM Control Register 75 (HW_DRAM_CTL75)	32	R/W	0000_0000h	14.8.70/1170
800E_0130	DRAM Control Register 76 (HW_DRAM_CTL76)	32	R/W	0000_0000h	14.8.71/1171
800E_0134	DRAM Control Register 77 (HW_DRAM_CTL77)	32	R/W	0000_0000h	14.8.72/1171
800E_0138	DRAM Control Register 78 (HW_DRAM_CTL78)	32	R/W	0000_0000h	14.8.73/1172
800E_013C	DRAM Control Register 79 (HW_DRAM_CTL79)	32	R/W	0000_0000h	14.8.74/1172
800E_0140	DRAM Control Register 80 (HW_DRAM_CTL80)	32	R/W	0000_0000h	14.8.75/1173
800E_0144	DRAM Control Register 81 (HW_DRAM_CTL81)	32	R/W	0000_0000h	14.8.76/1174
800E_0148	DRAM Control Register 82 (HW_DRAM_CTL82)	32	R/W	0000_0000h	14.8.77/1174
800E_014C	DRAM Control Register 83 (HW_DRAM_CTL83)	32	R/W	0000_0000h	14.8.78/1175
800E_0150	DRAM Control Register 84 (HW_DRAM_CTL84)	32	R/W	0000_0000h	14.8.79/1177
800E_0154	DRAM Control Register 85 (HW_DRAM_CTL85)	32	R/W	0000_0000h	14.8.80/1179
800E_0158	DRAM Control Register 86 (HW_DRAM_CTL86)	32	R	0000_2040h	14.8.81/1180
800E_015C	DRAM Control Register 87 (HW_DRAM_CTL87)	32	R/W	0000_0000h	14.8.82/1180
800E_0160	DRAM Control Register 88 (HW_DRAM_CTL88)	32	R/W	0000_0000h	14.8.83/1181
800E_0164	DRAM Control Register 89 (HW_DRAM_CTL89)	32	R/W	0000_0000h	14.8.84/1182
800E_0168	DRAM Control Register 90 (HW_DRAM_CTL90)	32	R/W	0000_0000h	14.8.85/1182
800E_016C	DRAM Control Register 91 (HW_DRAM_CTL91)	32	R/W	0000_0000h	14.8.86/1183

HW_DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_0170	DRAM Control Register 92 (HW_DRAM_CTL92)	32	R/W	0000_0000h	14.8.87/1184
800E_0174	DRAM Control Register 93 (HW_DRAM_CTL93)	32	R/W	0000_0000h	14.8.88/1184
800E_0178	DRAM Control Register 94 (HW_DRAM_CTL94)	32	R/W	0000_0000h	14.8.89/1185
800E_017C	DRAM Control Register 95 (HW_DRAM_CTL95)	32	R	0000_0000h	14.8.90/1185
800E_0180	DRAM Control Register 96 (HW_DRAM_CTL96)	32	R	0000_0000h	14.8.91/1186
800E_0184	DRAM Control Register 97 (HW_DRAM_CTL97)	32	R	0000_0000h	14.8.92/1186
800E_0188	DRAM Control Register 98 (HW_DRAM_CTL98)	32	R	0000_0000h	14.8.93/1187
800E_018C	DRAM Control Register 99 (HW_DRAM_CTL99)	32	R	0000_0000h	14.8.94/1188
800E_0190	DRAM Control Register 100 (HW_DRAM_CTL100)	32	R	0000_0000h	14.8.95/1188
800E_0194	DRAM Control Register 101 (HW_DRAM_CTL101)	32	R	0000_0000h	14.8.96/1189
800E_0198	DRAM Control Register 102 (HW_DRAM_CTL102)	32	R	0000_0000h	14.8.97/1190
800E_019C	DRAM Control Register 103 (HW_DRAM_CTL103)	32	R	0000_0000h	14.8.98/1191
800E_01A0	DRAM Control Register 104 (HW_DRAM_CTL104)	32	R	0000_0000h	14.8.99/1191
800E_01A4	DRAM Control Register 105 (HW_DRAM_CTL105)	32	R	0000_0000h	14.8.100/1192
800E_01A8	DRAM Control Register 106 (HW_DRAM_CTL106)	32	R	0000_0000h	14.8.101/1192
800E_01AC	DRAM Control Register 107 (HW_DRAM_CTL107)	32	R	0000_0000h	14.8.102/1192
800E_01B0	DRAM Control Register 108 (HW_DRAM_CTL108)	32	R	0000_0000h	14.8.103/1193
800E_01B4	DRAM Control Register 109 (HW_DRAM_CTL109)	32	R	0000_0000h	14.8.104/1193
800E_01B8	DRAM Control Register 110 (HW_DRAM_CTL110)	32	R	0000_0000h	14.8.105/1194
800E_01BC	DRAM Control Register 111 (HW_DRAM_CTL111)	32	R	0000_0000h	14.8.106/1194
800E_01C0	DRAM Control Register 112 (HW_DRAM_CTL112)	32	R	0000_0000h	14.8.107/1195
800E_01C4	DRAM Control Register 113 (HW_DRAM_CTL113)	32	R	0000_0000h	14.8.108/1195
800E_01C8	DRAM Control Register 114 (HW_DRAM_CTL114)	32	R	0000_0000h	14.8.109/1196
800E_01CC	DRAM Control Register 115 (HW_DRAM_CTL115)	32	R	0000_0000h	14.8.110/1196
800E_01D0	DRAM Control Register 116 (HW_DRAM_CTL116)	32	R	0000_0000h	14.8.111/1197
800E_01D4	DRAM Control Register 117 (HW_DRAM_CTL117)	32	R	0000_0000h	14.8.112/1197
800E_01D8	DRAM Control Register 118 (HW_DRAM_CTL118)	32	R	0000_0000h	14.8.113/1197
800E_01DC	DRAM Control Register 119 (HW_DRAM_CTL119)	32	R	0000_0000h	14.8.114/1198
800E_01E0	DRAM Control Register 120 (HW_DRAM_CTL120)	32	R	0000_0000h	14.8.115/1198
800E_01E4	DRAM Control Register 121 (HW_DRAM_CTL121)	32	R	0000_0000h	14.8.116/1199
800E_01E8	DRAM Control Register 122 (HW_DRAM_CTL122)	32	R	0000_0000h	14.8.117/1199
800E_01EC	DRAM Control Register 123 (HW_DRAM_CTL123)	32	R	0000_0000h	14.8.118/1200
800E_01F0	DRAM Control Register 124 (HW_DRAM_CTL124)	32	R	0000_0000h	14.8.119/1200
800E_01F4	DRAM Control Register 125 (HW_DRAM_CTL125)	32	R	0000_0000h	14.8.120/1201

HW_DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_01F8	DRAM Control Register 126 (HW_DRAM_CTL126)	32	R	0000_0000h	14.8.121/1201
800E_01FC	DRAM Control Register 127 (HW_DRAM_CTL127)	32	R	0000_0000h	14.8.122/1201
800E_0200	DRAM Control Register 128 (HW_DRAM_CTL128)	32	R	0000_0000h	14.8.123/1202
800E_0204	DRAM Control Register 129 (HW_DRAM_CTL129)	32	R	0000_0000h	14.8.124/1202
800E_0208	DRAM Control Register 130 (HW_DRAM_CTL130)	32	R	0000_0000h	14.8.125/1203
800E_020C	DRAM Control Register 131 (HW_DRAM_CTL131)	32	R	0000_0000h	14.8.126/1203
800E_0210	DRAM Control Register 132 (HW_DRAM_CTL132)	32	R	0000_0000h	14.8.127/1204
800E_0214	DRAM Control Register 133 (HW_DRAM_CTL133)	32	R	0000_0000h	14.8.128/1204
800E_0218	DRAM Control Register 134 (HW_DRAM_CTL134)	32	R	0000_0000h	14.8.129/1205
800E_021C	DRAM Control Register 135 (HW_DRAM_CTL135)	32	R	0000_0000h	14.8.130/1205
800E_0220	DRAM Control Register 136 (HW_DRAM_CTL136)	32	R	0000_0000h	14.8.131/1206
800E_0224	DRAM Control Register 137 (HW_DRAM_CTL137)	32	R	0000_0000h	14.8.132/1206
800E_0228	DRAM Control Register 138 (HW_DRAM_CTL138)	32	R	0000_0000h	14.8.133/1206
800E_022C	DRAM Control Register 139 (HW_DRAM_CTL139)	32	R	0000_0000h	14.8.134/1207
800E_0230	DRAM Control Register 140 (HW_DRAM_CTL140)	32	R	0000_0000h	14.8.135/1207
800E_0234	DRAM Control Register 141 (HW_DRAM_CTL141)	32	R	0000_0000h	14.8.136/1208
800E_0238	DRAM Control Register 142 (HW_DRAM_CTL142)	32	R	0000_0000h	14.8.137/1208
800E_023C	DRAM Control Register 143 (HW_DRAM_CTL143)	32	R	0000_0000h	14.8.138/1209
800E_0240	DRAM Control Register 144 (HW_DRAM_CTL144)	32	R	0000_0000h	14.8.139/1209
800E_0244	DRAM Control Register 145 (HW_DRAM_CTL145)	32	R	0000_0000h	14.8.140/1210
800E_0248	DRAM Control Register 146 (HW_DRAM_CTL146)	32	R	0000_0000h	14.8.141/1210
800E_024C	DRAM Control Register 147 (HW_DRAM_CTL147)	32	R	0000_0000h	14.8.142/1210
800E_0250	DRAM Control Register 148 (HW_DRAM_CTL148)	32	R	0000_0000h	14.8.143/1211
800E_0254	DRAM Control Register 149 (HW_DRAM_CTL149)	32	R	0000_0000h	14.8.144/1211
800E_0258	DRAM Control Register 150 (HW_DRAM_CTL150)	32	R	0000_0000h	14.8.145/1212
800E_025C	DRAM Control Register 151 (HW_DRAM_CTL151)	32	R	0000_0000h	14.8.146/1212
800E_0260	DRAM Control Register 152 (HW_DRAM_CTL152)	32	R	0000_0000h	14.8.147/1213
800E_0264	DRAM Control Register 153 (HW_DRAM_CTL153)	32	R	0000_0000h	14.8.148/1213
800E_0268	DRAM Control Register 154 (HW_DRAM_CTL154)	32	R	0000_0000h	14.8.149/1214
800E_026C	DRAM Control Register 155 (HW_DRAM_CTL155)	32	R	0000_0000h	14.8.150/1214
800E_0270	DRAM Control Register 156 (HW_DRAM_CTL156)	32	R	0000_0000h	14.8.151/1215
800E_0274	DRAM Control Register 157 (HW_DRAM_CTL157)	32	R	0000_0000h	14.8.152/1215
800E_0278	DRAM Control Register 158 (HW_DRAM_CTL158)	32	R	0000_0000h	14.8.153/1215
800E_027C	DRAM Control Register 159 (HW_DRAM_CTL159)	32	R	0000_0000h	14.8.154/1216

HW_DRAM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800E_0280	DRAM Control Register 160 (HW_DRAM_CTL160)	32	R	0000_0000h	14.8.155/1216
800E_0284	DRAM Control Register 161 (HW_DRAM_CTL161)	32	R	0000_0000h	14.8.156/1217
800E_0288	DRAM Control Register 162 (HW_DRAM_CTL162)	32	R/W	0000_0000h	14.8.157/1217
800E_028C	DRAM Control Register 163 (HW_DRAM_CTL163)	32	R/W	0000_0000h	14.8.158/1218
800E_0290	DRAM Control Register 164 (HW_DRAM_CTL164)	32	R/W	0000_0000h	14.8.159/1219
800E_02AC	DRAM Control Register 171 (HW_DRAM_CTL171)	32	R/W	0000_0000h	14.8.160/1219
800E_02B0	DRAM Control Register 172 (HW_DRAM_CTL172)	32	R/W	0000_0000h	14.8.161/1220
800E_02B4	DRAM Control Register 173 (HW_DRAM_CTL173)	32	R/W	0000_0000h	14.8.162/1221
800E_02B8	DRAM Control Register 174 (HW_DRAM_CTL174)	32	R/W	0000_0000h	14.8.163/1222
800E_02BC	DRAM Control Register 175 (HW_DRAM_CTL175)	32	R/W	0000_0000h	14.8.164/1223
800E_02C0	DRAM Control Register 176 (HW_DRAM_CTL176)	32	R/W	0000_0000h	14.8.165/1224
800E_02C4	DRAM Control Register 177 (HW_DRAM_CTL177)	32	R/W	0000_0000h	14.8.166/1225
800E_02C8	DRAM Control Register 178 (HW_DRAM_CTL178)	32	R/W	0000_0000h	14.8.167/1226
800E_02CC	DRAM Control Register 179 (HW_DRAM_CTL179)	32	R/W	0000_0000h	14.8.168/1227
800E_02D0	DRAM Control Register 180 (HW_DRAM_CTL180)	32	R/W	0000_0000h	14.8.169/1228
800E_02D4	DRAM Control Register 181 (HW_DRAM_CTL181)	32	R/W	0000_0000h	14.8.170/1229
800E_02D8	DRAM Control Register 182 (HW_DRAM_CTL182)	32	R/W	0000_0000h	14.8.171/1230
800E_02DC	DRAM Control Register 183 (HW_DRAM_CTL183)	32	R/W	0000_0000h	14.8.172/1231
800E_02E0	DRAM Control Register 184 (HW_DRAM_CTL184)	32	R/W	0000_0000h	14.8.173/1232
800E_02E4	DRAM Control Register 185 (HW_DRAM_CTL185)	32	R/W	0000_0000h	14.8.174/1234
800E_02E8	DRAM Control Register 186 (HW_DRAM_CTL186)	32	R/W	0000_0000h	14.8.175/1235
800E_02EC	DRAM Control Register 187 (HW_DRAM_CTL187)	32	R/W	0000_0000h	14.8.176/1236
800E_02F0	DRAM Control Register 188 (HW_DRAM_CTL188)	32	R/W	0000_0000h	14.8.177/1237
800E_02F4	DRAM Control Register 189 (HW_DRAM_CTL189)	32	R/W	0000_0000h	14.8.178/1238

14.8.1 DRAM Control Register 00 (HW_DRAM_CTL00)

This is a DRAM configuration register.

Address: HW_DRAM_CTL00 – 800E_0000h base + 0h offset = 800E_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	USER_DEF_REG_0_1[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	USER_DEF_REG_0_1[15:3]													CKE_ SELECT	SREFRESH_ ENTER	BRESP_ TIMING
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL00 field descriptions

Field	Description
31–3 USER_DEF_ REG_0_1	User-defined output register 0. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)
2 CKE_SELECT	This bit selects which output logic drives the CKE pin. This is required since the initial state of CKE is low for DDR2 and high for LPDDR. This bit MUST be set appropriately before (NOT concurrently) the EMI clock is enabled by setting field EMI_CLK_DLL_ENABLE. 0x0 **DDR2** — CKE will be low before the EMI clock is enabled and the initialization sequence commences. 0x1 **LPDDR** — CKE will be high before the EMI clock is enabled and the initialization sequence commences.
1 SREFRESH_ ENTER	Initiates a self-refresh to the DRAMs. This pin updates the srefresh parameter.
0 BRESP_TIMING	This bit changes when the BRESP is issued over the AXI bus interface for bufferable AXI write transactions. 0x0 **BUFFERABLE** — BRESP is returned when command and data are received by the AXI port. 0x1 **SEMI_BUFFERABLE** — BRESP is returned when command is accepted into the internal EMI command queue.

14.8.2 AXI Monitor Control (HW_DRAM_CTL01)

This is a DRAM configuration register.

Address: HW_DRAM_CTL01 – 800E_0000h base + 4h offset = 800E_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	USER_DEF_REG_1[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_1[15:9]							MON_DBG_ STB	SLVERR				MON_DISABLE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL01 field descriptions

Field	Description
31–9 USER_DEF_ REG_1	User-defined output register 1. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)
8 MON_DBG_STB	Setting this bit to a logic 1 will initiate sampling the state of each AXI interface monitor. The diagnostic information available in the debug registers is valid only after this bit is set to a logic one. To take a subsequent snapshot of the state of each AXI monitor, set this bit low, and then high again.
7–4 SLVERR	Setting this bit will cause all AXI transactions to be processed with a slave error when the respective monitor is enabled.
3–0 MON_DISABLE	Setting this bit will disable the AXI monitors. None of the EMI AXI error conditions will be checked and all AXI traffic will proceed.

14.8.3 DRAM Control Register 02 (HW_DRAM_CTL02)

This is a DRAM configuration register.

Address: HW_DRAM_CTL02 – 800E_0000h base + 8h offset = 800E_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_2																															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL02 field descriptions

Field	Description
31–0 USER_DEF_ REG_2	User-defined output register 2. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.4 DRAM Control Register 03 (HW_DRAM_CTL03)

This is a DRAM configuration register.

Address: HW_DRAM_CTL03 – 800E_0000h base + Ch offset = 800E_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL03 field descriptions

Field	Description
31–0 USER_DEF_ REG_3	User-defined output register 3. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.5 DRAM Control Register 04 (HW_DRAM_CTL04)

This is a DRAM configuration register.

Address: HW_DRAM_CTL04 – 800E_0000h base + 10h offset = 800E_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL04 field descriptions

Field	Description
31–0 USER_DEF_ REG_4	User-defined output register 4. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.6 DRAM Control Register 05 (HW_DRAM_CTL05)

This is a DRAM configuration register.

Address: HW_DRAM_CTL05 – 800E_0000h base + 14h offset = 800E_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_5																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL05 field descriptions

Field	Description
31–0 USER_DEF_ REG_5	User-defined output register 5. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.7 DRAM Control Register 06 (HW_DRAM_CTL06)

This is a DRAM configuration register.

Address: HW_DRAM_CTL06 – 800E_0000h base + 18h offset = 800E_0018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_6																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL06 field descriptions

Field	Description
31–0 USER_DEF_ REG_6	User-defined output register 6. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.8 DRAM Control Register 07 (HW_DRAM_CTL07)

This is a DRAM configuration register.

Address: HW_DRAM_CTL07 – 800E_0000h base + 1Ch offset = 800E_001Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USER_DEF_REG_7																															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL07 field descriptions

Field	Description
31–0 USER_DEF_ REG_7	User-defined output register 7. Holds user-defined values that will be available as output signals param_user_def_reg_X (where X ranges from 0 to 7)

14.8.9 DRAM Control Register 08 (HW_DRAM_CTL08)

This is a DRAM configuration register.

Address: HW_DRAM_CTL08 – 800E_0000h base + 20h offset = 800E_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USER_DEF_REG_RO_0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USER_DEF_REG_RO_0[15:9]							CONTROLLER_ BUSY	REFRESH_IN_ PROCESS	Q_ALMOST_ FULL	SREFRESH_ACK	CKE_STATUS	COMMAND_ACCEPTED			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

HW_DRAM_CTL08 field descriptions

Field	Description
31–9 USER_DEF_ REG_RO_0	User-defined input register 0. Holds the value driven to the MC by the PHY on the signals param_user_def_reg_ro_X (where X ranges from 0 to 7) at the EMI core level. There are a total of 8 user-defined input registers.
8 CONTROLLER_ BUSY	Status signal from the EMI. This will only be low when the EMI is not reading data, writing data or processing a command.
7 REFRESH_IN_ PROCESS	Active-high signal that indicates that the EMI is executing a refresh command. This signals is asserted when a refresh command is sent t othe DRAM devices and the remains asserted until the refresh command has completed.
6 Q_ALMOST_ FULL	Indicates that the queue has reached the value set in the q_fullness parameter in the controller.
5 SREFRESH_ ACK	Acknowledge signal that indicates that the memory devices are in self-refresh mode. This signal will only be asserted if the DRAMs have been placed into self-refresh mode through the assertion of the srefresh_enter signal and if the signal is still held high until the memory enters self-refresh mode.
4 CKE_STATUS	Indicates the memory devices are in either their self-refresh or power-down mode. This signals is the status of the control_cke signal inside the EMI, but may be delayed through the cke_delay paramter in the contoller to reflect the inverted CKE status on the memory bus. NOTE: this parameter is inverted with respect to the logic state on the external CKE pin.
3–0 COMMAND_ ACCEPTED	Indicates when bus interface commands are accepted.

14.8.10 DRAM Control Register 09 (HW_DRAM_CTL09)

This is a DRAM configuration register.

Address: HW_DRAM_CTL09 – 800E_0000h base + 24h offset = 800E_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USER_DEF_REG_RO_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL09 field descriptions

Field	Description
31–0 USER_DEF_ REG_RO_1	User-defined input register 1. Holds the value driven to the MC by the PHY on the signals param_user_def_reg_ro_X (where X ranges from 0 to 7) at the EMI core level. There are a total of 8 user-defined input registers.

14.8.11 AXI0 Debug 0 (HW_DRAM_CTL10)

AXI0 Monitor Debug register 0.

Address: HW_DRAM_CTL10 – 800E_0000h base + 28h offset = 800E_0028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD								READ_CNT								WRESP_CNT								WDATA_CNT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL10 field descriptions

Field	Description
31–24 RSVD	Reserved.
23–16 READ_CNT	This field will indicate how many read commands have been accepted in which their respective data beats have not been returned over the data bus.
15–8 WRESP_CNT	This field will indicate how many write commands have been accepted in which their respective RESP signaling has not completed.
7–0 WDATA_CNT	This field will indicate how many write commands have been accepted in which their respective data beats have not been transferred over the data bus.

14.8.12 AXI0 Debug 1 (HW_DRAM_CTL11)

AXI0 Monitor Debug register 1.

Address: HW_DRAM_CTL11 – 800E_0000h base + 2Ch offset = 800E_002Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSTATE								RSTATE								RLEN								WLEN							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL11 field descriptions

Field	Description
31–24 WSTATE	This reflects the monitor write machine's state.
23–16 RSTATE	This reflects the monitor read machine's state.
15–8 RLEN	This reflects the read length beat counter.

HW_DRAM_CTL11 field descriptions (continued)

Field	Description
7–0 WLEN	This reflects the write length beat counter.

14.8.13 AXI1 Debug 0 (HW_DRAM_CTL12)

AXI1 Monitor Debug register 0.

Address: HW_DRAM_CTL12 – 800E_0000h base + 30h offset = 800E_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD								READ_CNT								WRESP_CNT								WDATA_CNT							
W	▨								▨								▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL12 field descriptions

Field	Description
31–24 RSVD	Reserved.
23–16 READ_CNT	This field will indicate how many read commands have been accepted in which their respective data beats have not been returned over the data bus.
15–8 WRESP_CNT	This field will indicate how many write commands have been accepted in which their respective RESP signaling has not completed.
7–0 WDATA_CNT	This field will indicate how many write commands have been accepted in which their respective data beats have not been transferred over the data bus.

14.8.14 AXI1 Debug 1 (HW_DRAM_CTL13)

AXI1 Monitor Debug register 1.

Address: HW_DRAM_CTL13 – 800E_0000h base + 34h offset = 800E_0034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSTATE								RSTATE								RLEN								WLEN							
W	▨								▨								▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL13 field descriptions

Field	Description
31–24 WSTATE	This reflects the monitor write machine's state.
23–16 RSTATE	This reflects the monitor read machine's state.
15–8 RLEN	This reflects the read length beat counter.
7–0 WLEN	This reflects the write length beat counter.

14.8.15 AXI2 Debug 0 (HW_DRAM_CTL14)

AXI2 Monitor Debug register 0.

Address: HW_DRAM_CTL14 – 800E_0000h base + 38h offset = 800E_0038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD								READ_CNT								WRESP_CNT								WDATA_CNT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL14 field descriptions

Field	Description
31–24 RSVD	Reserved.
23–16 READ_CNT	This field will indicate how many read commands have been accepted in which their respective data beats have not been returned over the data bus.
15–8 WRESP_CNT	This field will indicate how many write commands have been accepted in which their respective RESP signaling has not completed.
7–0 WDATA_CNT	This field will indicate how many write commands have been accepted in which their respective data beats have not been transferred over the data bus.

14.8.16 AXI2 Debug 1 (HW_DRAM_CTL15)

AXI2 Monitor Debug register 1.

Address: HW_DRAM_CTL15 – 800E_0000h base + 3Ch offset = 800E_003Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WSTATE								RSTATE								RLEN								WLEN							
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL15 field descriptions

Field	Description
31–24 WSTATE	This reflects the monitor write machine's state.
23–16 RSTATE	This reflects the monitor read machine's state.
15–8 RLEN	This reflects the read length beat counter.
7–0 WLEN	This reflects the write length beat counter.

14.8.17 DRAM Control Register 16 (HW_DRAM_CTL16)

This is a DRAM configuration register.

Address: HW_DRAM_CTL16 – 800E_0000h base + 40h offset = 800E_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3								RSVD2							POWER_DOWN
W								WRITE_MODEREG								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE								RSVD1							START
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

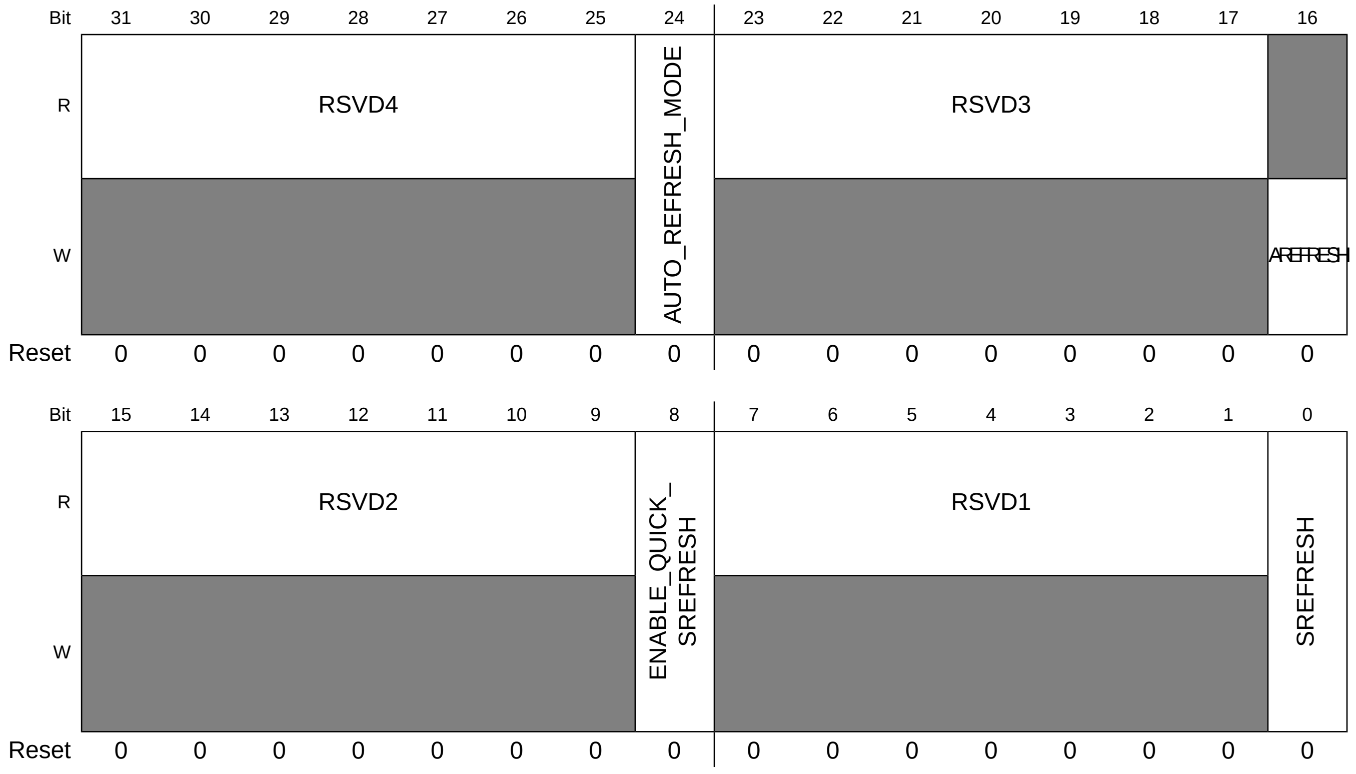
HW_DRAM_CTL16 field descriptions

Field	Description
31–25 RSVD3	Always write zeroes to this field.
24 WRITE_ MODEREG	Write mode register data to the DRAMs. WRITE-ONLY Writes mode register information into the memory devices. The user should program the appropriate mrY_data_X parameters with valid information based on the memory type being used. All of the mode registers that are relevant for the memory type specified in the dram_class parameter will be written on each write_modereg setting. This parameter will always read back as 'b0. The mode registers are automatically written at initialization of the EMI. There is no need to initiate a mode register write after setting the start parameter in the EMI unless some value in these registers needs to be changed after initialization. This parameter may not be changed when the memory is in power-down mode (when the CKE input is de-asserted).
23–17 RSVD2	Always write zeroes to this field.
16 POWER_DOWN	Disable clock enable and set DRAMs in power-down state. When this parameter is set to 'b1, the EMI will complete processing of the current burst for the current transaction (if any), issue a pre-charge all command and then disable the clock enable signal to the DRAM devices. Any subsequent commands in the command queue will be suspended until this parameter is cleared to 'b0. 'b0 = Enable full power state. 'b1 = Disable the clock enable and power down the EMI.
15–8 OBSOLETE	Always write zeroes to this field.
7–1 RSVD1	Always write zeroes to this field.
0 START	Initiate cmd processing in the controller. Prior to setting this parameter to 1'b1, the EMI will not issue any commands to the DRAM devices or respond to any signal activity except for reading and writing parameters and accepting traffic that the customer may send to the EMI internal queues. Once this parameter is set to 1'b1, the EMI will respond to inputs from the ASIC and begin to process memory access commands. Note that resetting this parameter to 1'b0 will not shut off traffic. However, cycling this parameter to 1'b0 and then resetting it to 1'b1 will reset the digital DLL to a new input clock frequency if desired. This protocol is described in detail in section Changing Input Clock Frequency. Note: Until the initialization complete bit is set in the int_status parameter and the dfi_init_complete signal is asserted from the PHY, commands will not be accepted into the core command queue. 'b0 = Controller is not in active mode. 'b1 = Initiate active mode for the EMI.

14.8.18 DRAM Control Register 17 (HW_DRAM_CTL17)

This is a DRAM configuration register.

Address: HW_DRAM_CTL17 – 800E_0000h base + 44h offset = 800E_0044h



HW_DRAM_CTL17 field descriptions

Field	Description
31–25 RSVD4	Always write zeroes to this field.
24 AUTO_ REFRESH_ MODE	Define auto-refresh to occur at next burst or next cmd boundary. Sets the mode for when the automatic refresh will occur. If the auto_refresh_mode parameter is set and a refresh is required to memory, the EMI will delay this refresh until the end of the current transaction (if the transaction is fully contained inside a single page), or until the current transaction hits the end of the current page. 'b0 = Issue refresh on the next DRAM burst boundary, even if the current command is not complete. 'b1 = Issue refresh on the next command boundary.
23–17 RSVD3	Always write zeroes to this field.
16 AREFRESH	Trigger auto-refresh at boundary specified by AUTO_REFRESH_MODE. WRITE-ONLY Initiates an automatic refresh to the DRAM devices based on the setting of the auto_refresh_mode parameter. If there are any open banks when this parameter is set, the EMI will automatically close these banks before issuing the auto-refresh command. This parameter will always read back as 0x0. 'b0 = No action 'b1 = Issue refresh to the DRAM devices
15–9 RSVD2	Always write zeroes to this field.

HW_DRAM_CTL17 field descriptions (continued)

Field	Description
8 ENABLE_ QUICK_ SREFRESH	Allow user to interrupt memory initialization to enter self-refresh mode. When this bit is set to 'b1, the memory initialization sequence may be interrupted and the memory may enter self-refresh mode. This is used to place the memory devices into self-refresh mode when a power loss is detected during the initialization process. 'b0 = Continue memory initialization. 'b1 = Interrupt memory initialization and enter self-refresh mode.
7–1 RSVD1	Always write zeroes to this field.
0 SREFRESH	Place DRAMs in self-refresh mode. When this parameter is set to 'b1, the DRAM device(s) will be placed in self-refresh mode. For this, the current burst for the current transaction (if any) will complete, all banks will be closed, the self-refresh command will be issued to the DRAM, and the clock enable signal will be de-asserted. The system will remain in self-refresh mode until this parameter is cleared to 'b0. The DRAM devices will return to normal operating mode after the self-refresh exit time (the txsr parameter) of the device and any DLL initialization time for the DRAM is reached. The EMI will resume processing of the commands from the interruption point. This parameter will be updated with an assertion of the srefresh_enter pin, regardless of the behavior on the register interface. To disable self-refresh again after a srefresh_enter pin assertion, the user will need to clear the parameter to 'b0. 'b0 = Disable self-refresh mode. 'b1 = Initiate self-refresh of the DRAM devices.

14.8.19 DRAM Control Register 21 (HW_DRAM_CTL21)

This is a DRAM configuration register.

Address: HW_DRAM_CTL21 – 800E_0000h base + 54h offset = 800E_0054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3					CKE_DELAY			DLL_LOCK							
W	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							DLLLOCKREG	RSVD1							DLL_BYPASS_MODE
W	▨							▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL21 field descriptions

Field	Description
31–27 RSVD3	Always write zeroes to this field.
26–24 CKE_DELAY	Additional cycles to delay CKE for status reporting. Sets the number of additional cycles of delay to include in the CKE signal cke_status for status reporting. The default delay is 0 cycles.
23–16 DLL_LOCK	Number of delay elements in master DLL lock. READ-ONLY Defines the actual number of delay elements used to capture one full clock cycle. This parameter is automatically updated every time a refresh operation is performed. This parameter is read-only.
15–9 RSVD2	Always write zeroes to this field.
8 DLLLOCKREG	Status of DLL lock coming out of master delay. READ-ONLY Shows status of the DLL as locked or unlocked. This parameter is read-only 'b0 = DLL is unlocked. 'b1 = DLL is locked.
7–1 RSVD1	Always write zeroes to this field.
0 DLL_BYPASS_ MODE	Enable the DLL bypass feature of the controller. This parameter has no meaning for this EMI.

14.8.20 DRAM Control Register 22 (HW_DRAM_CTL22)

This is a DRAM configuration register.

Address: HW_DRAM_CTL22 – 800E_0000h base + 58h offset = 800E_0058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE								RSVD3				LOWPOWER_ REFRESH_ ENABLE				RSVD2			LOWPOWER_ CONTROL					RSVD1			LOWPOWER_ AUTO_ENABLE				
W																																
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL22 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–20 RSVD3	Always write zeroes to this field.
19–16 LOWPOWER_ REFRESH_ ENABLE	Enable refreshes while in low power mode. Sets whether refreshes will occur while the EMI is in any of the low power modes. This parameter is active low. 'b0 = Refreshes still occur 'b1 = Refreshes do not occur
15–13 RSVD2	Always write zeroes to this field.
12–8 LOWPOWER_ CONTROL	Controls entry into the low power modes. Enables the individual low power modes of the device. Bit [4] = Controls memory power-down mode (Mode 1). Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2). Bit [2] = Controls memory self-refresh mode (Mode 3). Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4). Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). For all bits: 'b0 = Disabled. 'b1 = Enabled.
7–5 RSVD1	Always write zeroes to this field.
4–0 LOWPOWER_ AUTO_ENABLE	Enables automatic entry into the low power mode on idle. Enables automatic entry into the low power modes of the EMI. Bit [4] = Controls memory power-down mode (Mode 1). Bit [3] = Controls memory power-down with memory clock gating mode (Mode 2). Bit [2] = Controls memory self-refresh mode (Mode 3). Bit [1] = Controls memory self-refresh with memory clock gating mode (Mode 4). Bit [0] = Controls memory self-refresh with memory and controller clock gating mode (Mode 5). It is not possible to enter Mode 5 manually. Setting bit [0] of lowpower_control with bit [0] of this parameter cleared will not result in any change.

HW_DRAM_CTL22 field descriptions (continued)

Field	Description
	The user should not use both automatic and manual modes for the various low power modes. All modes should be entered automatically or all entered manually. For all bits: 'b0 = Automatic entry into this mode is disabled. The user may enter this mode manually by setting the associated lowpower_control bit. 'b1 = Automatic entry into this mode is enabled. The mode will be entered automatically when the proper counters expire, and only if the associated lowpower_control bit is set.

14.8.21 DRAM Control Register 23 (HW_DRAM_CTL23)

This is a DRAM configuration register.

Address: HW_DRAM_CTL23 – 800E_0000h base + 5Ch offset = 800E_005Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOWPOWER_INTERNAL_CNT																LOWPOWER_EXTERNAL_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL23 field descriptions

Field	Description
31–16 LOWPOWER_ INTERNAL_CNT	Counts idle cycles to self-refresh with memory and controller clk gating. Counts the number of idle cycles before memory self-refresh with memory and controller clock gating low power mode.
15–0 LOWPOWER_ EXTERNAL_CNT	Counts idle cycles to self-refresh with memory clock gating. Counts the number of idle cycles before memory self-refresh with memory clock gating low power mode.

14.8.22 DRAM Control Register 24 (HW_DRAM_CTL24)

This is a DRAM configuration register.

Address: HW_DRAM_CTL24 – 800E_0000h base + 60h offset = 800E_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOWPOWER_SELF_REFRESH_CNT																LOWPOWER_REFRESH_HOLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL24 field descriptions

Field	Description
31–16 LOWPOWER_ SELF_ REFRESH_CNT	Counts idle cycles to memory self-refresh. Counts the number of cycles to the next memory self-refresh low power mode.
15–0 LOWPOWER_ REFRESH_ HOLD	Re-Sync counter for DLL in Clock Gate Mode. Sets the number of cycles that the EMI will wait before attempting to re-lock the DLL when using the controller clock gating mode low power mode. This counter will ONLY be used in this mode, the deepest low power mode. When this counter expires, the DLL will be un-gated for at least 16 cycles during which the DLL will attempt to re-lock. After 16 cycles have elapsed and the DLL has locked, then the DLL controller clock will be gated again and the counter will reset to this value. If the DLL requires more than 16 cycles to re-lock, then the un-gated time will be longer.

14.8.23 DRAM Control Register 25 (HW_DRAM_CTL25)

This is a DRAM configuration register.

Address: HW_DRAM_CTL25 – 800E_0000h base + 64h offset = 800E_0064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																LOWPOWER_POWER_DOWN_CNT															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL25 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–0 LOWPOWER_ POWER_DOWN_ CNT	Counts idle cycles to memory power-down. Counts the number of idle cycles before memory power-down or power-down with memory clock gating low power mode.

14.8.24 DRAM Control Register 26 (HW_DRAM_CTL26)

This is a DRAM configuration register.

Address: HW_DRAM_CTL26 – 800E_0000h base + 68h offset = 800E_0068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD3							PRIORITY_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							ADDR_CMP_EN	RSVD1							PLACEMENT_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL26 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–17 RSVD3	Always write zeroes to this field.
16 PRIORITY_EN	Enable priority for cmd queue placement logic. Enables priority as a condition when using the placement logic to fill the command queue. 'b0 = Disabled 'b1 = Enabled
15–9 RSVD2	Always write zeroes to this field.
8 ADDR_CMP_EN	Enable address collision detection for cmd queue placement logic. Enables address collision/data coherency detection as a condition when using the placement logic to fill the command queue. 'b0 = Disabled 'b1 = Enabled
7–1 RSVD1	Always write zeroes to this field.
0 PLACEMENT_EN	Enable placement logic for cmd queue. Enables using the placement logic to fill the command queue. 'b0 = Placement logic is disabled. The command queue is a straight FIFO. 'b1 = Placement logic is enabled. The command queue will be filled according to the placement logic factors.

14.8.25 DRAM Control Register 27 (HW_DRAM_CTL27)

This is a DRAM configuration register.

Address: HW_DRAM_CTL27 – 800E_0000h base + 6Ch offset = 800E_006Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4							SWAP_PORT_RW_SAME_EN	RSVD3							SWAP_EN
W	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							BANK_SPLIT_EN	RSVD1							RW_SAME_EN
W	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL27 field descriptions

Field	Description
31–25 RSVD4	Always write zeroes to this field.
24 SWAP_PORT_ RW_SAME_EN	No meaning for this MC. When swapping in enabled (swap_en is set to 'b1), this parameter enables swapping between read commands on the same port from different requestors. Commands from the same requestor can not be swapped. While read data may be interleaved, write data must be accepted in order on a port. The MC logic will not swap write commands from the same port, even if this bit is enabled. This situation will be uncommon since all read commands on the same port enter the command queue at the priority defined in axiY_r_priority. However, this situation may occur through command aging. 'b0 = Disabled 'b1 = Enabled
23–17 RSVD3	Always write zeroes to this field.
16 SWAP_EN	Enable command swapping logic in execution unit. Enables swapping of the active command for a new higher-priority command when using the placement logic. 'b0 = Disabled 'b1 = Enabled
15–9 RSVD2	Always write zeroes to this field.
8 BANK_SPLIT_EN	Enable bank splitting for cmd queue placement logic. Enables bank splitting as a condition when using the placement logic to fill the command queue.

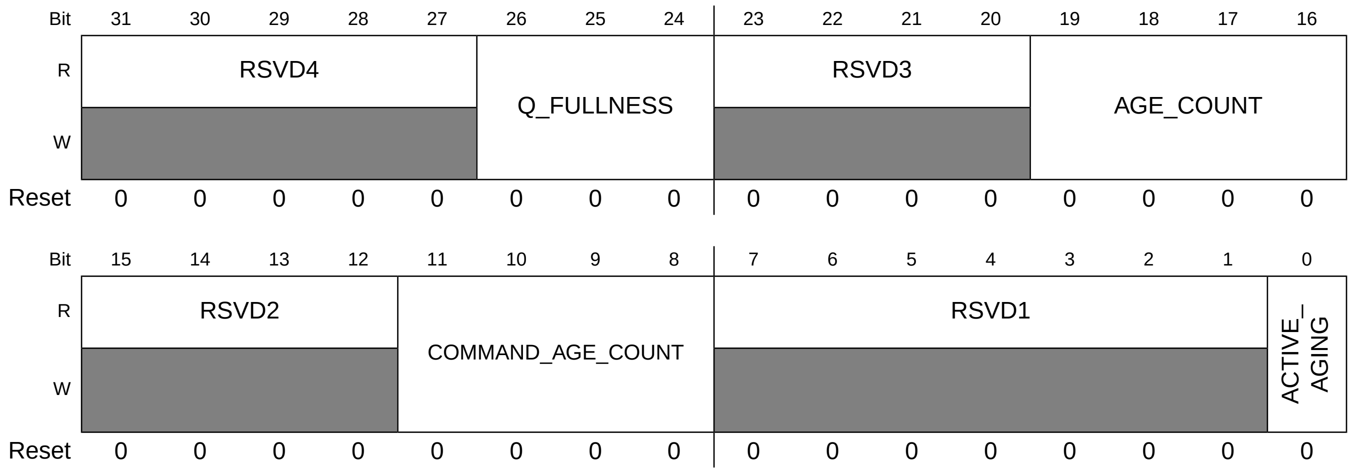
HW_DRAM_CTL27 field descriptions (continued)

Field	Description
	'b0 = Disabled 'b1 = Enabled
7–1 RSVD1	Always write zeroes to this field.
0 RW_SAME_EN	Enable read/write grouping for cmd queue placement logic. Enables read/write grouping as a condition when using the placement logic to fill the command queue. 'b0 = Disabled 'b1 = Enabled

14.8.26 DRAM Control Register 28 (HW_DRAM_CTL28)

This is a DRAM configuration register.

Address: HW_DRAM_CTL28 – 800E_0000h base + 70h offset = 800E_0070h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R W
RSVD4
Q_FULLNESS
RSVD3
AGE_COUNT
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R W
RSVD2
COMMAND_AGE_COUNT
RSVD1
ACTIVE_AGING
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_DRAM_CTL28 field descriptions

Field	Description
31–27 RSVD4	Always write zeroes to this field.
26–24 Q_FULLNESS	Quantity that determines cmd queue full. Defines quantity of data that will be considered full for the command queue.
23–20 RSVD3	Always write zeroes to this field.
19–16 AGE_COUNT	Initial value of master aging-rate counter for cmd aging. Holds the initial value of the master aging-rate counter. When using the placement logic to fill the command queue, the command aging counters will be decremented one each time the master aging-rate counter counts down age_count cycles.

HW_DRAM_CTL28 field descriptions (continued)

Field	Description
15–12 RSVD2	Always write zeroes to this field.
11–8 COMMAND_ AGE_COUNT	Initial value of individual cmd aging counters for cmd aging. Holds the initial value of the command aging counters associated with each command in the command queue. When using the placement logic to fill the command queue, the command aging counters decrement one each time the master aging-rate counter counts down the number of cycles in the age_count parameter.
7–1 RSVD1	Always write zeroes to this field.
0 ACTIVE_AGING	Enable command aging in the command queue. Enables aging of commands in the command queue when using the placement logic to fill the command queue. The total number of cycles required to decrement the priority value on a command by one is the product of the values in the age_count and command_age_count parameters. 'b0 = Disabled 'b1 = Enabled

14.8.27 DRAM Control Register 29 (HW_DRAM_CTL29)

This is a DRAM configuration register.

Address: HW_DRAM_CTL29 – 800E_0000h base + 74h offset = 800E_0074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				CS_MAP				RSVD3					COLUMN_SIZE			RSVD2					ADDR_PINS			RSVD1				APREBIT			
W	▨	▨	▨	▨					▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL29 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 CS_MAP	Defines which chip selects are active. Sets the mask that determines which chip select pins are active, with each bit representing a different chip select. The user address chip select field will be mapped into the active chip selects indicated by this parameter in ascending order from lowest to highest. This allows the EMI to map the entire contiguous user address into any group of chip selects. Bit [0] of this parameter corresponds to chip select [0], bit [1] corresponds to chip select [1], etc. The number of chip selects, the number of bits set to 1 in this parameter, must be a power of 2.
23–19 RSVD3	Always write zeroes to this field.

HW_DRAM_CTL29 field descriptions (continued)

Field	Description
18–16 COLUMN_SIZE	Difference between number of column pins available and number being used. Shows the difference between the maximum column width available (12) and the actual number of column pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter.
15–11 RSVD2	Always write zeroes to this field.
10–8 ADDR_PINS	Difference between number of addr pins available and number being used. Defines the difference between the maximum number of address pins configured (15) and the actual number of pins being used. The user address is automatically shifted so that the user address space is mapped contiguously into the memory map based on the value of this parameter.
7–4 RSVD1	Always write zeroes to this field.
3–0 APREBIT	Location of the auto pre-charge bit in the DRAM address. Defines the location of the auto pre-charge bit in the DRAM address in decimal encoding.

14.8.28 DRAM Control Register 30 (HW_DRAM_CTL30)

This is a DRAM configuration register.

Address: HW_DRAM_CTL30 – 800E_0000h base + 78h offset = 800E_0078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD3					MAX_CS_REG		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				MAX_ROW_REG				RSVD1				MAX_COL_REG			
W																
Reset	0	0	0	0	1	1	1	1	0	0	0	0	1	1	0	0

HW_DRAM_CTL30 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–19 RSVD3	Always write zeroes to this field.

HW_DRAM_CTL30 field descriptions (continued)

Field	Description
18–16 MAX_CS_REG	Maximum number of chip selects available. READ-ONLY Displays the maximum number of chip selects configured for this EMI. This parameter is read-only.
15–12 RSVD2	Always write zeroes to this field.
11–8 MAX_ROW_REG	Maximum width of memory address bus. READ-ONLY Defines the maximum width of the memory address bus for the EMI. This value can be used to set the addr_pins parameter. This parameter is read-only. addr_pins = max_row_reg - <number of row bits in memory device>.
7–4 RSVD1	Always write zeroes to this field.
3–0 MAX_COL_REG	Maximum width of column address in DRAMs. READ-ONLY Defines the maximum width of column address in the DRAM devices. This value can be used to set the column_size parameter. This parameter is read-only. column_size = max_col_reg - <number of column bits in memory device>.

14.8.29 DRAM Control Register 31 (HW_DRAM_CTL31)

This is a DRAM configuration register.

Address: HW_DRAM_CTL31 – 800E_0000h base + 7Ch offset = 800E_007Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD3							EIGHT_BANK_MODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							DRIVE_DQ_DQS	RSVD1							DQS_N_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL31 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.

HW_DRAM_CTL31 field descriptions (continued)

Field	Description
23–17 RSVD3	Always write zeroes to this field.
16 EIGHT_BANK_ MODE	Number of banks on the DRAM(s). Indicates that the memory devices have eight banks. 'b0 = Memory devices have 4 banks. 'b1 = Memory devices have 8 banks.
15–9 RSVD2	Always write zeroes to this field.
8 DRIVE_DQ_DQS	Sets DQ/DQS output enable behavior when controller is idle. Selects if the DQ output enables and DQS output enables will be driven active when the EMI is in an idle state. 'b0 = Leave the output enables in their current state when idle. 'b1 = Drive the idle_drive_enable signal high when idle.
7–1 RSVD1	Always write zeroes to this field.
0 DQS_N_EN	Set DQS pin as single-ended or differential. Enables differential data strobe signals from the DRAM. This parameter is only relevant for the DDR PHY. 'b0 = Single-ended DQS signal from the DRAM. 'b1 = Differential DQS signal from the DRAM.

14.8.30 DRAM Control Register 32 (HW_DRAM_CTL32)

This is a DRAM configuration register.

Address: HW_DRAM_CTL32 – 800E_0000h base + 80h offset = 800E_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							REDUC	RSVD1							REG_DIMM_ ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

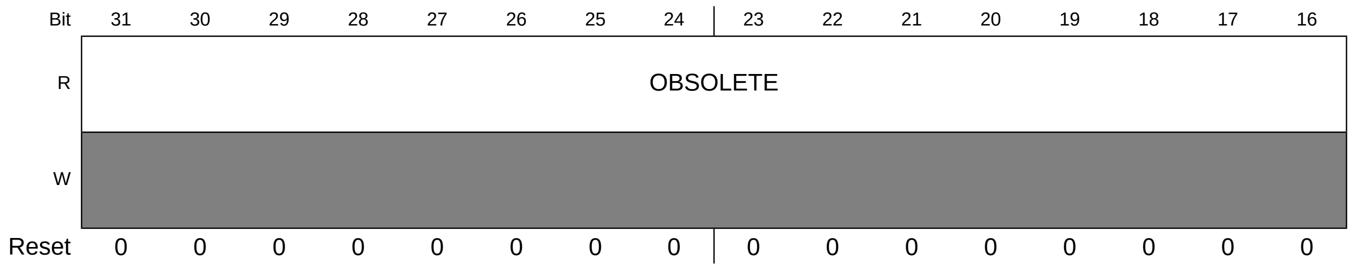
HW_DRAM_CTL32 field descriptions

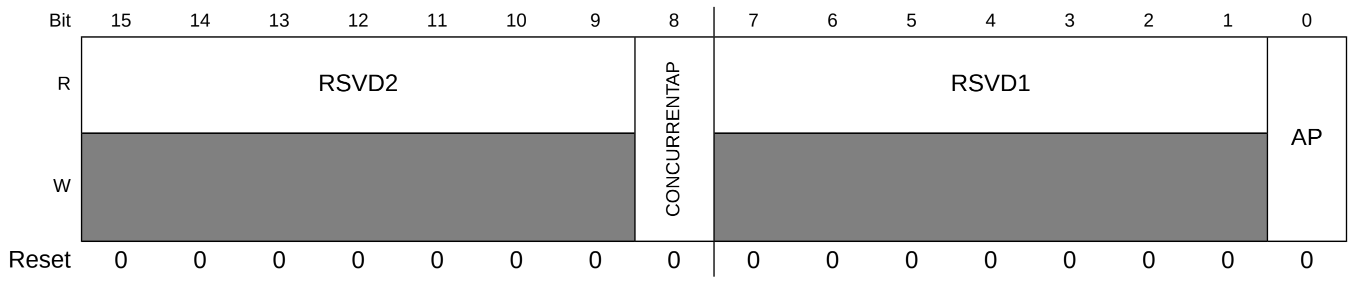
Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD2	Always write zeroes to this field.
8 REDUC	Enable the half datapath feature of the controller. Controls the width of the memory datapath. When enabled, the upper half of the memory buses (DQ, DQS and DM) are unused and relevant data only exists in the lower half of the buses. This parameter expands the EMI for use with memory devices of the configured width or half of the configured width. The entire user datapath is used regardless of this setting. 'b0 = Standard operation using full memory bus. 'b1 = Memory datapath width is half of the maximum size. The upper half of the data_byte_disable bus will be driven to 'b1.
7–1 RSVD1	Always write zeroes to this field.
0 REG_DIMM_ ENABLE	Enable registered DIMM operation of the controller. Enables registered DIMM operations to control the address and command pipeline of the EMI. 'b0 = Normal operation 'b1 = Enable registered DIMM operation

14.8.31 DRAM Control Register 33 (HW_DRAM_CTL33)

This is a DRAM configuration register.

Address: HW_DRAM_CTL33 – 800E_0000h base + 84h offset = 800E_0084h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
OBSOLETE
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0


Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSVD2
CONCURRENTAP
RSVD1
AP
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_DRAM_CTL33 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD2	Always write zeroes to this field.
8 CONCURRENTAP	Allow controller to issue cmds to other banks while a bank is in auto pre-charge. Enables concurrent auto pre-charge. Some DRAM devices do not allow one bank to be auto pre-charged while another bank is reading or writing. The JEDEC standard allows concurrent auto pre-charge. The user should set this parameter if the DRAM device supports this feature. 'b0 = Concurrent auto pre-charge disabled. 'b1 = Concurrent auto pre-charge enabled.
7–1 RSVD1	Always write zeroes to this field.
0 AP	Enable auto pre-charge mode of controller. Enables auto pre-charge mode for DRAM devices. This parameter may not be modified after the start parameter has been asserted. 'b0 = Auto pre-charge mode disabled. Memory banks will stay open until another request requires this bank, the maximum open time (tras_max) has elapsed, or a refresh command closes all the banks. 'b1 = Auto pre-charge mode enabled. All read and write transactions must be terminated by an auto pre-charge command. If a transaction consists of multiple read or write bursts, only the last command is issued with an auto pre-charge.

14.8.32 DRAM Control Register 34 (HW_DRAM_CTL34)

This is a DRAM configuration register.

Address: HW_DRAM_CTL34 – 800E_0000h base + 88h offset = 800E_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4							WRITEINTERP	RSVD3							INTRPTWRITEA
W	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							INTRPTREADA	RSVD1							INTRPTAPBURST
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL34 field descriptions

Field	Description
31–25 RSVD4	Always write zeroes to this field.
24 WRITEINTERP	Allow controller to interrupt a write burst to the DRAMs with a read cmd. Defines whether the EMI can interrupt a write burst with a read command. Some memory devices do not allow this functionality. For DDR1 or LPDDR1 memory devices, consult the memory specification for the setting for this parameter. For DDR2memory devices, this parameter must be cleared to 'b0. 'b0 = The device does not support read commands interrupting write commands. 'b1 = The device does support read commands interrupting write commands.
23–17 RSVD3	Always write zeroes to this field.
16 INTRPTWRITEA	Allow the controller to interrupt a combined write with auto pre-charge cmd with another write cmd. Enables interrupting of a combined write with auto pre-charge command with another read or write command to the same bank before the first write command is completed. 'b0 = Disable interrupting a combined write with auto pre-charge command with another read or write command to the same bank. 'b1 = Enable interrupting a combined write with auto pre-charge command with another read or write command to the same bank.
15–9 RSVD2	Always write zeroes to this field.
8 INTRPTREADA	Allow the controller to interrupt a combined read with auto pre-charge cmd with another read cmd. Enables interrupting of a combined read with auto pre-charge command with another read command to the same bank before the first read command is completed. 'b0 = Disable interrupting the combined read with auto pre-charge command with another read command to the same bank. 'b1 = Enable interrupting the combined read with auto pre-charge command with another read command to the same bank.
7–1 RSVD1	Always write zeroes to this field.
0 INTRPTAPBURST	Allow the controller to interrupt an auto pre-charge cmd with another cmd. Enables interrupting an auto pre-charge command with another command for a different bank. If enabled, the current operation will be interrupted. However, the bank will be pre-charged as if the current operation were allowed to continue. 'b0 = Disable interrupting an auto pre-charge operation on a different bank.

HW_DRAM_CTL34 field descriptions (continued)

Field	Description
	'b1 = Enable interrupting an auto pre-charge operation on a different bank.

14.8.33 DRAM Control Register 35 (HW_DRAM_CTL35)

This is a DRAM configuration register.

Address: HW_DRAM_CTL35 – 800E_0000h base + 8Ch offset = 800E_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD3							PWRUP_SREFRESH_EXIT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2							NO_CMD_INIT	RSVD1				INITAREF			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL35 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–17 RSVD3	Always write zeroes to this field.
16 PWRUP_SREFRESH_EXIT	Allow powerup through self-refresh instead of full memory initialization. Allows controller to exit power-down mode by executing a self-refresh exit instead of the full memory initialization. This parameter provides a means to skip full initialization when the DRAM devices are in a known self-refresh state. 'b0 = Disabled 'b1 = Enabled
15–9 RSVD2	Always write zeroes to this field.
8 NO_CMD_INIT	Disable DRAM cmds until TDLL has expired during initialization. Disables DRAM commands until DLL initialization is complete and tdll has expired. 'b0 = Issue only REF and PRE commands during DLL initialization of the DRAM devices. If PRE commands are issued before DLL initialization is complete, the command will be executed immediately, and then the DLL initialization will continue.

HW_DRAM_CTL35 field descriptions (continued)

Field	Description
	'b1 = Do not issue any type of command during DLL initialization of the DRAM devices. If any other commands are issued during the initialization time, they will be held off until DLL initialization is complete.
7–4 RSVD1	Always write zeroes to this field.
3–0 INITAREF	Number of auto-refresh cmds to execute during DRAM initialization. Defines the number of auto-refresh commands needed by the DRAM devices to satisfy the initialization sequence.

14.8.34 DRAM Control Register 36 (HW_DRAM_CTL36)

This is a DRAM configuration register.

Address: HW_DRAM_CTL36 – 800E_0000h base + 90h offset = 800E_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3							TREF_ENABLE	RSVD2							TRAS_LOCKOUT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE								RSVD1							FAST_WRITE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL36 field descriptions

Field	Description
31–25 RSVD3	Always write zeroes to this field.
24 TREF_ENABLE	Issue auto-refresh cmds to the DRAMs every TREF cycles. Enables refresh commands. If command refresh mode is configured, then refresh commands will be automatically issued based on the tref parameter value and any refresh commands sent through the command interface or the register interface. Refreshes will still occur even if the DRAM devices have been placed in power down state by the assertion of the power_down parameter. 'b0 = Refresh commands disabled. 'b1 = Refresh commands enabled.
23–17 RSVD2	Always write zeroes to this field.

HW_DRAM_CTL36 field descriptions (continued)

Field	Description
16 TRAS_LOCKOUT	Allow the controller to execute auto pre-charge cmds before TRAS_MIN expires. Defines the tRAS lockout setting for the DRAM device. tRAS lockout allows the EMI to execute auto pre-charge commands before the tras_min parameter has expired. 'b0 = tRAS lockout not supported by memory device. 'b1 = tRAS lockout supported by memory device.
15–8 OBSOLETE	Always write zeroes to this field.
7–1 RSVD1	Always write zeroes to this field.
0 FAST_WRITE	Define when write cmds are issued to DRAM devices. Controls when the write commands are issued to the DRAM devices. 'b0 = The EMI will issue a write command to the DRAM devices when it has received enough data for one DRAM burst. In this mode, write data can be sent in any cycle relative to the write command. This mode also allows for multi-word write command data to arrive in non-sequential cycles. 'b1 = The EMI will issue a write command to the DRAM devices after the first word of the write data is received by the EMI. The first word can be sent at any time relative to the write command. In this mode, multi-word write command data must be available to the EMI in sequential cycles.

14.8.35 DRAM Control Register 37 (HW_DRAM_CTL37)

This is a DRAM configuration register.

Address: HW_DRAM_CTL37 – 800E_0000h base + 94h offset = 800E_0094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				CASLAT_LIN_GATE				RSVD3				CASLAT_LIN				RSVD2					CASLAT			RSVD1				WRLAT			
W	▨								▨								▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL37 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 CASLAT_LIN_ GATE	Adjusts data capture gate open by half cycles. Adjusts the data capture gate open time by 1/2 cycle increments. This parameter is programmed differently than caslat_lin when there are fixed offsets in the flight path between the memories and the EMI for clock gating. When caslat_lin_gate is a larger value than caslat_lin, the data capture window will become shorter. A caslat_lin_gate value smaller than caslat_lin may have no effect on the data capture window, depending on the fixed offsets in the ASIC and the board. For optimal synthesis behavior, the ODT path for a CAS latency of three is clocked at a 200 MHz clock regardless of configured maximum speed.

HW_DRAM_CTL37 field descriptions (continued)

Field	Description
	'b0000 - 'b0001 = Reserved 'b0010 = 1 cycle 'b0011 = 1.5 cycles 'b0100 = 2 cycles 'b0101 = 2.5 cycles 'b0110 = 3 cycles 'b0111 = 3.5 cycles 'b1000 = 4 cycles 'b1001 = 4.5 cycles 'b1010 = 5 cycles 'b1011 = 5.5 cycles 'b1100 = 6 cycles 'b1101 = 6.5 cycles 'b1110 = 7 cycles 'b1111 = 7.5 cycles
23–20 RSVD3	Always write zeroes to this field.
19–16 CASLAT_LIN	Sets latency from read cmd send to data receive from/to controller. Sets the CAS latency linear value in 1/2 cycle increments. This sets an internal adjustment for the delay from when the read command is sent from the EMI to when data will be received back. The window of time in which the data is captured is a fixed length. The caslat_lin parameter adjusts the start of this data capture window. Not all linear values will be supported for the memory devices being used. Refer to the specification for the memory devices being used. For optimal synthesis behavior, the ODT path for a CAS latency of three is clocked at a 200 MHz clock regardless of configured maximum speed. 'b0000 - 'b0001 = Reserved 'b0010 = 1 cycle 'b0011 = 1.5 cycles 'b0100 = 2 cycles 'b0101 = 2.5 cycles 'b0110 = 3 cycles 'b0111 = 3.5 cycles 'b1000 = 4 cycles 'b1001 = 4.5 cycles 'b1010 = 5 cycles 'b1011 = 5.5 cycles 'b1100 = 6 cycles 'b1101 = 6.5 cycles

HW_DRAM_CTL37 field descriptions (continued)

Field	Description
	'b1110 = 7 cycles 'b1111 = 7.5 cycles
15–11 RSVD2	Always write zeroes to this field.
10–8 CASLAT	Encoded CAS latency sent to DRAMs during initialization. Sets the CAS (Column Address Strobe) latency encoding that the memory uses. The binary value programmed into this parameter is dependent on the memory device, since the same caslat value may have different meanings to different memories. This will be programmed into the DRAM devices at initialization. The CAS encoding will be specified in the DRAM spec sheet, and should correspond to the caslat_lin parameter. Refer to the files in the regconfigs/ directory in the release for actual settings for each particular device. For optimal synthesis behavior, the ODT path for a CAS latency of three is clocked at a 200 MHz clock regardless of configured maximum speed.
7–4 RSVD1	Always write zeroes to this field.
3–0 WRLAT	DRAM WRLAT parameter in cycles. Defines the write latency from when the write command is issued to the time the write data is presented to the DRAM devices, in cycles. This parameter must be set to 'b1 when the EMI is used in DDR1 mode.

14.8.36 DRAM Control Register 38 (HW_DRAM_CTL38)

This is a DRAM configuration register.

Address: HW_DRAM_CTL38 – 800E_0000h base + 98h offset = 800E_0098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2			TDAL					TCPD																RSVD1					TCKE		
W	▨	▨	▨																						▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL38 field descriptions

Field	Description
31–29 RSVD2	Always write zeroes to this field.
28–24 TDAL	DRAM TDAL parameter in cycles. Defines the auto pre-charge write recovery time when auto pre-charge is enabled (the ap parameter is set to 'b1), in cycles. This is defined internally as tRP (pre-charge time) + auto pre-charge write recovery time.

HW_DRAM_CTL38 field descriptions (continued)

Field	Description
	Not all memories use this parameter. If tDAL is defined in the memory specification, then program this parameter to the specified value. If the memory does not specify a tDAL time, then program this parameter to tWR + tRP. DO NOT program this parameter with a value of 0x0 or the EMI will not function properly when auto pre-charge is enabled.
23–8 TCPD	DRAM TCPD parameter in cycles. Defines the clock enable to pre-charge delay time for the DRAM devices, in cycles.
7–3 RSVD1	Always write zeroes to this field.
2–0 TCKE	Minimum CKE pulse width. Defines the minimum CKE pulse width, in cycles.

14.8.37 DRAM Control Register 39 (HW_DRAM_CTL39)

This is a DRAM configuration register.

Address: HW_DRAM_CTL39 – 800E_0000h base + 9Ch offset = 800E_009Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		TFAW						OBSOLETE								TDLL															
W	▨	▨							▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL39 field descriptions

Field	Description
31–30 RSVD1	Always write zeroes to this field.
29–24 TFAW	DRAM TFAW parameter in cycles. Defines the DRAM tFAW parameter, in cycles.
23–16 OBSOLETE	Always write zeroes to this field.
15–0 TDLL	DRAM TDLL parameter in cycles. Defines the DRAM DLL lock time, in cycles.

14.8.38 DRAM Control Register 40 (HW_DRAM_CTL40)

This is a DRAM configuration register.

Address: HW_DRAM_CTL40 – 800E_0000h base + A0h offset = 800E_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1			TMRD					TINIT																							
W	▨	▨	▨																													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL40 field descriptions

Field	Description
31–29 RSVD1	Always write zeroes to this field.
28–24 TMRD	DRAM TMRD parameter in cycles. Defines the minimum number of cycles required between two mode register write commands. This is the time required to complete the write operation to the mode register.
23–0 TINIT	DRAM TINIT parameter in cycles. Defines the DRAM initialization time, in cycles.

14.8.39 DRAM Control Register 41 (HW_DRAM_CTL41)

This is a DRAM configuration register.

Address: HW_DRAM_CTL41 – 800E_0000h base + A4h offset = 800E_00A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TPDEX																TRCD_INT								RSVD1		TRC					
W																									▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL41 field descriptions

Field	Description
31–16 TPDEX	DRAM TPDEX parameter in cycles. Defines the DRAM power-down exit command period, in cycles.
15–8 TRCD_INT	DRAM TRCD parameter in cycles. Defines the DRAM RAS to CAS delay, in cycles.
7–6 RSVD1	Always write zeroes to this field.
5–0 TRC	DRAM TRC parameter in cycles. Defines the DRAM period between active commands for the same bank, in cycles.

14.8.40 DRAM Control Register 42 (HW_DRAM_CTL42)

This is a DRAM configuration register.

Address: HW_DRAM_CTL42 – 800E_0000h base + A8h offset = 800E_00A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE								TRAS_MAX																TRAS_MIN							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL42 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–8 TRAS_MAX	DRAM TRAS_MAX parameter in cycles. Defines the DRAM maximum row active time, in cycles.
7–0 TRAS_MIN	DRAM TRAS_MIN parameter in cycles. Defines the DRAM minimum row activate time, in cycles.

14.8.41 DRAM Control Register 43 (HW_DRAM_CTL43)

This is a DRAM configuration register.

Address: HW_DRAM_CTL43 – 800E_0000h base + ACh offset = 800E_00ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				TRP				TRFC								RSVD1		TREF													
W	▨	▨	▨	▨													▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL43 field descriptions

Field	Description
31–28 RSVD2	Always write zeroes to this field.
27–24 TRP	DRAM TRP parameter in cycles. Defines the DRAM pre-charge command time, in cycles.
23–16 TRFC	DRAM TRFC parameter in cycles. Defines the DRAM refresh command time, in cycles.

HW_DRAM_CTL43 field descriptions (continued)

Field	Description
15–14 RSVD1	Always write zeroes to this field.
13–0 TREF	DRAM TREF parameter in cycles. Defines the DRAM cycles between refresh commands.

14.8.42 DRAM Control Register 44 (HW_DRAM_CTL44)

This is a DRAM configuration register.

Address: HW_DRAM_CTL44 – 800E_0000h base + B0h offset = 800E_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				TWTR				RSVD3			TWR_INT					RSVD2					TRTP			RSVD1					TRRD		
W	▨	▨	▨	▨					▨	▨	▨						▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL44 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 TWTR	DRAM TWTR parameter in cycles. Sets the number of cycles needed to switch from a write to a read operation, as dictated by the DDR SDRAM specification.
23–21 RSVD3	Always write zeroes to this field.
20–16 TWR_INT	DRAM TWR parameter in cycles. Defines the DRAM write recovery time, in cycles.
15–11 RSVD2	Always write zeroes to this field.
10–8 TRTP	DRAM TRTP parameter in cycles. For DDR1, this parameter has no meaning. For LPDDR1 or DDR2, defines the DRAM tRTP (read to pre-charge time) parameter, in cycles.
7–3 RSVD1	Always write zeroes to this field.
2–0 TRRD	DRAM TRRD parameter in cycles. Defines the DRAM activate to activate delay for different banks, in cycles.

14.8.43 DRAM Control Register 45 (HW_DRAM_CTL45)

This is a DRAM configuration register.

Address: HW_DRAM_CTL45 – 800E_0000h base + B4h offset = 800E_00B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TXSR																TXSNR															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL45 field descriptions

Field	Description
31–16 TXSR	DRAM TXSR parameter in cycles. Defines the DRAM time from a self-refresh exit to a command that requires the memory DLL to be locked.
15–0 TXSNR	DRAM TXSNR parameter in cycles. Defines the DRAM time from a self-refresh exit to a command that does not require the memory DLL to be locked. (txs)

14.8.44 DRAM Control Register 48 (HW_DRAM_CTL48)

This is a DRAM configuration register.

Address: HW_DRAM_CTL48 – 800E_0000h base + C0h offset = 800E_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4	AXI0_CURRENT_BDW							RSVD3							AXI0_BDW_OVFLOW
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2	AXI0_BDW							RSVD1						AXI0_FIFO_TYPE_REG	
W	▨								▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL48 field descriptions

Field	Description
31 RSVD4	Always write zeroes to this field.

HW_DRAM_CTL48 field descriptions (continued)

Field	Description
30–24 AXI0_ CURRENT_BDW	Current bandwidth usage percentage for port 0. READ-ONLY.
23–17 RSVD3	Always write zeroes to this field.
16 AXI0_BDW_ OVFLOW	Port 0 behavior when bandwidth maximized.
15 RSVD2	Always write zeroes to this field.
14–8 AXI0_BDW	Maximum bandwidth percentage for port 0.
7–2 RSVD1	Always write zeroes to this field.
1–0 AXI0_FIFO_ TYPE_REG	Clock domain relativity between AXI port 0 and core logic.

14.8.45 DRAM Control Register 49 (HW_DRAM_CTL49)

This is a DRAM configuration register.

Address: HW_DRAM_CTL49 – 800E_0000h base + C4h offset = 800E_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI0_EN_SIZE_LT_WIDTH_INSTR																RSVD2					AXI0_W_PRIORITY			RSVD1					AXI0_R_PRIORITY		
W																	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL49 field descriptions

Field	Description
31–16 AXI0_EN_SIZE_ LT_WIDTH_ INSTR	Allow narrow instructions from AXI port 0 requestors with bit enabled.
15–11 RSVD2	Always write zeroes to this field.
10–8 AXI0_W_ PRIORITY	Priority of write cmds from AXI port 0.
7–3 RSVD1	Always write zeroes to this field.

HW_DRAM_CTL49 field descriptions (continued)

Field	Description
2–0 AXI0_R_ PRIORITY	Priority of read cmds from AXI port 0.

14.8.46 DRAM Control Register 50 (HW_DRAM_CTL50)

This is a DRAM configuration register.

Address: HW_DRAM_CTL50 – 800E_0000h base + C8h offset = 800E_00C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4	AXI1_CURRENT_BDW							RSVD3							AXI1_BDW_OVFLOW
W	▨	▨							▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2	AXI1_BDW							RSVD1						AXI1_FIFO_TYPE_REG	
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL50 field descriptions

Field	Description
31 RSVD4	Always write zeroes to this field.
30–24 AXI1_ CURRENT_BDW	Current bandwidth usage percentage for port 1. READ-ONLY.
23–17 RSVD3	Always write zeroes to this field.
16 AXI1_BDW_ OVFLOW	Port 1 behavior when bandwidth maximized.
15 RSVD2	Always write zeroes to this field.
14–8 AXI1_BDW	Maximum bandwidth percentage for port 1.
7–2 RSVD1	Always write zeroes to this field.

HW_DRAM_CTL50 field descriptions (continued)

Field	Description
1–0 AXI1_FIFO_ TYPE_REG	Clock domain relativity between AXI port 1 and core logic.

14.8.47 DRAM Control Register 51 (HW_DRAM_CTL51)

This is a DRAM configuration register.

Address: HW_DRAM_CTL51 – 800E_0000h base + CCh offset = 800E_00CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI1_EN_SIZE_LT_WIDTH_INSTR																RSVD2					AXI1_W_ PRIORITY			RSVD1					AXI1_R_ PRIORITY		
W																	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

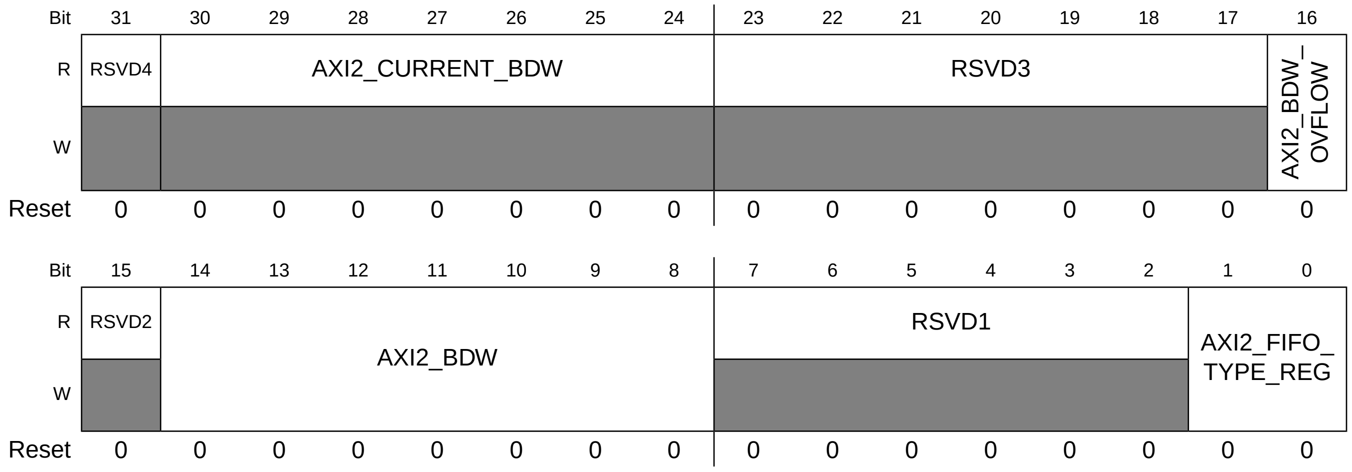
HW_DRAM_CTL51 field descriptions

Field	Description
31–16 AXI1_EN_SIZE_ LT_WIDTH_ INSTR	Allow narrow instructions from AXI port 1 requestors with bit enabled.
15–11 RSVD2	Always write zeroes to this field.
10–8 AXI1_W_ PRIORITY	Priority of write cmds from AXI port 1.
7–3 RSVD1	Always write zeroes to this field.
2–0 AXI1_R_ PRIORITY	Priority of read cmds from AXI port 1.

14.8.48 DRAM Control Register 52 (HW_DRAM_CTL52)

This is a DRAM configuration register.

Address: HW_DRAM_CTL52 – 800E_0000h base + D0h offset = 800E_00D0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R RSVD4 AXI2_CURRENT_BDW RSVD3 AXI2_BDW_OVFLOW
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R RSVD2 AXI2_BDW RSVD1 AXI2_FIFO_TYPE_REG
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_DRAM_CTL52 field descriptions

Field	Description
31 RSVD4	Always write zeroes to this field.
30–24 AXI2_ CURRENT_BDW	Current bandwidth usage percentage for port 2. READ-ONLY.
23–17 RSVD3	Always write zeroes to this field.
16 AXI2_BDW_ OVFLOW	Port 2 behavior when bandwidth maximized.
15 RSVD2	Always write zeroes to this field.
14–8 AXI2_BDW	Maximum bandwidth percentage for port 2.
7–2 RSVD1	Always write zeroes to this field.
1–0 AXI2_FIFO_ TYPE_REG	Clock domain relativity between AXI port 2 and core logic.

14.8.49 DRAM Control Register 53 (HW_DRAM_CTL53)

This is a DRAM configuration register.

Address: HW_DRAM_CTL53 – 800E_0000h base + D4h offset = 800E_00D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI2_EN_SIZE_LT_WIDTH_INSTR																RSVD2					AXI2_W_ PRIORITY			RSVD1					AXI2_R_ PRIORITY		
W																	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL53 field descriptions

Field	Description
31–16 AXI2_EN_SIZE_ LT_WIDTH_ INSTR	Allow narrow instructions from AXI port 2 requestors with bit enabled.
15–11 RSVD2	Always write zeroes to this field.
10–8 AXI2_W_ PRIORITY	Priority of write cmds from AXI port 2.
7–3 RSVD1	Always write zeroes to this field.
2–0 AXI2_R_ PRIORITY	Priority of read cmds from AXI port 2.

14.8.50 DRAM Control Register 54 (HW_DRAM_CTL54)

This is a DRAM configuration register.

Address: HW_DRAM_CTL54 – 800E_0000h base + D8h offset = 800E_00D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4	AXI3_CURRENT_BDW							RSVD3							AXI3_BDW_ OVFLOW
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2	AXI3_BDW							RSVD1						AXI3_FIFO_ TYPE_REG	
W	▨								▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL54 field descriptions

Field	Description
31 RSVD4	Always write zeroes to this field.
30–24 AXI3_ CURRENT_BDW	Current bandwidth usage percentage for port 3. READ-ONLY.
23–17 RSVD3	Always write zeroes to this field.
16 AXI3_BDW_ OVFLOW	Port 3 behavior when bandwidth maximized.
15 RSVD2	Always write zeroes to this field.
14–8 AXI3_BDW	Maximum bandwidth percentage for port 3.
7–2 RSVD1	Always write zeroes to this field.
1–0 AXI3_FIFO_ TYPE_REG	Clock domain relativity between AXI port 3 and core logic.

14.8.51 DRAM Control Register 55 (HW_DRAM_CTL55)

This is a DRAM configuration register.

Address: HW_DRAM_CTL55 – 800E_0000h base + DCh offset = 800E_00DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AXI3_EN_SIZE_LT_WIDTH_INSTR																RSVD2					AXI3_W_PRIORITY			RSVD1					AXI3_R_PRIORITY		
W																	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL55 field descriptions

Field	Description
31–16 AXI3_EN_SIZE_ LT_WIDTH_ INSTR	Allow narrow instructions from AXI port 3 requestors with bit enabled.
15–11 RSVD2	Always write zeroes to this field.
10–8 AXI3_W_ PRIORITY	Priority of write cmds from AXI port 3.

HW_DRAM_CTL55 field descriptions (continued)

Field	Description
7–3 RSVD1	Always write zeroes to this field.
2–0 AXI3_R_ PRIORITY	Priority of read cmds from AXI port 3.

14.8.52 DRAM Control Register 56 (HW_DRAM_CTL56)

This is a DRAM configuration register.

Address: HW_DRAM_CTL56 – 800E_0000h base + E0h offset = 800E_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE[15:8]								RSVD1					ARB_CMD_Q_ THRESHOLD		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL56 field descriptions

Field	Description
31–8 OBSOLETE	Always write zeroes to this field.
7–3 RSVD1	Always write zeroes to this field.
2–0 ARB_CMD_Q_ THRESHOLD	Threshold for cmd queue fullness related to overflow. Sets the command queue fullness that determines if ports will be allowed to overflow. This parameter is used in conjunction with the axiY_bdw_ovflow parameters.

14.8.53 DRAM Control Register 58 (HW_DRAM_CTL58)

This is a DRAM configuration register.

Address: HW_DRAM_CTL58 – 800E_0000h base + E8h offset = 800E_00E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2					INT_STATUS											RSVD1					INT_MASK										
W	▨					▨											▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL58 field descriptions

Field	Description
31–27 RSVD2	Always write zeroes to this field.
26–16 INT_STATUS	Status of interrupt features in the controller. READ-ONLY Shows the status of all possible interrupts generated by the EMI. The MSB is the result of a logical OR of all the lower bits. This parameter is read-only. Backwards compatibility is available for register parameters across configurations. However, even with this compatibility, the individual bits, their meaning and the size of the int_status parameter may change. The int_status bits correspond to these interrupts: Bit [10] = Logical OR of all lower bits. Bit [9] = User-initiated DLL resync is finished. Bit [8] = DLL lock state change condition detected. (i.e. lock to unlock or unlock to lock) Bit [7] = Indicates that a read DQS gate error occurred. Bit [6] = ODT enabled and CAS Latency 3 programmed error detected. This is an unsupported programming option. Bit [5] = Both DDR2 and Mobile modes have been enabled. Bit [4] = DRAM initialization complete. Bit [3] = Error was found with command data channel in a port. Bit [2] = Error was found with command channel in a port. Bit [1] = Multiple accesses outside the defined PHYSICAL memory space detected. Bit [0] = A single access outside the defined PHYSICAL memory space detected.
15–11 RSVD1	Always write zeroes to this field.
10–0 INT_MASK	Mask for controller_int signals from the INT_STATUS parameter. Active-high mask bits that control the value of the EMI_int signal on the ASIC interface. Unless the user has suppressed interrupt reporting (by setting bit [10] of this parameter to 'b1), bits [9:0] of the int_mask parameter will be inverted and logically AND'ed with bits [9:0] of the int_status parameter and the result is reported on the controller_int signal.

14.8.54 DRAM Control Register 59 (HW_DRAM_CTL59)

This is a DRAM configuration register.

Address: HW_DRAM_CTL59 – 800E_0000h base + ECh offset = 800E_00ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OUT_OF_RANGE_ADDR																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL59 field descriptions

Field	Description
31–0 OUT_OF_ RANGE_ADDR	Address of cmd that caused an Out-of-Range interrupt. READ-ONLY.

14.8.55 DRAM Control Register 60 (HW_DRAM_CTL60)

This is a DRAM configuration register.

Address: HW_DRAM_CTL60 – 800E_0000h base + F0h offset = 800E_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE[15:8]								RSVD1						OUT_OF_ RANGE_ ADDR	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL60 field descriptions

Field	Description
31–8 OBSOLETE	Always write zeroes to this field.
7–2 RSVD1	Always write zeroes to this field.
1–0 OUT_OF_ RANGE_ADDR	Address of cmd that caused an Out-of-Range interrupt. READ-ONLY.

14.8.56 DRAM Control Register 61 (HW_DRAM_CTL61)

This is a DRAM configuration register.

Address: HW_DRAM_CTL61 – 800E_0000h base + F4h offset = 800E_00F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3		OUT_OF_RANGE_TYPE						RSVD2	OUT_OF_RANGE_LENGTH						
W	▨		▨						▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1			OUT_OF_RANGE_SOURCE_ID												
W	▨			▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL61 field descriptions

Field	Description
31–30 RSVD3	Always write zeroes to this field.
29–24 OUT_OF_ RANGE_TYPE	Type of cmd that caused an Out-of-Range interrupt. READ-ONLY Holds the type of command that caused an out-of-range interrupt request to the memory devices. This parameter is read-only.
23 RSVD2	Always write zeroes to this field.
22–16 OUT_OF_ RANGE_ LENGTH	Length of cmd that caused an Out-of-Range interrupt. READ-ONLY Holds the length of the command that caused an out-of-range interrupt request to the memory devices. This parameter is read-only.
15–13 RSVD1	Always write zeroes to this field.
12–0 OUT_OF_ RANGE_ SOURCE_ID	Source ID of cmd that caused an Out-of-Range interrupt. READ-ONLY Holds the Source ID of the command that caused an out-of-range interrupt request to the memory devices. This parameter is read-only.

14.8.57 DRAM Control Register 62 (HW_DRAM_CTL62)

This is a DRAM configuration register.

Address: HW_DRAM_CTL62 – 800E_0000h base + F8h offset = 800E_00F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORT_CMD_ERROR_ADDR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL62 field descriptions

Field	Description
31–0 PORT_CMD_ ERROR_ADDR	Address of port that caused the PORT cmd error. READ-ONLY.

14.8.58 DRAM Control Register 63 (HW_DRAM_CTL63)

This is a DRAM configuration register.

Address: HW_DRAM_CTL63 – 800E_0000h base + FCh offset = 800E_00FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE[15:8]								RSVD1						PORT_CMD_ERROR_ADDR	
W	▨								▨						▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL63 field descriptions

Field	Description
31–8 OBSOLETE	Always write zeroes to this field.
7–2 RSVD1	Always write zeroes to this field.
1–0 PORT_CMD_ ERROR_ADDR	Address of port that caused the PORT cmd error. READ-ONLY.

14.8.59 DRAM Control Register 64 (HW_DRAM_CTL64)

This is a DRAM configuration register.

Address: HW_DRAM_CTL64 – 800E_0000h base + 100h offset = 800E_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD2			PORT_CMD_ERROR_ID[20:16]				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORT_CMD_ERROR_ID[15:8]								RSVD1				PORT_CMD_ERROR_TYPE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL64 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–21 RSVD2	Always write zeroes to this field.
20–8 PORT_CMD_ ERROR_ID	Source ID of cmd that caused the PORT cmd error. READ-ONLY Holds the source ID of the command that caused a port command error condition. For AXI ports, the source ID is comprised of the Port ID and the Requestor ID, where the Requestor ID is the axiY_AWID for write commands or the axiY_ARID for read commands. This parameter is read-only.
7–4 RSVD1	Always write zeroes to this field.
3–0 PORT_CMD_ ERROR_TYPE	Type of error and access type that caused the PORT cmd error. READ-ONLY Defines the type of error and the access type that caused the port command error condition. If multiple bits are set to 'b1, then multiple errors were found. This parameter is read-only. Bit [3] = Narrow transfer requested for a requestor Y whose axiY_en_size_lt_width_instr parameter is clear. Bit [2] = Reserved. Bit [1] = Reserved. Bit [0] = Reserved.

14.8.60 DRAM Control Register 65 (HW_DRAM_CTL65)

This is a DRAM configuration register.

Address: HW_DRAM_CTL65 – 800E_0000h base + 104h offset = 800E_0104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD2			PORT_DATA_ERROR_ID[20:16]				
W	▨								▨			▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PORT_DATA_ERROR_ID[15:8]								RSVD1					PORT_DATA_ ERROR_TYPE		
W	▨								▨					▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL65 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–21 RSVD2	Always write zeroes to this field.
20–8 PORT_DATA_ ERROR_ID	Source ID of cmd that caused the PORT data error. READ-ONLY Holds the source ID of the command that caused a port data error condition. For AXI ports, the source ID is comprised of the Port ID and the Requestor ID, where the Requestor ID is the axiY_BID for write response errors, the axiY_RID for read data errors, or the axiY_WID for write data errors. This parameter is read-only.
7–3 RSVD1	Always write zeroes to this field.
2–0 PORT_DATA_ ERROR_TYPE	Type of error and access type that caused the PORT data error. READ-ONLY Defines the type of error and the access type that caused the port data error condition. If multiple bits are set to 'b1, then multiple errors were found. This parameter is read-only. Bit [2] = Reserved. Bit [1] = Reserved. Bit [0] = Reserved.

14.8.61 DRAM Control Register 66 (HW_DRAM_CTL66)

This is a DRAM configuration register.

Address: HW_DRAM_CTL66 – 800E_0000h base + 108h offset = 800E_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD2				TDFI_CTRLUPD_MIN			
W	▨								▨				▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1		TDFI_CTRLUPD_MAX													
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL66 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–20 RSVD2	Always write zeroes to this field.
19–16 TDFI_CTRLUPD_MIN	Holds the DFI tCTRLUPD_MIN timing parameter. READ-ONLY Holds the DFI tctrlupd_min timing parameter. This parameter is read-only.
15–14 RSVD1	Always write zeroes to this field.
13–0 TDFI_CTRLUPD_MAX	Holds the DFI tCTRLUPD_MAX timing parameter. Holds the DFI tctrlupd_max timing parameter. This parameter is read-only.

14.8.62 DRAM Control Register 67 (HW_DRAM_CTL67)

This is a DRAM configuration register.

Address: HW_DRAM_CTL67 – 800E_0000h base + 10Ch offset = 800E_010Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				TDFI_DRAM_ CLK_ENABLE				RSVD3					TDFI_ DRAM_CLK_ DISABLE			RSVD2				DRAM_CLK_ ENABLE				RSVD1				TDFI_CTRL_ DELAY			
W	▨	▨	▨	▨					▨	▨	▨	▨	▨				▨	▨	▨	▨					▨	▨	▨	▨				
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL67 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 TDFI_DRAM_ CLK_ENABLE	Delay from DFI clock enable to memory clock enable. Holds the DFI tdram_clk_enable timing parameter. This parameter is currently unused in the EMI.
23–19 RSVD3	Always write zeroes to this field.
18–16 TDFI_DRAM_ CLK_DISABLE	Delay from DFI clock disable to memory clock disable. Holds the DFI tdram_clk_disable timing parameter. This parameter should be programmed with the number of cycles that the PHY requires to disable the clock after the dfi_dram_clk_disable signal is asserted.
15–12 RSVD2	Always write zeroes to this field.
11–8 DRAM_CLK_ ENABLE	Set value for the dfi_dram_clk_disable signal. Sets value for the DFI output signal dfi_dram_clk_disable. Bit [0] controls CS0, Bit [1] controls CS1. For each bit: 'b0 = Memory clock/s should be disabled. 'b1 = Memory clock/s should be active.
7–4 RSVD1	Always write zeroes to this field.
3–0 TDFI_CTRL_ DELAY	Delay from DFI command to memory command. Holds the DFI tctrl_delay timing parameter. This parameter should be programmed with the number of cycles that the PHY requires to send a power-down or self-refresh command to the DRAM devices.

14.8.63 DRAM Control Register 68 (HW_DRAM_CTL68)

This is a DRAM configuration register.

Address: HW_DRAM_CTL68 – 800E_0000h base + 110h offset = 800E_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2		TDFI_PHYUPD_TYPE0														RSVD1		TDFI_PHYUPD_RESP													
W	▨	▨															▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL68 field descriptions

Field	Description
31–30 RSVD2	Always write zeroes to this field.
29–16 TDFI_PHYUPD_ TYPE0	Holds the DFI tPHYUPD_TYPE0 timing parameter. Holds the DFI tphyupd_typeX timing parameter. This parameter is read-only.
15–14 RSVD1	Always write zeroes to this field.
13–0 TDFI_PHYUPD_ RESP	Holds the DFI tPHYUPD_RESP timing parameter. Holds the DFI tphyupd_resp timing parameter. This parameter is read-only.

14.8.64 DRAM Control Register 69 (HW_DRAM_CTL69)

This is a DRAM configuration register.

Address: HW_DRAM_CTL69 – 800E_0000h base + 114h offset = 800E_0114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				TDFI_PHY_WRLAT_BASE				RSVD1				TDFI_PHY_WRLAT			
W	▨	▨	▨	▨					▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL69 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–12 RSVD2	Always write zeroes to this field.

HW_DRAM_CTL69 field descriptions (continued)

Field	Description
11–8 TDFI_PHY_ WRLAT_BASE	Sets DFI base value for the tPHY_WRLAT timing parameter. Used to adjust the tdfi_phy_wrlat parameter for the difference between the PHY's command path delay and data path delay. 'b0000 = Command path delay is 2 cycles shorter than the data path delay. 'b0001 = Command path delay is 1 cycle shorter than the data path delay. 'b0010 = Command path delay and data path delay are equivalent. 'b0011 = Command path delay is 1 cycle longer than the data path delay. 'b0100 = Command path delay is 2 cycles longer than the data path delay. 'b0101 = Command path delay is 3 cycles longer than the data path delay. 'b0110, etc.
7–4 RSVD1	Always write zeroes to this field.
3–0 TDFI_PHY_ WRLAT	Holds the calculated DFI tPHY_WRLAT timing parameter. READ-ONLY Holds the calculated value of the tphy_wrlat timing parameter and is used to adjust the dfi_wrdata_en signal timing. This equation is dependent on the latency setting for the address / control path of the PHY as set in the phy_ctrl_reg_2 [25] parameter bit. If phy_ctrl_reg_2 [25] = 0: If (tdfi_phy_wrlat_base + wrlat_adj) = 2: tdfi_phy_wrlat = reg_dimm_enable If (tdfi_phy_wrlat_base + wrlat_adj) > 2: tdfi_phy_wrlat = tdfi_phy_wrlat_base + wrlat_adj + reg_dimm_enable - WRLAT_WIDTH'h3 Values of (tdfi_phy_wrlat_base + wrlat_adj) < 2 are not supported. If phy_ctrl_reg_2 [25] = 1: If (tdfi_phy_wrlat_base + wrlat_adj) < 4: tdfi_phy_wrlat = reg_dimm_enable If (tdfi_phy_wrlat_base + wrlat_adj) >= 4: tdfi_phy_wrlat = tdfi_phy_wrlat_base + wrlat_adj + reg_dimm_enable - WRLAT_WIDTH'h4 This parameter is read-only.

14.8.65 DRAM Control Register 70 (HW_DRAM_CTL70)

This is a DRAM configuration register.

Address: HW_DRAM_CTL70 – 800E_0000h base + 118h offset = 800E_0118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OBSOLETE								RSVD3				TDFI_RDDATA_EN_BASE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				TDFI_RDDATA_EN				RSVD1				TDFI_PHY_RDLAT			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL70 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–20 RSVD3	Always write zeroes to this field.
19–16 TDFI_RDDATA_EN_BASE	Sets DFI base value for the tRDDATA_EN timing parameter. Used to adjust the tdfi_rddata_en parameter to account for the desired delay from the read command to the the read data enable signal. The CAS latency is defined in the caslat parameter. The dfi_rddata_en signal can only be sent at a minimum of 1 cycle after the read command. If the programmed value results in a delay less than 1 cycle, this value will be ignored and a delay value of 1 will be used. 'b0000 = Delay from read command to read data enable is equivalent to CAS latency-3. 'b0001 = Delay from read command to read data enable is equivalent to CAS latency-2. 'b0010 = Delay from read command to read data enable is equivalent to CAS latency-1. 'b0011 = Delay from read command to read data enable is equivalent to CAS latency. 'b0100 = Delay from read command to read data enable is equivalent to CAS latency+1. 'b0101 = Delay from read command to read data enable is equivalent to CAS latency+2. 'b0110, etc.
15–12 RSVD2	Always write zeroes to this field.
11–8 TDFI_RDDATA_EN	Holds the calculated DFI tRDDATA_EN timing parameter. READ-ONLY Holds the calculated value of the trddata_en timing parameter. This equation is dependent on the latency setting for the address / control path of the PHY as set in the phy_ctrl_reg_2 [25] parameter bit. If phy_ctrl_reg_2 [25] = 0: If (tdfi_rddata_en_base + rdlat_adj) = 2: tdfi_rddata_en = reg_dimm_enable If (tdfi_rddata_en_base + rdlat_adj) > 2: tdfi_rddata_en = tdfi_rddata_en_base + rdlat_adj + reg_dimm_enable - RDLAT_WIDTH'h3 Values of (tdfi_rddata_en_base + rdlat_adj) < 2 are not supported.

HW_DRAM_CTL70 field descriptions (continued)

Field	Description
	If phy_ctrl_reg_2 [25] = 1: If (tdfi_rddata_en_base + rdlat_adj) < 4: tdfi_rddata_en = reg_dimm_enable If (tdfi_rddata_en_base + rdlat_adj) >= 4: tdfi_rddata_en = tdfi_rddata_en_base + rdlat_adj + reg_dimm_enable - RDLAT_WIDTH'h4 This parameter is read-only.
7–4 RSVD1	Always write zeroes to this field.
3–0 TDFI_PHY_ RDLAT	Holds the tPHY_RDLAT timing parameter. Holds the tphy_rdlat timing parameter.

14.8.66 DRAM Control Register 71 (HW_DRAM_CTL71)

This is a DRAM configuration register.

Address: HW_DRAM_CTL71 – 800E_0000h base + 11Ch offset = 800E_011Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_0_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL71 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_0	Controls pad output enable times and other PHY parameters for data slice 0. There is a separate phy_ctrl_reg_0_X parameter for each of the slices of data sent on the DFI data bus. Bit [31] = Enables dynamic termination select in the PHY for the DM pads. 'b0 = Disabled 'b1 = Enabled Bit [29] = Controls termination enable for the DM pads. Set to 'b1 to disable termination. 'b0 = Enabled 'b1 = Disabled Bits [28] = Echo gate control for data slice X. Default 0x0. 'b0 = Uses the dfi_rddata_en signal to create a gate. 'b1 = Creates an echo_gate signal. Bit [27] = Gather FIFO Enable 'b0 = Disabled

HW_DRAM_CTL71 field descriptions (continued)

Field	Description
	'b1 = Enabled Bits [26:24] = Defines the read data delay. Holds the number of cycles to delay the dfi_rddata_en signal prior to enabling the read FIFO. After this delay, the read pointers begin incrementing the read FIFO. Default 0x3. Bit [20] = Sets the pad output enable polarity. Default 0x0. 'b0 = OEN pad 'b1 = OE pad Bit [16] = Subtracts 1/2 cycle from the DQS gate value programmed into phy_ctrl_reg_1_X [2:0] by 1/2 cycle. Default 0x0. This is used when the gate is being aligned to the first DQS, and then is removed to move the gate back into the center of the preamble. This parameter is controlled by the hardware logic when hardware gate training is enabled but should be controlled by software when software leveling is used. 'b0 = Do not adjust 'b1 = Adjust the DQS gate by 1/2 clock forward. Bits [15:12] = Adjusts the starting point of the DQS pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x2. Bits [11:8] = Adjusts the ending point of the DQS pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x7. Bits [6:4] = Adjusts the starting point of the DQ pad output enable window. Lower numbers pull the rising edge earlier in time, and larger numbers cause the rising edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x1. Bits [2:0] = Adjusts the ending point of the DQ pad output enable window. Lower numbers pull the falling edge earlier in time, and larger numbers cause the falling edge to be delayed. Each bit changes the output enable time by a 1/4 cycle resolution. Default 0x4. All other bits undefined.

14.8.67 DRAM Control Register 72 (HW_DRAM_CTL72)

This is a DRAM configuration register.

Address: HW_DRAM_CTL72 – 800E_0000h base + 120h offset = 800E_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_0_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL72 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_1	Controls pad output enable times and other PHY parameters for data slice 1. There is a separate phy_ctrl_reg_0_X parameter for each of the slices of data sent on the DFI data bus. The definition of phy_ctrl_reg_0_X parameter is same for data slice 0~3.

14.8.68 DRAM Control Register 73 (HW_DRAM_CTL73)

This is a DRAM configuration register.

Address: HW_DRAM_CTL73 – 800E_0000h base + 124h offset = 800E_0124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_0_2																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL73 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_2	Controls pad output enable times and other PHY parameters for data slice 2. Data slice 2 is disabled.

14.8.69 DRAM Control Register 74 (HW_DRAM_CTL74)

This is a DRAM configuration register.

Address: HW_DRAM_CTL74 – 800E_0000h base + 128h offset = 800E_0128h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_0_3																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL74 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_0_3	Controls pad output enable times and other PHY parameters for data slice 3. Data slice 3 is disabled.

14.8.70 DRAM Control Register 75 (HW_DRAM_CTL75)

This is a DRAM configuration register.

Address: HW_DRAM_CTL75 – 800E_0000h base + 12Ch offset = 800E_012Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_1_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL75 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_0	Controls pad termination and loopback for data slice 0. There is a separate phy_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. Bits [31:28] = Defines the pad dynamic termination select enable time. Larger values add greater delay to when tsel turns on. Each bit changes the output enable time by a 1/2 cycle resolution. Bits [27:24] = Defines the pad dynamic termination select disable time. Larger values reduce the delay to when tsel turns off. Each bit changes the output enable time by a 1/2 cycle resolution. Bit [23] = Enables dynamic termination select in the PHY for the DQS pads. 'b0 = Disabled 'b1 = Enabled Bit [22] = Controls the polarity of the tsel signal for the DQS and DM pads. 'b0 = Positive Polarity 'b1 = Negative Polarity Bit [21] = Triggers a data return to the EMI. 'b0 = No action 'b1 = Sends loopback data on the dfi_rddata signal. Bit [20] = Selects data output type for phy_obs_reg_0_X [23:8]. 'b0 = Return the expected data. 'b1 = Return the actual data. Bits [19:18] = Loopback control. 'b00 = Normal operational mode. 'b01 = Enables loopback write mode. 'b10 = Stop loopback to check the error register. 'b11 = Clear loopback registers. Bits [17] = Controls the loopback read multiplexer. Bits [16] = Controls the internal write multiplexer. Bits [14:12] = Sets the cycle delay between the LFSR and loopback error check logic. Note that 'h7 is not a valid selection and will result in a false passing result.

HW_DRAM_CTL75 field descriptions (continued)

Field	Description
	Bits [10:8] = Gate Error Delay. Allows the user to adjust the length of time from the dfi_rddata_en assertion to the maximum time in which all of the data should have been returned. If the data is not returned in the time expected by the MC, an error condition may occur. This field default value is based on the delay information provided at configuration. Bits [5:4] = Adjusts the closing of the gate. This field default value is based on the delay information provided at configuration. Bits [2:0] = Coarse adjust of gate open time. This value is the number of cycles to delay the dfi_rddata_en signal prior to opening the gate in full cycle increments. Decreasing this value pulls the gate earlier in time. This field, along with the phy_ctrl_reg_0_X [16] bit should be programmed such that the gate signal lands in the valid DQS gate window.

14.8.71 DRAM Control Register 76 (HW_DRAM_CTL76)

This is a DRAM configuration register.

Address: HW_DRAM_CTL76 – 800E_0000h base + 130h offset = 800E_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_1_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL76 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_1	Controls pad termination and loopback for data slice 1. There is a separate phy_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. The definition of phy_ctrl_reg_1_X parameter is same for data slice 0~3.

14.8.72 DRAM Control Register 77 (HW_DRAM_CTL77)

This is a DRAM configuration register.

Address: HW_DRAM_CTL77 – 800E_0000h base + 134h offset = 800E_0134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_1_2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL77 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_2	Controls pad termination and loopback for data slice 2. Data slice 2 is disabled.

14.8.73 DRAM Control Register 78 (HW_DRAM_CTL78)

This is a DRAM configuration register.

Address: HW_DRAM_CTL78 – 800E_0000h base + 138h offset = 800E_0138h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_1_3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL78 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_1_3	Controls pad termination and loopback for data slice 3. Data slice 3 is disabled.

14.8.74 DRAM Control Register 79 (HW_DRAM_CTL79)

This is a DRAM configuration register.

Address: HW_DRAM_CTL79 – 800E_0000h base + 13Ch offset = 800E_013Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PHY_CTRL_REG_2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL79 field descriptions

Field	Description
31–0 PHY_CTRL_ REG_2	Selects the dfi_rddata_valid delay. Controls PHY functionality. Bit [25] = Selects the address / command path. 'b0 = DFI control signals are captured 1 cycle after being sent from the MC. 'b1 = DFI control signals are captured 1/4 cycle after being sent from the MC.

HW_DRAM_CTL79 field descriptions (continued)

Field	Description
	The Low Latency option is not functional at this time. Bit [24] = Selects the write latency path. 'b0 = Write Data is captured 1 cycle after being sent from the MC 'b1 = Write data is captured 1/4 cycle after being sent from the MC. The Low Latency option is not functional at this time. Bit [23] = DFI control for Mobile MCs. Bit [5] = Enables the pad inputs specifically for external loopback. This bit is tied to the internal signal lpbk_rddata_en. 'b0 = Normal Operation 'b1 = Loopback Mode Bit [4] = Enables the pad outputs specifically for external loopback. This bit is tied to the internal signal lpbk_wrdata_en. 'b0 = Normal Operation 'b1 = Loopback Mode Bits [3:0] = Sets the dfi_rddata_valid delay relative to dfi_rddata_en. All other bits undefined.

14.8.75 DRAM Control Register 80 (HW_DRAM_CTL80)

This is a DRAM configuration register.

Address: HW_DRAM_CTL80 – 800E_0000h base + 140h offset = 800E_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DFT_CTRL_REG																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL80 field descriptions

Field	Description
31–0 DFT_CTRL_REG	Enables the PHY testing mode. Enables the PHY testing mode. Bits [2:1] are used. Bits [2:1] ='b10 = Normal Mode Bits [2:1] ='b01 = Scan Mode Currently Not Functional

14.8.76 DRAM Control Register 81 (HW_DRAM_CTL81)

This is a DRAM configuration register.

Address: HW_DRAM_CTL81 – 800E_0000h base + 144h offset = 800E_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD2			OCD_ADJUST_PUP_CS_0					RSVD1			OCD_ADJUST_PDN_CS_0				
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL81 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–13 RSVD2	Always write zeroes to this field.
12–8 OCD_ADJUST_PUP_CS_0	OCD pull-up adjust setting for DRAMs for chip select 0. This parameter is reserved for future use and should be programmed to 0x0.
7–5 RSVD1	Always write zeroes to this field.
4–0 OCD_ADJUST_PDN_CS_0	OCD pull-down adjust setting for DRAMs for chip select 0. This parameter is reserved for future use and should be programmed to 0x0.

14.8.77 DRAM Control Register 82 (HW_DRAM_CTL82)

This is a DRAM configuration register.

Address: HW_DRAM_CTL82 – 800E_0000h base + 148h offset = 800E_0148h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1							ODT_ALT_EN	OBSOLETE[23:16]							
W	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL82 field descriptions

Field	Description
31–25 RSVD1	Always write zeroes to this field.
24 ODT_ALT_EN	Enable use of non-DFI odt_alt signal. Enables the use of the non-DFI compliant alternative ODT internal signal odt_alt, which is externally viewed as the signal reserved0. This signal is only required if the user intends to use a CAS latency of 3 with ODT support. The user will need to modify the dram_asic.v file for this support. 'b0 = ODT support with CAS latency 3 is not supported. 'b1 = ODT support with CAS latency 3 is supported but is not DFI compliant. This disables the interrupt bit for ODT-with-CAS3 and disables the OVL error.
23–0 OBSOLETE	Always write zeroes to this field.

14.8.78 DRAM Control Register 83 (HW_DRAM_CTL83)

This is a DRAM configuration register.

Address: HW_DRAM_CTL83 – 800E_0000h base + 14Ch offset = 800E_014Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				ODT_RD_ MAP_CS3				RSVD3				ODT_RD_ MAP_CS2				RSVD2				ODT_RD_ MAP_CS1				RSVD1				ODT_RD_ MAP_CS0			
W																																
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL83 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 ODT_RD_MAP_ CS3	Determines which chip(s) will have termination when a read occurs on chip 3. Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_rd_map_cs0 is set to 'b0010, then when CS0 is performing a read, CS1 will have active ODT termination. And if odt_rd_map_cs0 was set to 'b1000, then instead CS3 would be active.

HW_DRAM_CTL83 field descriptions (continued)

Field	Description
	Bit [3] = CS3 will have active ODT termination when chip select X is performing a read. Bit [2] = CS2 will have active ODT termination when chip select X is performing a read. Bit [1] = CS1 will have active ODT termination when chip select X is performing a read. Bit [0] = CS0 will have active ODT termination when chip select X is performing a read. Etc.
23–20 RSVD3	Always write zeroes to this field.
19–16 ODT_RD_MAP_ CS2	Determines which chip(s) will have termination when a read occurs on chip 2. Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_rd_map_cs0 is set to 'b0010, then when CS0 is performing a read, CS1 will have active ODT termination. And if odt_rd_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a read. Bit [2] = CS2 will have active ODT termination when chip select X is performing a read. Bit [1] = CS1 will have active ODT termination when chip select X is performing a read. Bit [0] = CS0 will have active ODT termination when chip select X is performing a read. Etc.
15–12 RSVD2	Always write zeroes to this field.
11–8 ODT_RD_MAP_ CS1	Determines which chip(s) will have termination when a read occurs on chip 1. Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_rd_map_cs0 is set to 'b0010, then when CS0 is performing a read, CS1 will have active ODT termination. And if odt_rd_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a read. Bit [2] = CS2 will have active ODT termination when chip select X is performing a read. Bit [1] = CS1 will have active ODT termination when chip select X is performing a read. Bit [0] = CS0 will have active ODT termination when chip select X is performing a read. Etc.
7–4 RSVD1	Always write zeroes to this field.
3–0 ODT_RD_MAP_ CS0	Determines which chip(s) will have termination when a read occurs on chip 0. Sets up which (if any) chip(s) will have their ODT termination active while a read occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications.

HW_DRAM_CTL83 field descriptions (continued)

Field	Description
	Example: If the system consists of 4 chip selects and odt_rd_map_cs0 is set to 'b0010, then when CS0 is performing a read, CS1 will have active ODT termination. And if odt_rd_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a read. Bit [2] = CS2 will have active ODT termination when chip select X is performing a read. Bit [1] = CS1 will have active ODT termination when chip select X is performing a read. Bit [0] = CS0 will have active ODT termination when chip select X is performing a read. Etc.

14.8.79 DRAM Control Register 84 (HW_DRAM_CTL84)

This is a DRAM configuration register.

Address: HW_DRAM_CTL84 – 800E_0000h base + 150h offset = 800E_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4				ODT_WR_MAP_CS3				RSVD3				ODT_WR_MAP_CS2				RSVD2				ODT_WR_MAP_CS1				RSVD1				ODT_WR_MAP_CS0			
W	▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨					▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL84 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 ODT_WR_MAP_CS3	Determines which chip(s) will have termination when a write occurs on chip 3. Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_wr_map_cs0 is set to 'b0010, then when CS0 is performing a write, CS1 will have active ODT termination. And if odt_wr_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a write. Bit [2] = CS2 will have active ODT termination when chip select X is performing a write. Bit [1] = CS1 will have active ODT termination when chip select X is performing a write. Bit [0] = CS0 will have active ODT termination when chip select X is performing a write. Etc.
23–20 RSVD3	Always write zeroes to this field.

HW_DRAM_CTL84 field descriptions (continued)

Field	Description
19–16 ODT_WR_MAP_ CS2	Determines which chip(s) will have termination when a write occurs on chip 2. Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_wr_map_cs0 is set to 'b0010, then when CS0 is performing a write, CS1 will have active ODT termination. And if odt_wr_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a write. Bit [2] = CS2 will have active ODT termination when chip select X is performing a write. Bit [1] = CS1 will have active ODT termination when chip select X is performing a write. Bit [0] = CS0 will have active ODT termination when chip select X is performing a write. Etc.
15–12 RSVD2	Always write zeroes to this field.
11–8 ODT_WR_MAP_ CS1	Determines which chip(s) will have termination when a write occurs on chip 1. Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_wr_map_cs0 is set to 'b0010, then when CS0 is performing a write, CS1 will have active ODT termination. And if odt_wr_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a write. Bit [2] = CS2 will have active ODT termination when chip select X is performing a write. Bit [1] = CS1 will have active ODT termination when chip select X is performing a write. Bit [0] = CS0 will have active ODT termination when chip select X is performing a write. Etc.
7–4 RSVD1	Always write zeroes to this field.
3–0 ODT_WR_MAP_ CS0	Determines which chip(s) will have termination when a write occurs on chip 0. Sets up which (if any) chip(s) will have their ODT termination active while a write occurs on chip select X. EMI will define this parameter at default for internal testing. The user is required is to change this setting to meet system specifications. Example: If the system consists of 4 chip selects and odt_wr_map_cs0 is set to 'b0010, then when CS0 is performing a write, CS1 will have active ODT termination. And if odt_wr_map_cs0 was set to 'b1000, then instead CS3 would be active. Bit [3] = CS3 will have active ODT termination when chip select X is performing a write. Bit [2] = CS2 will have active ODT termination when chip select X is performing a write. Bit [1] = CS1 will have active ODT termination when chip select X is performing a write. Bit [0] = CS0 will have active ODT termination when chip select X is performing a write. Etc.

14.8.80 DRAM Control Register 85 (HW_DRAM_CTL85)

This is a DRAM configuration register.

Address: HW_DRAM_CTL85 – 800E_0000h base + 154h offset = 800E_0154h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PAD_CTRL_REG_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL85 field descriptions

Field	Description
31–0 PAD_CTRL_ REG_0	Controls pad termination, type and IDDQ settings. These bit settings are specific to the pad models included with this PHY. The user should alter the programming relative to the particular pads being used in their design. EMI provides a dynamic termination signal tsel that provides one bit per byte of memory data. The user must attach this signal to the pads dependent on pad requirements. Bit [12] = Defines the pad impedance if the parallel termination option is enabled. Default 'b1. 'b0 = 75 Ohm 'b1 = 150 Ohm Bit [8] = Defines the pad type. Default 'b1. 'b0 = DDR1 'b1 = DDR2 Bit [5] = Defines the IDDQ_RX select for the signal pads. Default 'b1. 'b0 = Do not feed input into IDDQ 'b1 = Feed input into IDDQ Bit [4] = Defines the IDDQ_TX select for the signal pads. Default 'b1. 'b0 = Do not feed output into IDDQ 'b1 = Feed output into IDDQ Bit [1] = Defines the IDDQ_RX select for the clock pads. Default 'b1. 'b0 = Do not feed input into IDDQ 'b1 = Feed input into IDDQ Bit [0] = Defines the IDDQ_TX select for the clock pads. Default 'b1. 'b0 = Do not feed output into IDDQ 'b1 = Feed output into IDDQ All other bits Reserved.

14.8.81 DRAM Control Register 86 (HW_DRAM_CTL86)

This is a DRAM configuration register.

Address: HW_DRAM_CTL86 – 800E_0000h base + 158h offset = 800E_0158h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																VERSION															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0

HW_DRAM_CTL86 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–0 VERSION	Controller version number. READ-ONLY Holds the EMI version number for this controller. This parameter is read-only.

14.8.82 DRAM Control Register 87 (HW_DRAM_CTL87)

This is a DRAM configuration register.

Address: HW_DRAM_CTL87 – 800E_0000h base + 15Ch offset = 800E_015Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL87 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_0	Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 0. There is a separate dll_ctrl_reg_0_X parameters for each of the slices of data sent on the DFI data bus. Bit [28] = DLL Bypass Control. 'b0 = Normal operational mode. In this mode, the DLL uses dll_ctrl_reg_0_X [14:8] for the read DQS and dll_ctrl_reg_1_X [14:8] for clk_wr. 'b1 = Bypass Mode is on. In this mode, the DLL uses dll_ctrl_reg_0_X [23:15] for the read DQS and dll_ctrl_reg_1_X [23:15] for clk_wr. Bits [23:15] = Holds the read DQS delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1)

HW_DRAM_CTL87 field descriptions (continued)

Field	Description
	Bits [14:8] = Holds the read DQS delay setting when the DLL is operating in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Start Point Control. This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock. Each increment of this field represents 1/128th of a clock cycle. This value MUST be >= 3. All other bits undefined.

14.8.83 DRAM Control Register 88 (HW_DRAM_CTL88)

This is a DRAM configuration register.

Address: HW_DRAM_CTL88 – 800E_0000h base + 160h offset = 800E_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL88 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_1	Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 1. There is a separate dll_ctrl_reg_0_X parameters for each of the slices of data sent on the DFI data bus. Bit [28] = DLL Bypass Control. 'b0 = Normal operational mode. In this mode, the DLL uses dll_ctrl_reg_0_X [14:8] for the read DQS and dll_ctrl_reg_1_X [14:8] for clk_wr. 'b1 = Bypass Mode is on. In this mode, the DLL uses dll_ctrl_reg_0_X [23:15] for the read DQS and dll_ctrl_reg_1_X [23:15] for clk_wr. Bits [23:15] = Holds the read DQS delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Bits [14:8] = Holds the read DQS delay setting when the DLL is operating in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Start Point Control. This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock. Each increment of this field represents 1/128th of a clock cycle. This value MUST be >= 3. All other bits undefined.

14.8.84 DRAM Control Register 89 (HW_DRAM_CTL89)

This is a DRAM configuration register.

Address: HW_DRAM_CTL89 – 800E_0000h base + 164h offset = 800E_0164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_2																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL89 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_2	Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 2. There is a separate dll_ctrl_reg_0_X parameters for each of the slices of data sent on the DFI data bus. Bit [28] = DLL Bypass Control. 'b0 = Normal operational mode. In this mode, the DLL uses dll_ctrl_reg_0_X [14:8] for the read DQS and dll_ctrl_reg_1_X [14:8] for clk_wr. 'b1 = Bypass Mode is on. In this mode, the DLL uses dll_ctrl_reg_0_X [23:15] for the read DQS and dll_ctrl_reg_1_X [23:15] for clk_wr. Bits [23:15] = Holds the read DQS delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Bits [14:8] = Holds the read DQS delay setting when the DLL is operating in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Start Point Control. This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock. Each increment of this field represents 1/128th of a clock cycle. This value MUST be >= 3. All other bits undefined.

14.8.85 DRAM Control Register 90 (HW_DRAM_CTL90)

This is a DRAM configuration register.

Address: HW_DRAM_CTL90 – 800E_0000h base + 168h offset = 800E_0168h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_0_3																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL90 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_0_3	Controls the DLL bypass logic and holds the DLL start point and read DQS delay settings for data slice 3. There is a separate dll_ctrl_reg_0_X parameters for each of the slices of data sent on the DFI data bus. Bit [28] = DLL Bypass Control. 'b0 = Normal operational mode. In this mode, the DLL uses dll_ctrl_reg_0_X [14:8] for the read DQS and dll_ctrl_reg_1_X [14:8] for clk_wr. 'b1 = Bypass Mode is on. In this mode, the DLL uses dll_ctrl_reg_0_X [23:15] for the read DQS and dll_ctrl_reg_1_X [23:15] for clk_wr. Bits [23:15] = Holds the read DQS delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Bits [14:8] = Holds the read DQS delay setting when the DLL is operating in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 1/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Start Point Control. This value is loaded into the DLL at initialization and is the value at which the DLL will begin searching for a lock. Each increment of this field represents 1/128th of a clock cycle. This value MUST be >= 3. All other bits undefined.

14.8.86 DRAM Control Register 91 (HW_DRAM_CTL91)

This is a DRAM configuration register.

Address: HW_DRAM_CTL91 – 800E_0000h base + 16Ch offset = 800E_016Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL91 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_0	Holds the clk_wr settings and the Dll increment value for data slice 0. There is a separate dll_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. Bits [23:15] = Holds the clk_wr delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Each increment of this field represents 1/128th of a clock cycle. Bits [14:8] = Holds the clk_wr delay setting in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Increment Value. This sets the increment used by the DLL when searching for a lock. It is recommended to keep this field small (around 0x4) to keep the steps gradual. All other bits undefined.

14.8.87 DRAM Control Register 92 (HW_DRAM_CTL92)

This is a DRAM configuration register.

Address: HW_DRAM_CTL92 – 800E_0000h base + 170h offset = 800E_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL92 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_1	Holds the clk_wr settings and the Dll increment value for data slice 1. There is a separate dll_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. Bits [23:15] = Holds the clk_wr delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Each increment of this field represents 1/128th of a clock cycle. Bits [14:8] = Holds the clk_wr delay setting in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Increment Value. This sets the increment used by the DLL when searching for a lock. It is recommended to keep this field small (around 0x4) to keep the steps gradual. All other bits undefined.

14.8.88 DRAM Control Register 93 (HW_DRAM_CTL93)

This is a DRAM configuration register.

Address: HW_DRAM_CTL93 – 800E_0000h base + 174h offset = 800E_0174h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL93 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_2	Holds the clk_wr settings and the Dll increment value for data slice 2. There is a separate dll_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. Bits [23:15] = Holds the clk_wr delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Each increment of this field represents 1/128th of a clock cycle. Bits [14:8] = Holds the clk_wr delay setting in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle.

HW_DRAM_CTL93 field descriptions (continued)

Field	Description
	Bits [7:0] = DLL Increment Value. This sets the increment used by the DLL when searching for a lock. It is recommended to keep this field small (around 0x4) to keep the steps gradual. All other bits undefined.

14.8.89 DRAM Control Register 94 (HW_DRAM_CTL94)

This is a DRAM configuration register.

Address: HW_DRAM_CTL94 – 800E_0000h base + 178h offset = 800E_0178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLL_CTRL_REG_1_3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL94 field descriptions

Field	Description
31–0 DLL_CTRL_ REG_1_3	Holds the clk_wr settings and the Dll increment value for data slice 3. There is a separate dll_ctrl_reg_1_X parameter for each of the slices of data sent on the DFI data bus. Bits [23:15] = Holds the clk_wr delay setting when the DLL is operating in bypass mode. (dll_ctrl_reg_0_X [28] = 'b1) Each increment of this field represents 1/128th of a clock cycle. Bits [14:8] = Holds the clk_wr delay setting in normal mode. (dll_ctrl_reg_0_X [28] = 'b0) Typically, this value is 3/4 of a clock cycle. Each increment of this field represents 1/128th of a clock cycle. Bits [7:0] = DLL Increment Value. This sets the increment used by the DLL when searching for a lock. It is recommended to keep this field small (around 0x4) to keep the steps gradual. All other bits undefined.

14.8.90 DRAM Control Register 95 (HW_DRAM_CTL95)

This is a DRAM configuration register.

Address: HW_DRAM_CTL95 – 800E_0000h base + 17Ch offset = 800E_017Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_0_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL95 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 0_0	Reports the lock status and the encoder value for data slice 0. READ-ONLY There is a separate dll_obs_reg_0_X parameter for each of the slices of data sent on the DFI data bus. Bits [31:1] = Reports the DLL encoder value from the master DLL to the slave DLL's. The slaves use this value to set up their delays for the clk_wr and read DQS signals. Bit [0] = DLL Lock Indicator. 'b0 = DLL has not locked. 'b1 = DLL is locked.

14.8.91 DRAM Control Register 96 (HW_DRAM_CTL96)

This is a DRAM configuration register.

Address: HW_DRAM_CTL96 – 800E_0000h base + 180h offset = 800E_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_0_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL96 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 0_1	Reports the lock status and the encoder value for data slice 1. READ-ONLY There is a separate dll_obs_reg_0_X parameter for each of the slices of data sent on the DFI data bus. Bits [31:1] = Reports the DLL encoder value from the master DLL to the slave DLL's. The slaves use this value to set up their delays for the clk_wr and read DQS signals. Bit [0] = DLL Lock Indicator. 'b0 = DLL has not locked. 'b1 = DLL is locked.

14.8.92 DRAM Control Register 97 (HW_DRAM_CTL97)

This is a DRAM configuration register.

Address: HW_DRAM_CTL97 – 800E_0000h base + 184h offset = 800E_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_0_2																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL97 field descriptions

Field	Description
31–0 DLL_OBS_REG_0_2	Reports the lock status and the encoder value for data slice 2. READ-ONLY There is a separate dll_obs_reg_0_X parameter for each of the slices of data sent on the DFI data bus. Bits [31:1] = Reports the DLL encoder value from the master DLL to the slave DLL's. The slaves use this value to set up their delays for the clk_wr and read DQS signals. Bit [0] = DLL Lock Indicator. 'b0 = DLL has not locked. 'b1 = DLL is locked.

14.8.93 DRAM Control Register 98 (HW_DRAM_CTL98)

This is a DRAM configuration register.

Address: HW_DRAM_CTL98 – 800E_0000h base + 188h offset = 800E_0188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_0_3																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL98 field descriptions

Field	Description
31–0 DLL_OBS_REG_0_3	Reports the lock status and the encoder value for data slice 3. READ-ONLY There is a separate dll_obs_reg_0_X parameter for each of the slices of data sent on the DFI data bus. Bits [31:1] = Reports the DLL encoder value from the master DLL to the slave DLL's. The slaves use this value to set up their delays for the clk_wr and read DQS signals. Bit [0] = DLL Lock Indicator. 'b0 = DLL has not locked. 'b1 = DLL is locked.

14.8.94 DRAM Control Register 99 (HW_DRAM_CTL99)

This is a DRAM configuration register.

Address: HW_DRAM_CTL99 – 800E_0000h base + 18Ch offset = 800E_018Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_OBS_REG_0_0																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL99 field descriptions

Field	Description
31–0 PHY_OBS_REG_ 0_0	Controls loopback status, data and masking info for slice 0. READ-ONLY Reports status for the PHY. Bit [24] = Status signal to indicate that the logic gate had to be forced closed. 'b0 = Normal operation 'b1 = Gate close was forced Bits [23:8] = Loopback data. Reports the actual data or expected data, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bits [5:4] = Loopback mask data. Reports the actual data mask or the expected data mask, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bit [1] = Reports status of loopback errors. 'b0 = Last Loopback test had no errors. 'b1 = Last Loopback test contained data errors. Bit [0] = Defines the status of the loopback mode. 'b0 = Not in loopback mode. 'b1 = In Loopback mode. All other bits Reserved.

14.8.95 DRAM Control Register 100 (HW_DRAM_CTL100)

This is a DRAM configuration register.

Address: HW_DRAM_CTL100 – 800E_0000h base + 190h offset = 800E_0190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_OBS_REG_0_1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL100 field descriptions

Field	Description
31–0 PHY_OBS_REG_ 0_1	Controls loopback status, data and masking info for slice 1. READ-ONLY Reports status for the PHY. Bit [24] = Status signal to indicate that the logic gate had to be forced closed. 'b0 = Normal operation 'b1 = Gate close was forced Bits [23:8] = Loopback data. Reports the actual data or expected data, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bits [5:4] = Loopback mask data. Reports the actual data mask or the expected data mask, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bit [1] = Reports status of loopback errors. 'b0 = Last Loopback test had no errors. 'b1 = Last Loopback test contained data errors. Bit [0] = Defines the status of the loopback mode. 'b0 = Not in loopback mode. 'b1 = In Loopback mode. All other bits Reserved.

14.8.96 DRAM Control Register 101 (HW_DRAM_CTL101)

This is a DRAM configuration register.

Address: HW_DRAM_CTL101 – 800E_0000h base + 194h offset = 800E_0194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_OBS_REG_0_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL101 field descriptions

Field	Description
31–0 PHY_OBS_REG_ 0_2	Controls loopback status, data and masking info for slice 2. READ-ONLY Reports status for the PHY. Bit [24] = Status signal to indicate that the logic gate had to be forced closed. 'b0 = Normal operation 'b1 = Gate close was forced Bits [23:8] = Loopback data. Reports the actual data or expected data, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit.

HW_DRAM_CTL101 field descriptions (continued)

Field	Description
	Bits [5:4] = Loopback mask data. Reports the actual data mask or the expected data mask, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bit [1] = Reports status of loopback errors. 'b0 = Last Loopback test had no errors. 'b1 = Last Loopback test contained data errors. Bit [0] = Defines the status of the loopback mode. 'b0 = Not in loopback mode. 'b1 = In Loopback mode. All other bits Reserved.

14.8.97 DRAM Control Register 102 (HW_DRAM_CTL102)

This is a DRAM configuration register.

Address: HW_DRAM_CTL102 – 800E_0000h base + 198h offset = 800E_0198h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PHY_OBS_REG_0_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL102 field descriptions

Field	Description
31–0 PHY_OBS_REG_0_3	Controls loopback status, data and masking info for slice 3. READ-ONLY Reports status for the PHY. Bit [24] = Status signal to indicate that the logic gate had to be forced closed. 'b0 = Normal operation 'b1 = Gate close was forced Bits [23:8] = Loopback data. Reports the actual data or expected data, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bits [5:4] = Loopback mask data. Reports the actual data mask or the expected data mask, depending on the setting of the phy_ctrl_reg_1_X [20] parameter bit. Bit [1] = Reports status of loopback errors. 'b0 = Last Loopback test had no errors. 'b1 = Last Loopback test contained data errors. Bit [0] = Defines the status of the loopback mode. 'b0 = Not in loopback mode. 'b1 = In Loopback mode.

HW_DRAM_CTL102 field descriptions (continued)

Field	Description
	All other bits Reserved.

14.8.98 DRAM Control Register 103 (HW_DRAM_CTL103)

This is a DRAM configuration register.

Address: HW_DRAM_CTL103 – 800E_0000h base + 19Ch offset = 800E_019Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL103 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_0	Reports master DLL info for delay line 0. READ-ONLY.

14.8.99 DRAM Control Register 104 (HW_DRAM_CTL104)

This is a DRAM configuration register.

Address: HW_DRAM_CTL104 – 800E_0000h base + 1A0h offset = 800E_01A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL104 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_0	Reports master DLL info for delay line 0. READ-ONLY.

14.8.100 DRAM Control Register 105 (HW_DRAM_CTL105)

This is a DRAM configuration register.

Address: HW_DRAM_CTL105 – 800E_0000h base + 1A4h offset = 800E_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL105 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_0	Reports master DLL info for delay line 0. READ-ONLY.

14.8.101 DRAM Control Register 106 (HW_DRAM_CTL106)

This is a DRAM configuration register.

Address: HW_DRAM_CTL106 – 800E_0000h base + 1A8h offset = 800E_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL106 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_0	Reports master DLL info for delay line 0. READ-ONLY.

14.8.102 DRAM Control Register 107 (HW_DRAM_CTL107)

This is a DRAM configuration register.

Address: HW_DRAM_CTL107 – 800E_0000h base + 1ACh offset = 800E_01ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_1_0								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL107 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 1_0	Reports master DLL info for delay line 0. READ-ONLY.

14.8.103 DRAM Control Register 108 (HW_DRAM_CTL108)

This is a DRAM configuration register.

Address: HW_DRAM_CTL108 – 800E_0000h base + 1B0h offset = 800E_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL108 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 1_1	Reports master DLL info for delay line 1. READ-ONLY.

14.8.104 DRAM Control Register 109 (HW_DRAM_CTL109)

This is a DRAM configuration register.

Address: HW_DRAM_CTL109 – 800E_0000h base + 1B4h offset = 800E_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL109 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_1	Reports master DLL info for delay line 1. READ-ONLY.

14.8.105 DRAM Control Register 110 (HW_DRAM_CTL110)

This is a DRAM configuration register.

Address: HW_DRAM_CTL110 – 800E_0000h base + 1B8h offset = 800E_01B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL110 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_1	Reports master DLL info for delay line 1. READ-ONLY.

14.8.106 DRAM Control Register 111 (HW_DRAM_CTL111)

This is a DRAM configuration register.

Address: HW_DRAM_CTL111 – 800E_0000h base + 1BCh offset = 800E_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL111 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 1_1	Reports master DLL info for delay line 1. READ-ONLY.

14.8.107 DRAM Control Register 112 (HW_DRAM_CTL112)

This is a DRAM configuration register.

Address: HW_DRAM_CTL112 – 800E_0000h base + 1C0h offset = 800E_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_1_1								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL112 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 1_1	Reports master DLL info for delay line 1. READ-ONLY.

14.8.108 DRAM Control Register 113 (HW_DRAM_CTL113)

This is a DRAM configuration register.

Address: HW_DRAM_CTL113 – 800E_0000h base + 1C4h offset = 800E_01C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL113 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 1_2	Reports master DLL info for delay line 2. READ-ONLY.

14.8.109 DRAM Control Register 114 (HW_DRAM_CTL114)

This is a DRAM configuration register.

Address: HW_DRAM_CTL114 – 800E_0000h base + 1C8h offset = 800E_01C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL114 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 1_2	Reports master DLL info for delay line 2. READ-ONLY.

14.8.110 DRAM Control Register 115 (HW_DRAM_CTL115)

This is a DRAM configuration register.

Address: HW_DRAM_CTL115 – 800E_0000h base + 1CCh offset = 800E_01CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL115 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 1_2	Reports master DLL info for delay line 2. READ-ONLY.

14.8.111 DRAM Control Register 116 (HW_DRAM_CTL116)

This is a DRAM configuration register.

Address: HW_DRAM_CTL116 – 800E_0000h base + 1D0h offset = 800E_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL116 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_2	Reports master DLL info for delay line 2. READ-ONLY.

14.8.112 DRAM Control Register 117 (HW_DRAM_CTL117)

This is a DRAM configuration register.

Address: HW_DRAM_CTL117 – 800E_0000h base + 1D4h offset = 800E_01D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_1_2								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL117 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_1_2	Reports master DLL info for delay line 2. READ-ONLY.

14.8.113 DRAM Control Register 118 (HW_DRAM_CTL118)

This is a DRAM configuration register.

Address: HW_DRAM_CTL118 – 800E_0000h base + 1D8h offset = 800E_01D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL118 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_3	Reports master DLL info for delay line 3. READ-ONLY.

14.8.114 DRAM Control Register 119 (HW_DRAM_CTL119)

This is a DRAM configuration register.

Address: HW_DRAM_CTL119 – 800E_0000h base + 1DCh offset = 800E_01DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL119 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_3	Reports master DLL info for delay line 3. READ-ONLY.

14.8.115 DRAM Control Register 120 (HW_DRAM_CTL120)

This is a DRAM configuration register.

Address: HW_DRAM_CTL120 – 800E_0000h base + 1E0h offset = 800E_01E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL120 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_3	Reports master DLL info for delay line 3. READ-ONLY.

14.8.116 DRAM Control Register 121 (HW_DRAM_CTL121)

This is a DRAM configuration register.

Address: HW_DRAM_CTL121 – 800E_0000h base + 1E4h offset = 800E_01E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_1_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL121 field descriptions

Field	Description
31–0 DLL_OBS_REG_1_3	Reports master DLL info for delay line 3. READ-ONLY.

14.8.117 DRAM Control Register 122 (HW_DRAM_CTL122)

This is a DRAM configuration register.

Address: HW_DRAM_CTL122 – 800E_0000h base + 1E8h offset = 800E_01E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_1_3								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL122 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.

HW_DRAM_CTL122 field descriptions (continued)

Field	Description
8–0 DLL_OBS_REG_ 1_3	Reports master DLL info for delay line 3. READ-ONLY.

14.8.118 DRAM Control Register 123 (HW_DRAM_CTL123)

This is a DRAM configuration register.

Address: HW_DRAM_CTL123 – 800E_0000h base + 1ECh offset = 800E_01ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_0																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL123 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_0	Reports the read DQS delay value for data slice 0. READ-ONLY.

14.8.119 DRAM Control Register 124 (HW_DRAM_CTL124)

This is a DRAM configuration register.

Address: HW_DRAM_CTL124 – 800E_0000h base + 1F0h offset = 800E_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_0																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL124 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_0	Reports the read DQS delay value for data slice 0. READ-ONLY.

14.8.120 DRAM Control Register 125 (HW_DRAM_CTL125)

This is a DRAM configuration register.

Address: HW_DRAM_CTL125 – 800E_0000h base + 1F4h offset = 800E_01F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL125 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_0	Reports the read DQS delay value for data slice 0. READ-ONLY.

14.8.121 DRAM Control Register 126 (HW_DRAM_CTL126)

This is a DRAM configuration register.

Address: HW_DRAM_CTL126 – 800E_0000h base + 1F8h offset = 800E_01F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL126 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_0	Reports the read DQS delay value for data slice 0. READ-ONLY.

14.8.122 DRAM Control Register 127 (HW_DRAM_CTL127)

This is a DRAM configuration register.

Address: HW_DRAM_CTL127 – 800E_0000h base + 1FCh offset = 800E_01FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_2_0								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL127 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_2_0	Reports the read DQS delay value for data slice 0. READ-ONLY.

14.8.123 DRAM Control Register 128 (HW_DRAM_CTL128)

This is a DRAM configuration register.

Address: HW_DRAM_CTL128 – 800E_0000h base + 200h offset = 800E_0200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL128 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_1	Reports the read DQS delay value for data slice 1. READ-ONLY.

14.8.124 DRAM Control Register 129 (HW_DRAM_CTL129)

This is a DRAM configuration register.

Address: HW_DRAM_CTL129 – 800E_0000h base + 204h offset = 800E_0204h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL129 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_1	Reports the read DQS delay value for data slice 1. READ-ONLY.

14.8.125 DRAM Control Register 130 (HW_DRAM_CTL130)

This is a DRAM configuration register.

Address: HW_DRAM_CTL130 – 800E_0000h base + 208h offset = 800E_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL130 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_1	Reports the read DQS delay value for data slice 1. READ-ONLY.

14.8.126 DRAM Control Register 131 (HW_DRAM_CTL131)

This is a DRAM configuration register.

Address: HW_DRAM_CTL131 – 800E_0000h base + 20Ch offset = 800E_020Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL131 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_1	Reports the read DQS delay value for data slice 1. READ-ONLY.

14.8.127 DRAM Control Register 132 (HW_DRAM_CTL132)

This is a DRAM configuration register.

Address: HW_DRAM_CTL132 – 800E_0000h base + 210h offset = 800E_0210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_2_1								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL132 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_2_1	Reports the read DQS delay value for data slice 1. READ-ONLY.

14.8.128 DRAM Control Register 133 (HW_DRAM_CTL133)

This is a DRAM configuration register.

Address: HW_DRAM_CTL133 – 800E_0000h base + 214h offset = 800E_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL133 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_2	Reports the read DQS delay value for data slice 2. READ-ONLY.

14.8.129 DRAM Control Register 134 (HW_DRAM_CTL134)

This is a DRAM configuration register.

Address: HW_DRAM_CTL134 – 800E_0000h base + 218h offset = 800E_0218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_2																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL134 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_2	Reports the read DQS delay value for data slice 2. READ-ONLY.

14.8.130 DRAM Control Register 135 (HW_DRAM_CTL135)

This is a DRAM configuration register.

Address: HW_DRAM_CTL135 – 800E_0000h base + 21Ch offset = 800E_021Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_2																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL135 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_2	Reports the read DQS delay value for data slice 2. READ-ONLY.

14.8.131 DRAM Control Register 136 (HW_DRAM_CTL136)

This is a DRAM configuration register.

Address: HW_DRAM_CTL136 – 800E_0000h base + 220h offset = 800E_0220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_2																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL136 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_2	Reports the read DQS delay value for data slice 2. READ-ONLY.

14.8.132 DRAM Control Register 137 (HW_DRAM_CTL137)

This is a DRAM configuration register.

Address: HW_DRAM_CTL137 – 800E_0000h base + 224h offset = 800E_0224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_2_2								
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL137 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_2_2	Reports the read DQS delay value for data slice 2. READ-ONLY.

14.8.133 DRAM Control Register 138 (HW_DRAM_CTL138)

This is a DRAM configuration register.

Address: HW_DRAM_CTL138 – 800E_0000h base + 228h offset = 800E_0228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL138 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_3	Reports the read DQS delay value for data slice 3. READ-ONLY.

14.8.134 DRAM Control Register 139 (HW_DRAM_CTL139)

This is a DRAM configuration register.

Address: HW_DRAM_CTL139 – 800E_0000h base + 22Ch offset = 800E_022Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL139 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 2_3	Reports the read DQS delay value for data slice 3. READ-ONLY.

14.8.135 DRAM Control Register 140 (HW_DRAM_CTL140)

This is a DRAM configuration register.

Address: HW_DRAM_CTL140 – 800E_0000h base + 230h offset = 800E_0230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL140 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_3	Reports the read DQS delay value for data slice 3. READ-ONLY.

14.8.136 DRAM Control Register 141 (HW_DRAM_CTL141)

This is a DRAM configuration register.

Address: HW_DRAM_CTL141 – 800E_0000h base + 234h offset = 800E_0234h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_2_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL141 field descriptions

Field	Description
31–0 DLL_OBS_REG_2_3	Reports the read DQS delay value for data slice 3. READ-ONLY.

14.8.137 DRAM Control Register 142 (HW_DRAM_CTL142)

This is a DRAM configuration register.

Address: HW_DRAM_CTL142 – 800E_0000h base + 238h offset = 800E_0238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_2_3								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL142 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.

HW_DRAM_CTL142 field descriptions (continued)

Field	Description
8–0 DLL_OBS_REG_ 2_3	Reports the read DQS delay value for data slice 3. READ-ONLY.

14.8.138 DRAM Control Register 143 (HW_DRAM_CTL143)

This is a DRAM configuration register.

Address: HW_DRAM_CTL143 – 800E_0000h base + 23Ch offset = 800E_023Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_0																															
W																																
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL143 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_0	Reports the clk_wr delay value for data slice 0. READ-ONLY.

14.8.139 DRAM Control Register 144 (HW_DRAM_CTL144)

This is a DRAM configuration register.

Address: HW_DRAM_CTL144 – 800E_0000h base + 240h offset = 800E_0240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_0																															
W																																
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL144 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_0	Reports the clk_wr delay value for data slice 0. READ-ONLY.

14.8.140 DRAM Control Register 145 (HW_DRAM_CTL145)

This is a DRAM configuration register.

Address: HW_DRAM_CTL145 – 800E_0000h base + 244h offset = 800E_0244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_0																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL145 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_0	Reports the clk_wr delay value for data slice 0. READ-ONLY.

14.8.141 DRAM Control Register 146 (HW_DRAM_CTL146)

This is a DRAM configuration register.

Address: HW_DRAM_CTL146 – 800E_0000h base + 248h offset = 800E_0248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_0																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL146 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_0	Reports the clk_wr delay value for data slice 0. READ-ONLY.

14.8.142 DRAM Control Register 147 (HW_DRAM_CTL147)

This is a DRAM configuration register.

Address: HW_DRAM_CTL147 – 800E_0000h base + 24Ch offset = 800E_024Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_3_0								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL147 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 3_0	Reports the clk_wr delay value for data slice 0. READ-ONLY.

14.8.143 DRAM Control Register 148 (HW_DRAM_CTL148)

This is a DRAM configuration register.

Address: HW_DRAM_CTL148 – 800E_0000h base + 250h offset = 800E_0250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL148 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_1	Reports the clk_wr delay value for data slice 1. READ-ONLY.

14.8.144 DRAM Control Register 149 (HW_DRAM_CTL149)

This is a DRAM configuration register.

Address: HW_DRAM_CTL149 – 800E_0000h base + 254h offset = 800E_0254h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL149 field descriptions

Field	Description
31–0 DLL_OBS_REG_3_1	Reports the clk_wr delay value for data slice 1. READ-ONLY.

14.8.145 DRAM Control Register 150 (HW_DRAM_CTL150)

This is a DRAM configuration register.

Address: HW_DRAM_CTL150 – 800E_0000h base + 258h offset = 800E_0258h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL150 field descriptions

Field	Description
31–0 DLL_OBS_REG_3_1	Reports the clk_wr delay value for data slice 1. READ-ONLY.

14.8.146 DRAM Control Register 151 (HW_DRAM_CTL151)

This is a DRAM configuration register.

Address: HW_DRAM_CTL151 – 800E_0000h base + 25Ch offset = 800E_025Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL151 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_1	Reports the clk_wr delay value for data slice 1. READ-ONLY.

14.8.147 DRAM Control Register 152 (HW_DRAM_CTL152)

This is a DRAM configuration register.

Address: HW_DRAM_CTL152 – 800E_0000h base + 260h offset = 800E_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_3_1								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL152 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 3_1	Reports the clk_wr delay value for data slice 1. READ-ONLY.

14.8.148 DRAM Control Register 153 (HW_DRAM_CTL153)

This is a DRAM configuration register.

Address: HW_DRAM_CTL153 – 800E_0000h base + 264h offset = 800E_0264h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL153 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_2	Reports the clk_wr delay value for data slice 2. READ-ONLY.

14.8.149 DRAM Control Register 154 (HW_DRAM_CTL154)

This is a DRAM configuration register.

Address: HW_DRAM_CTL154 – 800E_0000h base + 268h offset = 800E_0268h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_2																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL154 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_2	Reports the clk_wr delay value for data slice 2. READ-ONLY.

14.8.150 DRAM Control Register 155 (HW_DRAM_CTL155)

This is a DRAM configuration register.

Address: HW_DRAM_CTL155 – 800E_0000h base + 26Ch offset = 800E_026Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_2																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL155 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_2	Reports the clk_wr delay value for data slice 2. READ-ONLY.

14.8.151 DRAM Control Register 156 (HW_DRAM_CTL156)

This is a DRAM configuration register.

Address: HW_DRAM_CTL156 – 800E_0000h base + 270h offset = 800E_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL156 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_2	Reports the clk_wr delay value for data slice 2. READ-ONLY.

14.8.152 DRAM Control Register 157 (HW_DRAM_CTL157)

This is a DRAM configuration register.

Address: HW_DRAM_CTL157 – 800E_0000h base + 274h offset = 800E_0274h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OBSOLETE																RSVD1							DLL_OBS_REG_3_2								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL157 field descriptions

Field	Description
31–16 OBSOLETE	Always write zeroes to this field.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 3_2	Reports the clk_wr delay value for data slice 2. READ-ONLY.

14.8.153 DRAM Control Register 158 (HW_DRAM_CTL158)

This is a DRAM configuration register.

Address: HW_DRAM_CTL158 – 800E_0000h base + 278h offset = 800E_0278h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL158 field descriptions

Field	Description
31–0 DLL_OBS_REG_3_3	Reports the clk_wr delay value for data slice 3. READ-ONLY.

14.8.154 DRAM Control Register 159 (HW_DRAM_CTL159)

This is a DRAM configuration register.

Address: HW_DRAM_CTL159 – 800E_0000h base + 27Ch offset = 800E_027Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL159 field descriptions

Field	Description
31–0 DLL_OBS_REG_3_3	Reports the clk_wr delay value for data slice 3. READ-ONLY.

14.8.155 DRAM Control Register 160 (HW_DRAM_CTL160)

This is a DRAM configuration register.

Address: HW_DRAM_CTL160 – 800E_0000h base + 280h offset = 800E_0280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL160 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_3	Reports the clk_wr delay value for data slice 3. READ-ONLY.

14.8.156 DRAM Control Register 161 (HW_DRAM_CTL161)

This is a DRAM configuration register.

Address: HW_DRAM_CTL161 – 800E_0000h base + 284h offset = 800E_0284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_OBS_REG_3_3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL161 field descriptions

Field	Description
31–0 DLL_OBS_REG_ 3_3	Reports the clk_wr delay value for data slice 3. READ-ONLY.

14.8.157 DRAM Control Register 162 (HW_DRAM_CTL162)

This is a DRAM configuration register.

Address: HW_DRAM_CTL162 – 800E_0000h base + 288h offset = 800E_0288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3					W2R_ SAMECS_ DLY			RSVD2					W2R_ DIFFCS_ DLY			RSVD1							DLL_OBS_REG_3_3								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL162 field descriptions

Field	Description
31–27 RSVD3	Always write zeroes to this field.
26–24 W2R_SAMECS_ DLY	Additional delay to insert between writes and reads to the same chip select. Defines the number of additional clocks of delay to insert from a write command to a read command to the same chip select.

HW_DRAM_CTL162 field descriptions (continued)

Field	Description
23–19 RSVD2	Always write zeroes to this field.
18–16 W2R_DIFFCS_ DLY	Additional delay to insert between writes and reads to different chip selects. Defines the number of additional clocks of delay to insert from a write command to one chip select to a read command to a different chip select.
15–9 RSVD1	Always write zeroes to this field.
8–0 DLL_OBS_REG_ 3_3	Reports the clk_wr delay value for data slice 3. READ-ONLY.

14.8.158 DRAM Control Register 163 (HW_DRAM_CTL163)

This is a DRAM configuration register.

Address: HW_DRAM_CTL163 – 800E_0000h base + 28Ch offset = 800E_028Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DLL_RST_ADJ_DLY								RSVD3				WRLAT_ADJ				RSVD2				RDLAT_ADJ				RSVD1				DRAM_CLASS			
W									▨								▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL163 field descriptions

Field	Description
31–24 DLL_RST_ADJ_ DLY	Minimum number of cycles after setting master delay in DLL until reset is released. Specifies the minimum number of cycles after the master delay value is programmed before the DLL reset may be asserted.
23–20 RSVD3	Always write zeroes to this field.
19–16 WRLAT_ADJ	Adjustment value for PHY write timing. Adjusts the relative timing between DFI write commands and the dfi_wrdata_en signal to conform to PHY timing requirements. When this parameter is programmed to 0x0, dfi_wrdata_en will assert on the same cycle as the dfi_address. This parameter only affects the DFI.
15–12 RSVD2	Always write zeroes to this field.
11–8 RDLAT_ADJ	Adjustment value for PHY read timing. Adjusts the relative timing between DFI read commands and the dfi_rddata_en signal to conform to PHY timing requirements. When this parameter is programmed to 0x0, dfi_rddata_en will assert one cycle after the dfi_address. This parameter only affects the DFI.

HW_DRAM_CTL163 field descriptions (continued)

Field	Description
7–4 RSVD1	Always write zeroes to this field.
3–0 DRAM_CLASS	Defines the mode of operation of the controller. Selects the mode of operation for the EMI. 'b0000 = DDR1 'b0001 = DDR1 with Mobile (LPDDR1) 'b0100 = DDR2 All other settings reserved.

14.8.159 DRAM Control Register 164 (HW_DRAM_CTL164)

This is a DRAM configuration register.

Address: HW_DRAM_CTL164 – 800E_0000h base + 290h offset = 800E_0290h

Bit	31–24	23–18	17–16	15–8	7–0
R	OBSOLETE	RSVD1	▨	▨	TMOD
W	▨	▨	INT_ACK	INT_ACK	TMOD
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0	0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

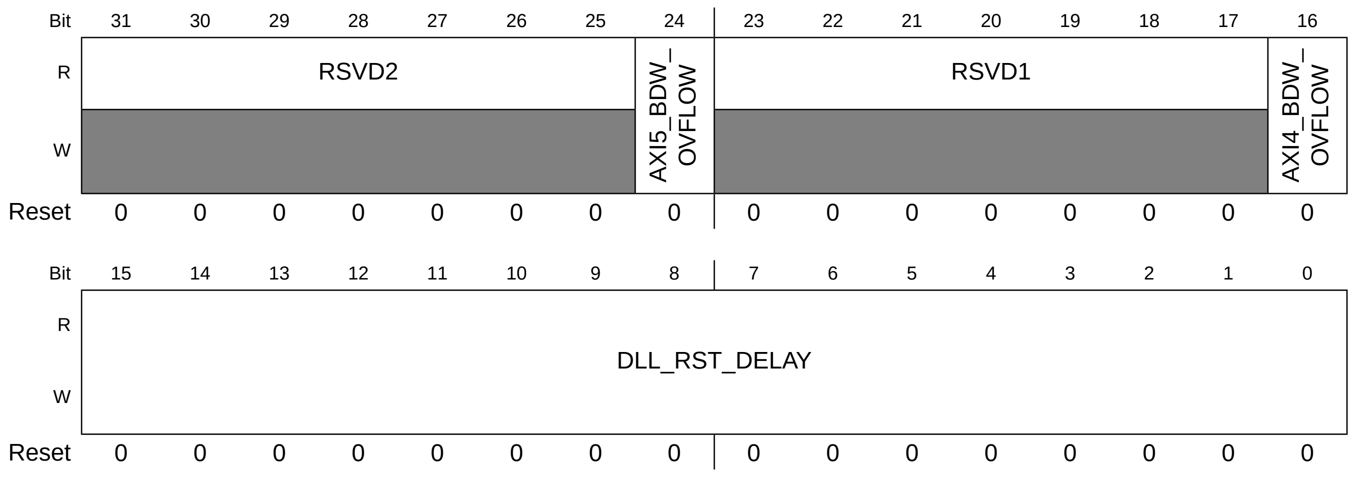
HW_DRAM_CTL164 field descriptions

Field	Description
31–24 OBSOLETE	Always write zeroes to this field.
23–18 RSVD1	Always write zeroes to this field.
17–8 INT_ACK	Clear mask of the INT_STATUS parameter. WRITE-ONLY Controls the clearing of the int_status parameter. If any of the int_ack bits are set to 'b1, the corresponding bit in the int_status parameter will be cleared to 'b0. Any int_ack bits cleared to 'b0 will not alter the corresponding bit in the int_status parameter. This parameter will always read back as 0x0.
7–0 TMOD	Number of clock cycles after MRS command and before any other command. Defines the number of cycles of wait time after a mode register write to any non-mode register write command.

14.8.160 DRAM Control Register 171 (HW_DRAM_CTL171)

This is a DRAM configuration register.

Address: HW_DRAM_CTL171 – 800E_0000h base + 2ACh offset = 800E_02ACh


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSVD2
AXI5_BDW_OVFLOW
RSVD1
AXI4_BDW_OVFLOW
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
DLL_RST_DELAY
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

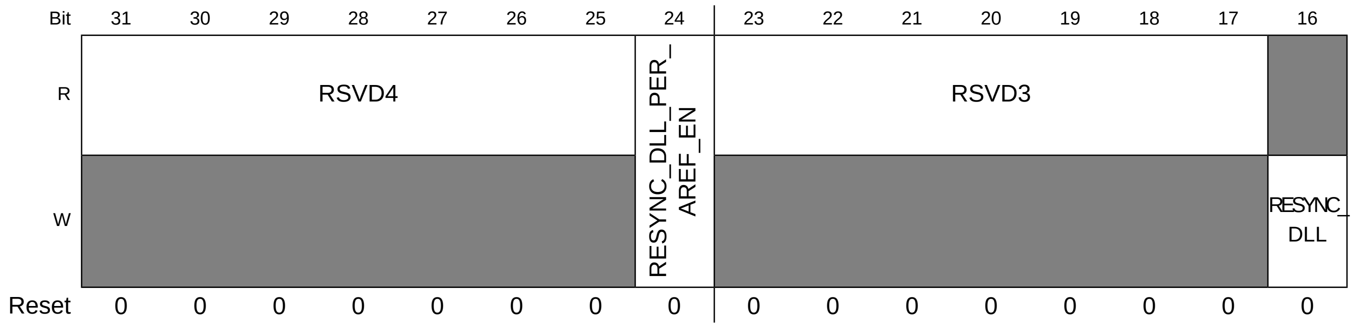
HW_DRAM_CTL171 field descriptions

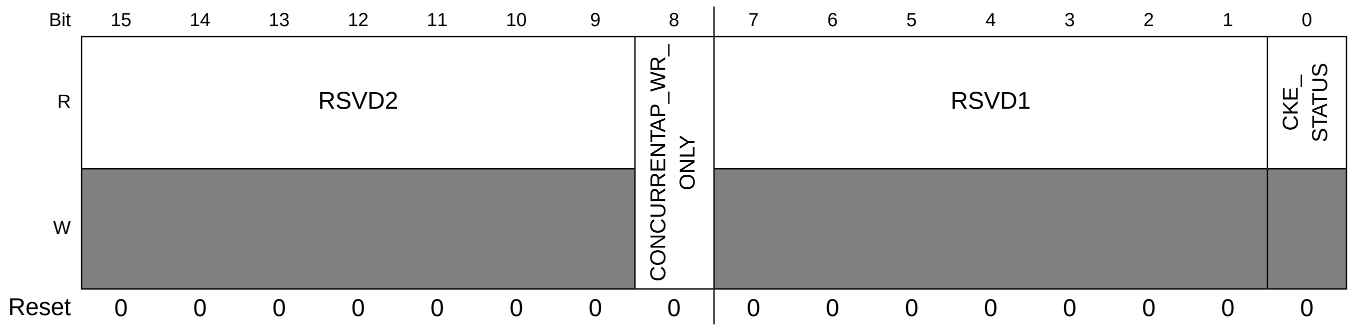
Field	Description
31–25 RSVD2	Always write zeroes to this field.
24 AXI5_BDW_ OVFLOW	Port 5 behavior when bandwidth maximized.
23–17 RSVD1	Always write zeroes to this field.
16 AXI4_BDW_ OVFLOW	Port 4 behavior when bandwidth maximized.
15–0 DLL_RST_ DELAY	Minimum number of cycles required for DLL reset. Sets the number of cycles that the reset must be held asserted for the DLL.

14.8.161 DRAM Control Register 172 (HW_DRAM_CTL172)

This is a DRAM configuration register.

Address: HW_DRAM_CTL172 – 800E_0000h base + 2B0h offset = 800E_02B0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSVD4
RESYNC_DLL_PER_AREF_EN
RSVD3
RESYNC_DLL
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0



HW_DRAM_CTL172 field descriptions

Field	Description
31–25 RSVD4	Always write zeroes to this field.
24 RESYNC_DLL_ PER_AREF_EN	Enables automatic DLL resyncs after every refresh. Enables an automatic re-synchronization of the DLL after every refresh.
23–17 RSVD3	Always write zeroes to this field.
16 RESYNC_DLL	Initiate a DLL resync. WRITE-ONLY Initiates a re-synchronization of the DLL. This parameter is write-only.
15–9 RSVD2	Always write zeroes to this field.
8 CONCURRENTAP_ WR_ONLY	Limit concurrent auto-precharge by waiting for the write recovery time, tWR, before issuing a read. Limit concurrent auto-precharge by waiting for the write recovery time, tWR, to complete after a write before issuing a read. 'b0 = Do not restrict concurrent auto-precharge. 'b1 = Wait tWR after a write before issuing a read.
7–1 RSVD1	Always write zeroes to this field.
0 CKE_STATUS	Register access to cke_status signal. This value also indicates the inverted state of the CKE pin on the external memory bus. READ-ONLY Provides the value of the cke_status signal in a parameter. This parameter is read-only.

14.8.162 DRAM Control Register 173 (HW_DRAM_CTL173)

This is a DRAM configuration register.

Address: HW_DRAM_CTL173 – 800E_0000h base + 2B4h offset = 800E_02B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4					AXI4_W_PRIORITY			RSVD3					AXI4_R_PRIORITY			RSVD2						AXI5_FIFO_TYPE_REG		RSVD1						AXI4_FIFO_TYPE_REG	
W	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨	▨	▨			▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL173 field descriptions

Field	Description
31–27 RSVD4	Always write zeroes to this field.
26–24 AXI4_W_ PRIORITY	Priority of write cmds from AXI port 4.
23–19 RSVD3	Always write zeroes to this field.
18–16 AXI4_R_ PRIORITY	Priority of read cmds from AXI port 4.
15–10 RSVD2	Always write zeroes to this field.
9–8 AXI5_FIFO_ TYPE_REG	Clock domain relativity between AXI port 5 and core logic.
7–2 RSVD1	Always write zeroes to this field.
1–0 AXI4_FIFO_ TYPE_REG	Clock domain relativity between AXI port 4 and core logic.

14.8.163 DRAM Control Register 174 (HW_DRAM_CTL174)

This is a DRAM configuration register.

Address: HW_DRAM_CTL174 – 800E_0000h base + 2B8h offset = 800E_02B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4					R2R_SAMECS_DLY			RSVD3					R2R_DIFFCS_DLY			RSVD2					AXI5_W_PRIORITY			RSVD1					AXI5_R_PRIORITY		
W	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL174 field descriptions

Field	Description
31–27 RSVD4	Always write zeroes to this field.
26–24 R2R_SAMECS_ DLY	Additional delay to insert between reads and reads to the same chip select. Defines the number of additional clocks of delay to insert between two read commands to the same chip select.
23–19 RSVD3	Always write zeroes to this field.
18–16 R2R_DIFFCS_ DLY	Additional delay to insert between reads and reads to different chip selects. Defines the number of additional clocks of delay to insert from a read command to one chip select to a read command to a different chip select.
15–11 RSVD2	Always write zeroes to this field.
10–8 AXI5_W_ PRIORITY	Priority of write cmds from AXI port 5.
7–3 RSVD1	Always write zeroes to this field.
2–0 AXI5_R_ PRIORITY	Priority of read cmds from AXI port 5.

14.8.164 DRAM Control Register 175 (HW_DRAM_CTL175)

This is a DRAM configuration register.

Address: HW_DRAM_CTL175 – 800E_0000h base + 2BCh offset = 800E_02BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4					W2W_ DIFFCS_ DLY			RSVD3					TBST_INT_ INTERVAL			RSVD2					R2W_ SAMECS_ DLY			RSVD1					R2W_ DIFFCS_ DLY		
W	▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨	▨				▨	▨	▨	▨	▨			
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL175 field descriptions

Field	Description
31–27 RSVD4	Always write zeroes to this field.
26–24 W2W_DIFFCS_ DLY	Additional delay to insert between writes and writes to different chip selects. Defines the number of additional clocks of delay to insert from a write command to one chip select to a write command to a different chip select.

HW_DRAM_CTL175 field descriptions (continued)

Field	Description
23–19 RSVD3	Always write zeroes to this field.
18–16 TBST_INT_ INTERVAL	DRAM burst interrupt interval in cycles. Defines the burst interrupt interval. This parameter is only relevant if the burst has not completed. This value is loaded into a parameter when a burst is issued and another command may only interrupt the current burst when this counter value hits 0. If the counter value hits 0 and the burst has not completed, the counter will be reset with the tbst_int_interval value. If a command is in progress and the burst has not completed, another command may only be issued on cycles after the parameter tccd value cycles have elapsed since the last CAS command and this counter value hits 0. For example, if the burst length is 8, tccd is 2, tbst_int_interval is 2 and a CAS command was issued on cycle 0, another CAS command could interrupt the current burst on cycle 2. After cycle 3, the current burst will complete and this parameter would not be relevant. If instead the tbst_int_interval was 1 for the same system, then the command could interrupt on cycles 2 or 3.
15–11 RSVD2	Always write zeroes to this field.
10–8 R2W_SAMECS_ DLY	Additional delay to insert between reads and writes to the same chip select. Defines the number of additional clocks of delay to insert from a read command to a write command to the same chip select.
7–3 RSVD1	Always write zeroes to this field.
2–0 R2W_DIFFCS_ DLY	Additional delay to insert between reads and writes to different chip selects. Defines the number of additional clocks of delay to insert from a read command to one chip select to a write command to a different chip select.

14.8.165 DRAM Control Register 176 (HW_DRAM_CTL176)

This is a DRAM configuration register.

Address: HW_DRAM_CTL176 – 800E_0000h base + 2C0h offset = 800E_02C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4				ADD_ODT_CLK_ SAMETYPE_DIFFCS				RSVD3				ADD_ODT_CLK_DIFFTYPE_ SAMECS			
W	▨	▨	▨	▨					▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				ADD_ODT_CLK_DIFFTYPE_ DIFFCS				RSVD1					W2W_SAMECS_DLY		
W	▨	▨	▨	▨					▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL176 field descriptions

Field	Description
31–28 RSVD4	Always write zeroes to this field.
27–24 ADD_ODT_CLK_ SAMETYPE_ DIFFCS	Additional delay to insert between same transaction types to different chip selects to meet ODT timing requirements. Defines the number of additional clocks of delay to insert between commands of the same type (read to read, write to write) to different chip selects to meet ODT timing requirements.
23–20 RSVD3	Always write zeroes to this field.
19–16 ADD_ODT_CLK_ DIFFTYPE_ SAMECS	Additional delay to insert between different transaction types to the same chip select to meet ODT timing requirements. Defines the number of additional clocks of delay to insert between commands of different types (read to write, write to read) to the same chip select to meet ODT timing requirements.
15–12 RSVD2	Always write zeroes to this field.
11–8 ADD_ODT_CLK_ DIFFTYPE_ DIFFCS	Additional delay to insert between different transaction types to different chip selects to meet ODT timing requirements. Defines the number of additional clocks of delay to insert between commands of different types (read to write, write to read) to different chip selects to meet ODT timing requirements.
7–3 RSVD1	Always write zeroes to this field.
2–0 W2W_SAMECS_ DLY	Additional delay to insert between writes and writes to the same chip select. Defines the number of additional clocks of delay to insert between two write commands to the same chip select.

14.8.166 DRAM Control Register 177 (HW_DRAM_CTL177)

This is a DRAM configuration register.

Address: HW_DRAM_CTL177 – 800E_0000h base + 2C4h offset = 800E_02C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD4			TCCD					RSVD3				TRP_AB				RSVD2				CKSRX				RSVD1				CKSRE			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL177 field descriptions

Field	Description
31–29 RSVD4	Always write zeroes to this field.
28–24 TCCD	DRAM CAS-to-CAS parameter in cycles. Defines the minimum delay between CAS commands, in cycles. This value is loaded into a counter when a burst is issued and a new command may be issued when the counter reaches 0.
23–20 RSVD3	Always write zeroes to this field.
19–16 TRP_AB	DRAM TRP All Bank parameter in cycles. Defines the DRAM TRP time for all banks, in cycles.
15–12 RSVD2	Always write zeroes to this field.
11–8 CKSRX	Clock stable delay on self-refresh exit. Sets the number of cycles to hold the clock stable before exiting self-refresh mode. The clock will run for a minimum of cksrx cycles before CKE rises.
7–4 RSVD1	Always write zeroes to this field.
3–0 CKSRE	Clock hold delay on self-refresh entry. Sets the number of cycles to hold the clock stable after entering self-refresh mode. The clock will run for a minimum of cksre cycles after CKE falls.

14.8.167 DRAM Control Register 178 (HW_DRAM_CTL178)

This is a DRAM configuration register.

Address: HW_DRAM_CTL178 – 800E_0000h base + 2C8h offset = 800E_02C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4	AXI5_BDW							RSVD3	AXI4_CURRENT_BDW						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2	AXI4_BDW							RSVD1			TCKESR				
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL178 field descriptions

Field	Description
31 RSVD4	Always write zeroes to this field.
30–24 AXI5_BDW	Maximum bandwidth percentage for port 5.
23 RSVD3	Always write zeroes to this field.
22–16 AXI4_ CURRENT_BDW	Current bandwidth usage percentage for port 4. READ-ONLY.
15 RSVD2	Always write zeroes to this field.
14–8 AXI4_BDW	Maximum bandwidth percentage for port 4.
7–5 RSVD1	Always write zeroes to this field.
4–0 TCKESR	Minimum CKE low pulse width during a self-refresh. Defines the minimum number of cycles that CKE must be held low during self-refresh. pulse width, in cycles. If the memory specification does not define a tckesr, then the tckesr parameter should be programmed with the tcke value.

14.8.168 DRAM Control Register 179 (HW_DRAM_CTL179)

This is a DRAM configuration register.

Address: HW_DRAM_CTL179 – 800E_0000h base + 2CCh offset = 800E_02CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3								RSVD2		TDFI_PHYUPD_TYPE1[21:16]					
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TDFI_PHYUPD_TYPE1[15:8]								RSVD1	AXI5_CURRENT_BDW						
W									▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL179 field descriptions

Field	Description
31–24 RSVD3	Always write zeroes to this field.
23–22 RSVD2	Always write zeroes to this field.
21–8 TDFI_PHYUPD_ TYPE1	Holds the DFI tPHYUPD_TYPE1 timing parameter. Holds the DFI tphyupd_typeX timing parameter. This parameter is read-only.
7 RSVD1	Always write zeroes to this field.
6–0 AXI5_ CURRENT_BDW	Current bandwidth usage percentage for port 5. READ-ONLY.

14.8.169 DRAM Control Register 180 (HW_DRAM_CTL180)

This is a DRAM configuration register.

Address: HW_DRAM_CTL180 – 800E_0000h base + 2D0h offset = 800E_02D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2		TDFI_PHYUPD_TYPE3														RSVD1		TDFI_PHYUPD_TYPE2													
W	▨																▨															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL180 field descriptions

Field	Description
31–30 RSVD2	Always write zeroes to this field.
29–16 TDFI_PHYUPD_ TYPE3	Holds the DFI tPHYUPD_TYPE3 timing parameter. Holds the DFI tphyupd_typeX timing parameter. This parameter is read-only.
15–14 RSVD1	Always write zeroes to this field.
13–0 TDFI_PHYUPD_ TYPE2	Holds the DFI tPHYUPD_TYPE2 timing parameter. Holds the DFI tphyupd_typeX timing parameter. This parameter is read-only.

14.8.170 DRAM Control Register 181 (HW_DRAM_CTL181)

This is a DRAM configuration register.

Address: HW_DRAM_CTL181 – 800E_0000h base + 2D4h offset = 800E_02D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR0_DATA_1														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR0_DATA_0														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL181 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR0_DATA_1	MRS data to program to memory mode register 0 for chip select 1. Holds the memory mode register 0 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.For DDR2 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. For LPDDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR0_DATA_0	MRS data to program to memory mode register 0 for chip select 0. Holds the memory mode register 0 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this EMI:

HW_DRAM_CTL181 field descriptions (continued)

Field	Description
	For DDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.For DDR2 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. For LPDDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.171 DRAM Control Register 182 (HW_DRAM_CTL182)

This is a DRAM configuration register.

Address: HW_DRAM_CTL182 – 800E_0000h base + 2D8h offset = 800E_02D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR0_DATA_3														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR0_DATA_2														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL182 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR0_DATA_3	MRS data to program to memory mode register 0 for chip select 3. Holds the memory mode register 0 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.For DDR2 memories: The EMI does not support interleaving and therefore the A3 bit

HW_DRAM_CTL182 field descriptions (continued)

Field	Description
	should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. For LPDDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR0_DATA_2	MRS data to program to memory mode register 0 for chip select 2. Holds the memory mode register 0 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. This parameter correlates to the memory mode register (MR). The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010.For DDR2 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. In addition, the DLL Reset bit (A8) will be ignored in favor of an internal state machine that sets the DLL Reset bit during initialization. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. For LPDDR1 memories: The EMI does not support interleaving and therefore the A3 bit should be cleared to 'b0. Also, the EMI only supports a burst length of 4 and therefore the bits A2:A0 should be set to 'b010. This data will be programmed into the memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.172 DRAM Control Register 183 (HW_DRAM_CTL183)

This is a DRAM configuration register.

Address: HW_DRAM_CTL183 – 800E_0000h base + 2DCh offset = 800E_02DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR1_DATA_1														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR1_DATA_0														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL183 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR1_DATA_1	Data to program into memory mode register 1 for chip select 1. Holds the memory mode register 1 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter correlates to the extended memory mode register (EMR). For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The EMI does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000. For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR1_DATA_0	Data to program into memory mode register 1 for chip select 0. Holds the memory mode register 1 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter correlates to the extended memory mode register (EMR). For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The EMI does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000. For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.173 DRAM Control Register 184 (HW_DRAM_CTL184)

This is a DRAM configuration register.

Address: HW_DRAM_CTL184 – 800E_0000h base + 2E0h offset = 800E_02E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR1_DATA_3														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR1_DATA_2														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL184 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR1_DATA_3	Data to program into memory mode register 1 for chip select 3. Holds the memory mode register 1 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter correlates to the extended memory mode register (EMR). For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The EMI does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000. For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR1_DATA_2	Data to program into memory mode register 1 for chip select 2. Holds the memory mode register 1 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter correlates to the extended memory mode register (EMR). For DDR2 memories: This parameter correlates to the extended memory mode register 1 (EMR1). The EMI does not support additive latency and therefore the A5:A3 bits should be cleared to 'b000. For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.174 DRAM Control Register 185 (HW_DRAM_CTL185)

This is a DRAM configuration register.

Address: HW_DRAM_CTL185 – 800E_0000h base + 2E4h offset = 800E_02E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR2_DATA_1														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR2_DATA_0														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL185 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR2_DATA_1	Data to program into memory mode register 2 for chip select 1. Holds the memory mode register 2 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2). For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR2_DATA_0	Data to program into memory mode register 2 for chip select 0. Holds the memory mode register 2 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2). For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.175 DRAM Control Register 186 (HW_DRAM_CTL186)

This is a DRAM configuration register.

Address: HW_DRAM_CTL186 – 800E_0000h base + 2E8h offset = 800E_02E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR2_DATA_3														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR2_DATA_2														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL186 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR2_DATA_3	Data to program into memory mode register 2 for chip select 3. Holds the memory mode register 2 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2). For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR2_DATA_2	Data to program into memory mode register 2 for chip select 2. Holds the memory mode register 2 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 2 (EMR2). For LPDDR1 memories: This parameter correlates to the extended mode register (EMR). This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.176 DRAM Control Register 187 (HW_DRAM_CTL187)

This is a DRAM configuration register.

Address: HW_DRAM_CTL187 – 800E_0000h base + 2ECh offset = 800E_02ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR3_DATA_1														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR3_DATA_0														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL187 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR3_DATA_1	Data to program into memory mode register 3 for chip select 1. Holds the memory mode register 3 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3). For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR3_DATA_0	Data to program into memory mode register 3 for chip select 0. Holds the memory mode register 3 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3). For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.177 DRAM Control Register 188 (HW_DRAM_CTL188)

This is a DRAM configuration register.

Address: HW_DRAM_CTL188 – 800E_0000h base + 2F0h offset = 800E_02F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	MR3_DATA_3														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	MR3_DATA_2														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL188 field descriptions

Field	Description
31 RSVD2	Always write zeroes to this field.
30–16 MR3_DATA_3	Data to program into memory mode register 3 for chip select 3. Holds the memory mode register 3 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3). For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.
15 RSVD1	Always write zeroes to this field.
14–0 MR3_DATA_2	Data to program into memory mode register 3 for chip select 2. Holds the memory mode register 3 data for chip select X written during memory initialization. Consult the memory specification for the fields of this mode register. The use of this parameter varies based on the memory type connected to this EMI: For DDR1 memories: This parameter has no meaning for this memory type. For DDR2 memories: This parameter correlates to the extended memory mode register 3 (EMR3). For LPDDR1 memories: This parameter has no meaning for this memory type. This data will be programmed into the appropriate memory register of the DRAM at initialization or when the write_modereg parameter is set to 'b1.

14.8.178 DRAM Control Register 189 (HW_DRAM_CTL189)

This is a DRAM configuration register.

Address: HW_DRAM_CTL189 – 800E_0000h base + 2F4h offset = 800E_02F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AXI5_EN_SIZE_LT_WIDTH_INSTR																AXI4_EN_SIZE_LT_WIDTH_INSTR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DRAM_CTL189 field descriptions

Field	Description
31–16 AXI5_EN_SIZE_LT_WIDTH_INSTR	Allow narrow instructions from AXI port 5 requestors with bit enabled.
15–0 AXI4_EN_SIZE_LT_WIDTH_INSTR	Allow narrow instructions from AXI port 4 requestors with bit enabled.

Chapter 15
General-Purpose Media Interface(GPMI)

15.1 General-Purpose Media Interface Overview

This chapter describes the general-purpose media interface.

The GPMI controller is a flexible interface to up to eight NAND Flash with configurable address and command behavior, providing support for future devices not yet specified. The GPMI resides on the APBH and provides an interface to the 20-bit BCH module to allow direct parity processing.

Registers are clocked on the HCLK domain. The I/O and pin timing are clocked on a dedicated GPMICLK domain. GPMICLK can be set to maximize I/O performance.

The following figure shows a block diagram of the GPMI controller.

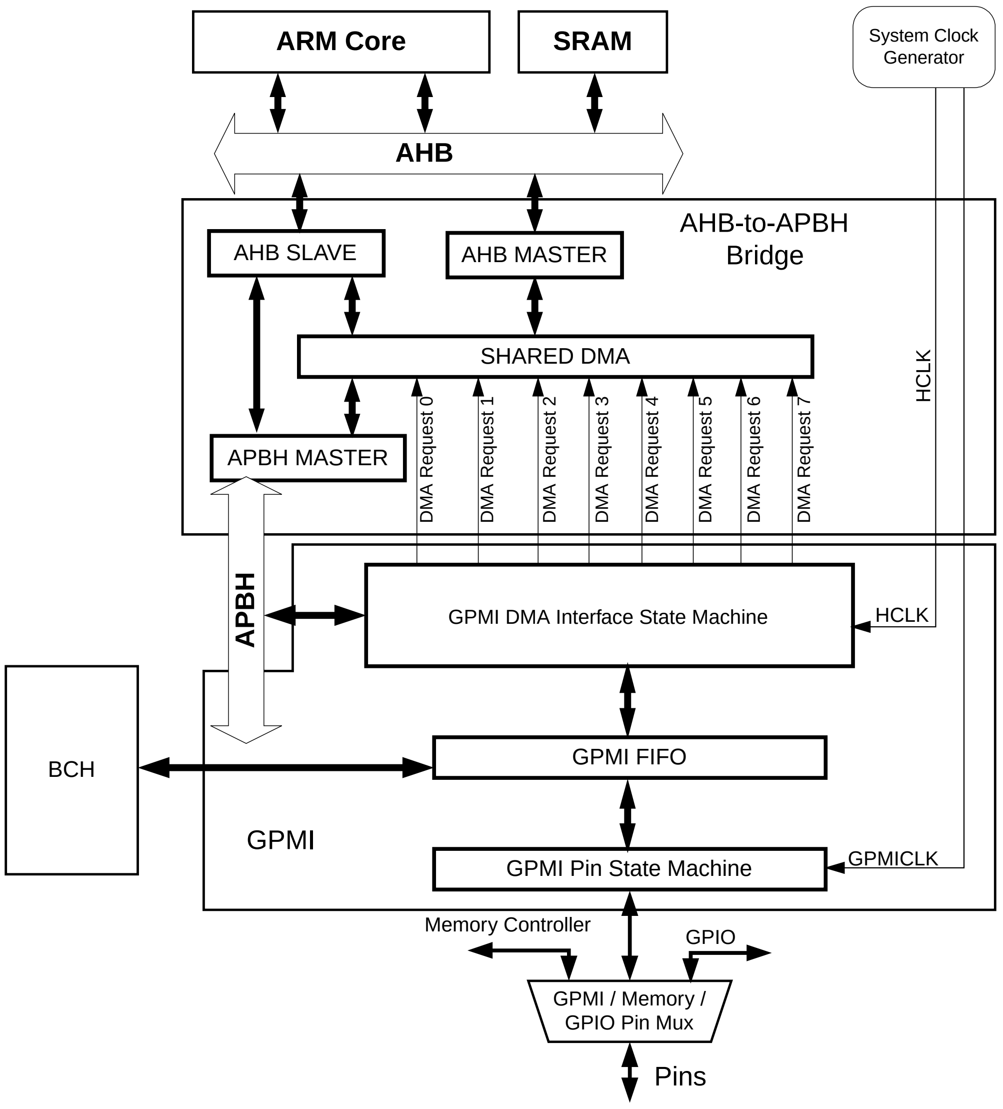


Figure 15-1. General-Purpose Media Interface Controller Block Diagram

15.2 GPMI NAND Mode

The general-purpose media interface has several features to efficiently support NAND:

- Individual chip select and ready/busy pins for up to eight NANDs.
- Individual state machine and DMA channel for each chip select.
- Special command modes work with DMA controller to perform all normal NAND functions without CPU intervention.
- Configurable timing based on a dedicated clock allows optimal balance of high NAND performance and low system power.

Since current NAND Flash does not support multiple page read/write commands, the GPMI and DMA have been designed to handle complex multi-page operations without CPU intervention. The DMA uses a linked descriptor function with branching capability to automatically handle all of the operations needed to read/write multiple pages:

- **Data/Register Read/Write**—The GPMI can be programmed to read or write multiple cycles to the NAND address, command or data registers.
- **Wait for NAND Ready**—The GPMI's Wait-for-Ready mode can monitor the ready/busy signal of a single NAND Flash and signal the DMA when the device has become ready. It also has a time-out counter and can indicate to the DMA that a time-out error has occurred. The DMAs can conditionally branch to a different descriptor in the case of an error.
- **Check Status**—The Read-and-Compare mode allows the GPMI to check NAND status against a reference. If an error is found, the GPMI can instruct the DMA to branch to an alternate descriptor, which attempts to fix the problem or asserts a CPU IRQ.

15.2.1 Multiple NAND Support

The GPMI supports up to eight NAND chip selects, each with independent ready/busy signals. Since they share a data bus and control lines, the GPMI can only actively communicate with a single NAND at a time. However, all NANDs can concurrently perform internal read, write, or erase operations. With fast NAND Flash and software support for concurrent NAND operations, this architecture allows the total throughput to approach the data bus speed, which can be as high as 50 MB/s (8-bit bus running at 50 MHz).

There are two options for controlling the eight NAND chip selects through the DMA interface. The first option is one to one mapping, where each DMA channel is attached to its own NAND. For example DMA channel 'n' accesses only NAND attached to chip select 'n'. The second option is the decoupled mode where a DMA channel can access any or all NANDs connected to the GPMI. A DMA channel will signify the NAND it wants to access by writing its chip select value in the HW_GPMI_CTRL0_CS field and setting the HW_GPMI_CTRL1_DECOUPLE_CS to '1'. This option is useful if software chooses to use only one DMA channel to access all the attached NAND devices.

15.2.2 GPMI NAND Timing and Clocking

The dedicated clock, GPMICLK, is used as a timing reference for NAND Flash I/O. Since various NANDs have different timing requirements, GPMICLK may need to be adjusted for each application. While the actual pin timings are limited by the NAND chips used, the

GPMI can support data bus speeds of up to 50 MHz x 8 bits. The actual read/write strobe timing parameters are adjusted as indicated in the GPMI register descriptions. See Clock Structure for more information about setting GPMICLK.

15.2.3 Basic NAND Timing

The following figure illustrates the operation of the timing parameters in NAND mode.

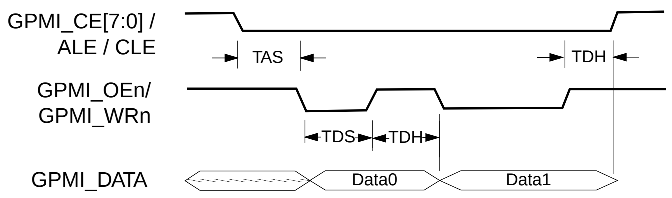


Figure 15-2. BASIC NAND Timing

15.2.4 High-Speed NAND Timing

In high-speed NANDs, the read data may not be valid until after the read strobe (RDN) deasserts. This is the case when the minimum tDS is programmed to achieve higher bandwidth. The GPMI implements a feedback read strobe to sample the read data. The feedback read strobe can be delayed to support fast NAND EDO (Extended Data Out) timing where the read strobe may deassert before the read data is valid, and read data is valid for some time after read strobe. NAND EDO timings is applied typically for read cycle frequency above 33 MHz.See Figure 15-3.

The GPMI provides control over the amount of delay applied to the feedback read strobe. This delay depends on the maximum read access time (tREA) of the NAND and the read pulse width (tRP) used to access the NAND. tRP is specified by HW_GPMI_TIMING0_DATA_SETUP register. When (tREA + 4 ns) is less than tRP, no delay is required to sample the NAND read data. (The 4 ns provides adequate data setup time for the GPMI.) In this case, set HW_GPMI_CTRL1_HALF_PERIOD=0; HW_GPMI_CTRL1_RDN_DELAY=0; HW_GPMI_CTRL1_DLL_ENABLE=0.

When (tREA + 4 ns) is greater than or equal to tRP, a delay of the feedback read strobe is required to sample the NAND read data. This delay is equal to the difference between these two timings:

DELAY= tREA+4 ns - tRP.

Since the GPMI delay chain is limited to 16 ns maximum, if DELAY > 16 ns then increase tRP by increasing the value of HW_GPMI_TIMING0_DATA_SETUP until DELAY is less than or equal to 16 ns.

The GPMI programming for this DELAY depends on the GPMICLK period. The GPMI DLL will not function properly if the GPMICLK period is greater than 32 ns: disable the DLL if this is the case. If the GPMICLK period is greater than 16 ns (and not greater than 32 ns), set the HW_GPMI_CTRL1_HALF_PERIOD=1; This will cause the DLL reference period (RP) to be one-half of the GPMICLK period. If the GPMICLK period is 16 ns or less then set the HW_GPMI_CTRL1_HALF_PERIOD=0; This will cause the DLL reference period (RP) to be equal to the GPMICLK period. DELAY is a multiple (0 to 1.875) of RP.

The HW_GPMI_CTRL1_RDN_DELAY is encoded as a 1-bit integer and 3-bit fraction delay factor. See table below. DELAY is a multiple of the delay factor and the reference period:

HW_GPMI_CTRL1_RDN_DELAY	0	1	2	3	4	5	6	7
Delay Factor	0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875
HW_GPMI_CTRL1_RDN_DELAY	8	9	10	11	12	13	14	15
Delay Factor	1.000	1.125	1.250	1.375	1.500	1.625	1.750	1.875

DELAY = DelayFactor x RP or DELAY = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x RP.

Use this equation to calculate the value for HW_GPMI_CTRL1_RDN_DELAY. Then set HW_GPMI_CTRL1_DLL_ENABLE=1.

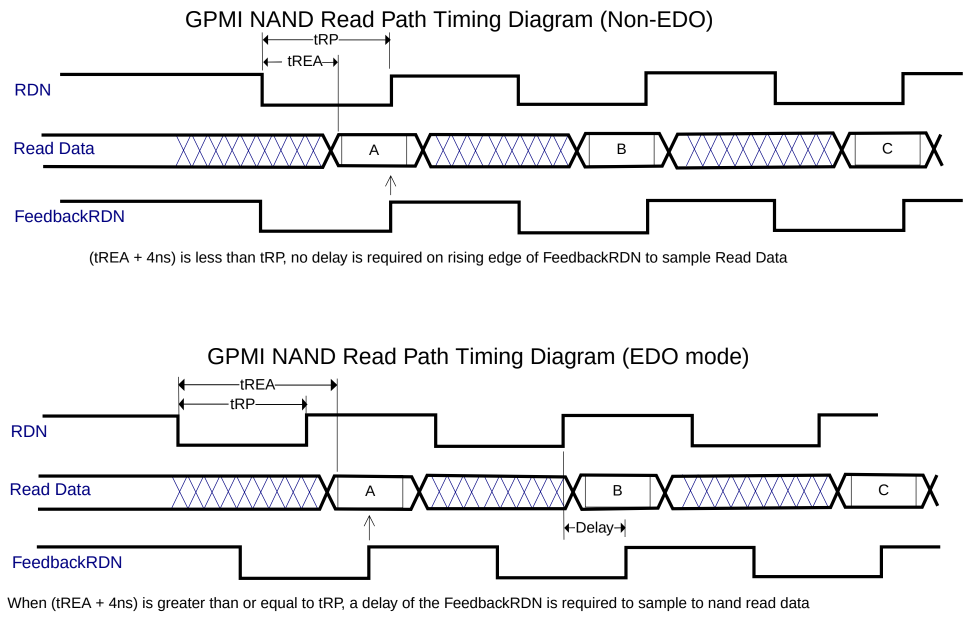

GPMI NAND Read Path Timing Diagram (Non-EDO)
tRP
tREA
RDN
Read Data
A
B
C
FeedbackRDN
(tREA + 4ns) is less than tRP, no delay is required on rising edge of FeedbackRDN to sample Read Data
GPMI NAND Read Path Timing Diagram (EDO mode)
tREA
tRP
RDN
Read Data
A
B
C
Delay
FeedbackRDN
When (tREA + 4ns) is greater than or equal to tRP, a delay of the FeedbackRDN is required to sample to nand read data

Figure 15-3. NAND Read Path Timing

For example, a NAND with tREAmax=20 ns, tRPmin=12 ns, and tRCmin=25 ns (read cycle time) may be programmed as follows:

- GPMICLK clock frequency: Consider 480/6=80 MHz which is 12.5 ns clock period. This is too close to the minimum NAND spec if we program the data setup and hold to 1 GPMICLK cycle. Consider 480/7=68.57 MHz which is 14.58 ns clock period. With data setup and hold set to 1, we have a tRP of 14.58 ns and a tRC of 29.16 ns (good margins).
- Since (tREA +4 ns) is greater than tRP, required DELAY = tREA+4 ns - tRP = 20 + 4 - 14.58 ns = 9.42 ns.
- HALF_PERIOD =0, since GPMICLK period is less than 16 ns. So RP=GPMICLK period = 14.58 ns.
- DELAY = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x RP.9.42 ns = HW_GPMI_CTRL1_RDN_DELAY x 0.125 x 14.58 ns.HW_GPMI_CTRL1_RDN_DELAY = 5 (round off 5.169)

Note

It is recommended that the drive strength of GPMI_RDn and GPMI_WRn output pins be set to 8 mA. This will reduce the transition time under heavy loads. Low transition times will be important when NAND interface read and write cycle times are below 30 ns. The other GPMI pins may remain at 4 mA, since their frequency is only up to half that of GPMI_RDn and GPMI_WRn.

15.2.5 NAND Command and Address Timing Example

The following figure illustrates a command and address being sent to a NAND Flash.

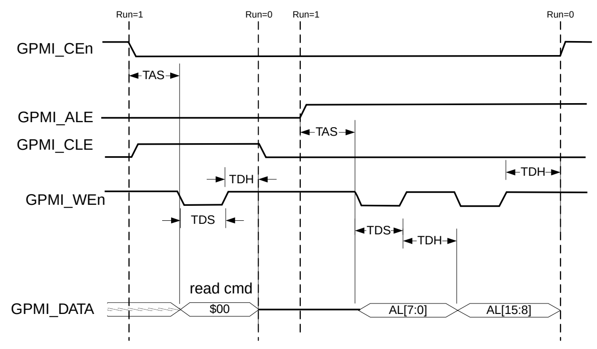


Figure 15-4. NAND Command and Address Timing Example

15.2.6 Hardware BCH Interface

The GPMI provides an interface to the BCH module. This reduces the SOC bus traffic and the software involvement. When in BCH mode, parity information is inserted on-the-fly during writes to 8-bit NAND devices. The BCH will supply payload and parity to the GPMI to write to the NAND. During NAND reads, parity is checked and ECC processing is performed after each read block. In this case, the GPMI reads the NAND device and redirects the data and parity to the BCH module for ECC processing.

To program the BCH for NAND writes, remove the soft reset and clock gates from HW_BCH_CTRL_SFTRST and HW_BCH_CTRL_CLKGATE. The bulk of BCH programming is actually applied to the GPMI through PIO operations embedded in its DMA command structures. This has a subtle implication when writing to the GPMI ECC registers: access to these registers must be written in a progressive register order. Therefore, to write to the HW_GPMI_ECCCOUNT register, write first (in order) to registers HW_GPMI_CTRL0, HW_GPMI_COMPARE, and HW_GPMI_ECCCTRL before writing to HW_GPMI_ECCCOUNT. These additional register writes need to be accounted for in the CMDWORDS field of the respective DMA channel command register. See the descriptive text, flowcharts, and code examples in Programming the BCH/GPMI Interfaces for more information about using GPMI registers to program the ECC function.

The HW_GPMI_PAYLOAD and HW_GPMI_AUXILIARY pointers need to be word-aligned for proper ECC operation. If those pointers are non-word-aligned, then the BCH engine will not operate properly and could possibly corrupt system memory in the adjoining memory regions.

15.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

15.4 Programmable Registers

GPMI Hardware Register Format Summary

HW_GPMI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_C000	GPMI Control Register 0 (HW_GPMI_CTRL0)	32	R/W	C000_0000h	15.4.1/1246
8000_C010	GPMI Compare Register (HW_GPMI_COMPARE)	32	R/W	0000_0000h	15.4.2/1248
8000_C020	GPMI Integrated ECC Control Register (HW_GPMI_ECCCTRL)	32	R/W	0000_0000h	15.4.3/1248
8000_C030	GPMI Integrated ECC Transfer Count Register (HW_GPMI_ECCCOUNT)	32	R/W	0000_0000h	15.4.4/1250
8000_C040	GPMI Payload Address Register (HW_GPMI_PAYLOAD)	32	R/W	0000_0000h	15.4.5/1250
8000_C050	GPMI Auxiliary Address Register (HW_GPMI_AUXILIARY)	32	R/W	0000_0000h	15.4.6/1251
8000_C060	GPMI Control Register 1 (HW_GPMI_CTRL1)	32	R/W	0004_0004h	15.4.7/1251
8000_C070	GPMI Timing Register 0 (HW_GPMI_TIMING0)	32	R/W	0001_0203h	15.4.8/1253
8000_C080	GPMI Timing Register 1 (HW_GPMI_TIMING1)	32	R/W	0000_0000h	15.4.9/1254
8000_C0A0	GPMI DMA Data Transfer Register (HW_GPMI_DATA)	32	R/W	0000_0000h	15.4.10/1254
8000_C0B0	GPMI Status Register (HW_GPMI_STAT)	32	R	0000_0005h	15.4.11/1255
8000_C0C0	GPMI Debug Information Register (HW_GPMI_DEBUG)	32	R	0000_0000h	15.4.12/1257
8000_C0D0	GPMI Version Register (HW_GPMI_VERSION)	32	R	0301_0000h	15.4.13/1258

15.4.1 GPMI Control Register 0 (HW_GPMI_CTRL0)

HW_GPMI_CTRL0: 0x000

HW_GPMI_CTRL0_SET: 0x004

HW_GPMI_CTRL0_CLR: 0x008

HW_GPMI_CTRL0_TOG: 0x00C

The GPMI control register 0 specifies the GPMI transaction to perform for the current command chain item.

Address: HW_GPMI_CTRL0 – 8000_C000h base + 0h offset = 8000_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RUN	RSVD1	LOCK_CS	RSVD0	COMMAND_MODE		WORD_LENGTH	CS			ADDRESS			ADDRESS_INCREMENT
W				▨		▨										
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XFER_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_CTRL0 field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. This will not work if the CLKGATE bit is already set to 1. CLKGATE must be cleared to 0 before issuing a soft reset. Also the GPMICLK must be running for this to work properly. 0x0 **RUN** — Allow GPMI to operate normally. 0x1 **RESET** — Hold GPMI in reset.
30 CLKGATE	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption. 0x0 **RUN** — Allow GPMI to operate normally. 0x1 **NO_CLKS** — Do not clock GPMI gates in order to minimize power consumption.
29 RUN	The GPMI is busy running a command whenever this bit is set to 1. The GPMI is idle whenever this bit set to zero. This can be set to one by a CPU write. In addition, the DMA sets this bit each time a DMA command has finished its PIO transfer phase. 0x0 **IDLE** — The GPMI is idle. 0x1 **BUSY** — The GPMI is busy running a command.
28 RSVD1	Always write zeros to this field.
27 LOCK_CS	For NAND mode: 0= Deassert chip select (CS) after RUN is complete. 1= Continue to assert chip select (CS) after RUN is complete. 0x0 **DISABLED** — Deassert chip select (CS) after RUN is complete. 0x1 **ENABLED** — Continue to assert chip select (CS) after RUN is complete.
26 RSVD0	Always write zeros to this field.
25–24 COMMAND_ MODE	00= Write mode. 01= Read Mode. 10= Read and Compare Mode (setting sense flop). 11= Wait for Ready. 0x0 **WRITE** — Write mode. 0x1 **READ** — Read mode. 0x2 **READ_AND_COMPARE** — Read and Compare mode (setting sense flop). 0x3 **WAIT_FOR_READY** — Wait for Ready mode.
23 WORD_LENGTH	0= not support. 1= 8-bit Data Bus mode. This bit should only 1.

HW_GPMI_CTRL0 field descriptions (continued)

Field	Description
22–20 CS	Selects which chip select is active for this command.
19–17 ADDRESS	In NAND mode, use A0 for CLE and A1 for ALE. 0x0 **NAND_DATA** — In NAND mode, this address is used to read and write data bytes. 0x1 **NAND_CLE** — In NAND mode, this address is used to write command bytes. 0x2 **NAND_ALE** — In NAND mode, this address is used to write address bytes.
16 ADDRESS_ INCREMENT	0= Address does not increment. 1= Increment address. In NAND mode, the address will increment once, after the first cycle (going from CLE to ALE). 0x0 **DISABLED** — Address does not increment. 0x1 **ENABLED** — Increment address.
15–0 XFER_COUNT	Number of words (8 bit wide) to transfer for this command. A value of zero will transfer 64K words.

15.4.2 GPMI Compare Register (HW_GPMI_COMPARE)

The GPMI compare register specifies the expect data and the xor mask for comparing to the status values read from the device. This register is used by the Read and Compare command.

Address: HW_GPMI_COMPARE – 8000_C000h base + 10h offset = 8000_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK																REFERENCE															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_COMPARE field descriptions

Field	Description
31–16 MASK	16-bit mask which is applied after the read data is XORed with the REFERENCE bit field.
15–0 REFERENCE	16-bit value which is XORed with data read from the NAND device.

15.4.3 GPMI Integrated ECC Control Register (HW_GPMI_ECCCTRL)

HW_GPMI_ECCCTRL: 0x020

HW_GPMI_ECCCTRL_SET: 0x024

HW_GPMI_ECCCTRL_CLR: 0x028

HW_GPMI_ECCCTRL_TOG: 0x02C

The GPMI ECC control register handles configuration of the integrated ECC accelerator.

Address: HW_GPMI_ECCCTRL – 8000_C000h base + 20h offset = 8000_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HANDLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	ECC_CMD		ENABLE_ECC	RSVD0			BUFFER_MASK								
W	▨				▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_ECCCTRL field descriptions

Field	Description
31–16 HANDLE	This is a register available to software to attach an identifier to a transaction in progress. This handle will be available from the ECC register space when the completion interrupt occurs.
15 RSVD1	Always write zeroes to this bit field.
14–13 ECC_CMD	ECC Command information. 0x0 **DECODE** — Decode. 0x1 **ENCODE** — Encode. 0x2 **RESERVE2** — Reserved. 0x3 **RESERVE3** — Reserved.
12 ENABLE_ECC	Enable ECC processing of GPMI transfers. 0x1 **ENABLE** — Use integrated ECC for read and write transfers. 0x0 **DISABLE** — Integrated ECC remains in idle.
11–9 RSVD0	Always write zeroes to this bit field.
8–0 BUFFER_MASK	ECC buffer information. The BCH error correction only allows two configurations of the buffer mask - software may either read just the first block on the flash page or the entire flash page. Write operations must be for the entire flash page. Invalid buffer mask values will cause the DMA descriptor command to be terminated. 0x100 **BCH_AUXONLY** — Set to request transfer from only the auxiliary buffer (block 0 on flash). 0x1FF **BCH_PAGE** — Set to request transfer to/from the entire page.

15.4.4 GPMI Integrated ECC Transfer Count Register (HW_GPMI_ECCCOUNT)

The GPMI ECC Transfer Count Register contains the count of bytes that flow through the ECC subsystem.

Address: HW_GPMI_ECCCOUNT – 8000_C000h base + 30h offset = 8000_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																COUNT															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_ECCCOUNT field descriptions

Field	Description
31–16 RSVD0	Always write zeroes to this bit field.
15–0 COUNT	Number of bytes to pass through ECC. This is the GPMI transfer count plus the syndrome count that will be inserted into the stream by the ECC. In DMA2ECC_MODE this count must match the HW_GPMI_CTRL0_XFER_COUNT. A value of zero will transfer 64K words.

15.4.5 GPMI Payload Address Register (HW_GPMI_PAYLOAD)

The GPMI payload address register specifies the location of the data buffers in system memory. This value must be word aligned.

Address: HW_GPMI_PAYLOAD – 8000_C000h base + 40h offset = 8000_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																														RSVD0	
W																															▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_PAYLOAD field descriptions

Field	Description
31–2 ADDRESS	Pointer to an array of one or more 512 byte payload buffers.
1–0 RSVD0	Always write zeroes to this bit field.

15.4.6 GPMI Auxiliary Address Register (HW_GPMI_AUXILIARY)

The GPMI auxiliary address register specifies the location of the auxiliary buffers in system memory. This value must be word aligned.

Address: HW_GPMI_AUXILIARY – 8000_C000h base + 50h offset = 8000_C050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADDRESS																														RSVD0	
W																															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_AUXILIARY field descriptions

Field	Description
31–2 ADDRESS	Pointer to ECC control structure and meta-data storage.
1–0 RSVD0	Always write zeroes to this bit field.

15.4.7 GPMI Control Register 1 (HW_GPMI_CTRL1)

HW_GPMI_CTRL1: 0x060

HW_GPMI_CTRL1_SET: 0x064

HW_GPMI_CTRL1_CLR: 0x068

HW_GPMI_CTRL1_TOG: 0x06C

The GPMI control register 1 specifies additional control fields that are not used on a per-transaction basis.

Address: HW_GPMI_CTRL1 – 8000_C000h base + 60h offset = 8000_C060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2							DECOUPLE_CS	WRN_DLY_SEL		RSVD1	TIMEOUT_IRQ_EN	GANGED_RDYBUSY	BCH_MODE	DLL_ENABLE	HALF_PERIOD
W	▨	▨	▨	▨	▨	▨	▨				▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDN_DELAY				DMA2ECC_ MODE	RSVD	TIMEOUT_IRQ	BURST_EN	ABORT_ WAIT_ REQUEST	ABORT_WAIT_FOR_ READY_CHANNEL			DEV_RESET	ATA_IRQRDY_ POLARITY	RSVD0	GPMI_MODE
W						▨									▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_GPMI_CTRL1 field descriptions

Field	Description
31–25 RSVD2	Always write zeroes to this bit field.
24 DECOUPLE_CS	Decouple Chip Select from DMA Channel. Setting this bit to 1 will allow a DMA channel to specify any value in the CTRL0_CS register field. Software can use one DMA channel to access all 8 Nand devices.
23–22 WRN_DLY_SEL	Since the GPMI write strobe (WRN) is a fast clock pin, the delay on this signal can be programmed to match the load on this pin. 0 = 4ns to 8ns; 1 = 6ns to 10ns; 2 = 7ns to 12ns; 3 = no delay.
21 RSVD1	Always write zeroes to this bit field.
20 TIMEOUT_IRQ_ EN	Setting this bit to 1 will enable timeout IRQ for WAIT_FOR_READY commands in Nand mode. The Device_Busy_Timeout value is used for this timeout.
19 GANGED_ RDYBUSY	Set this bit to 1 will force all Nand RDY_BUSY inputs to be sourced from (tied to) RDY_BUSY0. This will free up all, except one, RDY_BUSY input pins.
18 BCH_MODE	This bit selects which error correction unit will access GPMI. This bit must always be set to 1, since only the BCH unit is available in this design.
17 DLL_ENABLE	Set this bit to 1 to enable the GPMI DLL. This is required for fast NAND reads (above 30MHz read strobe). After setting this bit, wait 64 GPMI clock cycles for the DLL to lock before performing a NAND read.
16 HALF_PERIOD	Set this bit to 1 if the GPMI clock period is greater than 16ns for proper DLL operation. DLL_ENABLE must be zero while changing this field.
15–12 RDN_DELAY	This variable is a factor in the calculated delay to apply to the internal read strobe for correct read data sampling. The applied delay (AD) is between 0 and 1.875 times the reference period (RP). RP is one half of the GPMI clock period if HALF_PERIOD=1 otherwise it is the full GPMI clock period. The equation is: AD = RDN_DELAY x 0.125 x RP. This value must not exceed 16ns. This variable is used to achieve faster NAND access. For example if the Read Strobe is asserted from time 0 to 13ns but the read access time is 20ns, then choose AD=12ns will cause the data to be sampled at time 25ns (13+12) giving a 5ns data setup time. If RP=13ns then RDN_DELAY = 12/(0.125 x 13ns) = 7.38 (0111b). DLL_ENABLE must be zero while changing this field.

HW_GPMI_CTRL1 field descriptions (continued)

Field	Description
11 DMA2ECC_ MODE	This is mainly for testing HWECC without involving the Nand device. Setting this bit will cause DMA write data to redirected to HWECC module (instead of Nand Device) for encoding or decoding.
10 RSVD	Always write zeros to this bit field.
9 TIMEOUT_IRQ	This bit is set when a timeout occurs using the Device_Busy_Timeout value. Write 0 to clear.
8 BURST_EN	When set to 1 each DMA request will generate a 4-transfer burst on the APB bus.
7 ABORT_WAIT_ REQUEST	Request to abort the wait for ready command on the channel indicated by ABORT_WAIT_FOR_READY_CHANNEL. Hardware clears this bit when the abort is done.
6–4 ABORT_WAIT_ FOR_READY_ CHANNEL	Abort a wait for ready command on selected channel. Set the ABORT_WAIT_REQUEST to kick of operation.
3 DEV_RESET	Nand Flash write protection control. 0x0 **ENABLED** — enable write protection for all Nand devices. 0x1 **DISABLED** — disable write protection for all Nand devices.
2 ATA_IRQRDY_ POLARITY	Always set this bit to 1.
1 RSVD0	Always write zeros to this bit field.
0 GPMI_MODE	0= NAND mode. This bit should be 0 only.

15.4.8 GPMI Timing Register 0 (HW_GPMI_TIMING0)

The GPMI timing register 0 specifies the timing parameters that are used by the cycle state machine to guarantee the various setup, hold and cycle times for the external media type.

Address: HW_GPMI_TIMING0 – 8000_C000h base + 70h offset = 8000_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								ADDRESS_SETUP								DATA_HOLD								DATA_SETUP							
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1

HW_GPMI_TIMING0 field descriptions

Field	Description
31–24 RSVD0	Always write zeroes to this bit field.
23–16 ADDRESS_ SETUP	Number of GPMICLK cycles that the CE/ADDR signals are active before a strobe is asserted. A value of zero is interpreted as 0.
15–8 DATA_HOLD	Data bus hold time in GPMICLK cycles. Also the time that the data strobe is de-asserted in a cycle. A value of zero is interpreted as 256.
7–0 DATA_SETUP	Data bus setup time in GPMICLK cycles. Also the time that the data strobe is asserted in a cycle. A value of zero is interpreted as 256.

15.4.9 GPMI Timing Register 1 (HW_GPMI_TIMING1)

The GPMI timing register 1 specifies the timeouts used when monitoring the NAND READY pin and IOWAIT signals.

Address: HW_GPMI_TIMING1 – 8000_C000h base + 80h offset = 8000_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEVICE_BUSY_TIMEOUT																RSVD0															
W																	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_TIMING1 field descriptions

Field	Description
31–16 DEVICE_BUSY_ TIMEOUT	Timeout waiting for NAND Ready/Busy. Used in WAIT_FOR_READY mode. This value is the number of GPMI_CLK cycles multiplied by 4096.
15–0 RSVD0	Always write zeroes to this bit field.

15.4.10 GPMI DMA Data Transfer Register (HW_GPMI_DATA)

The GPMI DMA data transfer register is used by the DMA to read or write data to or from the NAND control state machine.

Address: HW_GPMI_DATA – 8000_C000h base + A0h offset = 8000_C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_DATA field descriptions

Field	Description
31–0 DATA	In 8-bit mode, one, two, three or four bytes can can be accessed to send the same number of bus cycles.

15.4.11 GPMI Status Register (HW_GPMI_STAT)

The GPMI control and status register provides a read back path for various operational states of the GPMI controller.

Address: HW_GPMI_STAT – 8000_C000h base + B0h offset = 8000_C0B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	READY_BUSY								RDY_TIMEOUT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DEV7_ ERROR	DEV6_ ERROR	DEV5_ ERROR	DEV4_ ERROR	DEV3_ ERROR	DEV2_ ERROR	DEV1_ ERROR	DEV0_ ERROR	RSVD0			GPMI_ RDY1	INVALID_ BUFFER_ MASK	FIFO_ EMPTY	FIFO_ FULL	PRESENT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

HW_GPMI_STAT field descriptions

Field	Description
31–24 READY_BUSY	Read-only view of NAND Ready_Busy Input pins.
23–16 RDY_TIMEOUT	State of the RDY/BUSY Timeout Flags. When any bit is set to '1' in this field, it indicates that a time out has occurred while waiting for the ready state of the requested NAND device. Multiple bits may be set simultaneously.

HW_GPMI_STAT field descriptions (continued)

Field	Description
	When HW_GPMI_CTRL1_DECOUPLE_CS = 0, RDY_TIMEOUT[n] is associated with the NAND device on chip_select[n]. When HW_GPMI_CTRL1_DECOUPLE_CS = 1, these flags become associated to a DMA channel instead of a NAND device. For example if DMA channel 6 sends a WAIT_FOR_READY command for NAND Device 2, and a timeout occured on READY_BUSY2, then READY_TIMEOUT[6] will be set instead of READY_TIMEOUT[2].
15 DEV7_ERROR	0= No error condition present on NAND Device accessed by DMA channel 7. 1= An Error has occurred on NAND Device accessed by DMA channel 7 (Timeout or compare failure, depending on COMMAND_MODE).
14 DEV6_ERROR	0= No error condition present on NAND Device accessed by DMA channel 6. 1= An Error has occurred on NAND Device accessed by DMA channel 6 (Timeout or compare failure, depending on COMMAND_MODE).
13 DEV5_ERROR	0= No error condition present on NAND Device accessed by DMA channel 5. 1= An Error has occurred on NAND Device accessed by DMA channel 5 (Timeout or compare failure, depending on COMMAND_MODE).
12 DEV4_ERROR	0= No error condition present on NAND Device accessed by DMA channel 4. 1= An Error has occurred on NAND Device accessed by DMA channel 4 (Timeout or compare failure, depending on COMMAND_MODE).
11 DEV3_ERROR	0= No error condition present on NAND Device accessed by DMA channel 3. 1= An Error has occurred on NAND Device accessed by DMA channel 3 (Timeout or compare failure, depending on COMMAND_MODE).
10 DEV2_ERROR	0= No error condition present on NAND Device accessed by DMA channel 2. 1= An Error has occurred on NAND Device accessed by DMA channel 2 (Timeout or compare failure, depending on COMMAND_MODE).
9 DEV1_ERROR	0= No error condition present on NAND Device accessed by DMA channel 1. 1= An Error has occurred onNAND Device accessed by DMA channel 1 (Timeout or compare failure, depending on COMMAND_MODE).
8 DEV0_ERROR	0= No error condition present on NAND Device accessed by DMA channel 0. 1= An Error has occurred on NAND Device accessed by DMA channel 0 (Timeout or compare failure, depending on COMMAND_MODE).
7–5 RSVD0	Always write zeroes to this bit field.
4 GPMI_RDY1	Status of theGPMI_RDY1 input pin.
3 INVALID_ BUFFER_MASK	0= ECC Buffer Mask is not invalid. 1= ECC Buffer Mask is invalid.

HW_GPMI_STAT field descriptions (continued)

Field	Description
2 FIFO_EMPTY	0= FIFO is not empty. 1= FIFO is empty. 0x0 **NOT_EMPTY** — FIFO is not empty. 0x1 **EMPTY** — FIFO is empty.
1 FIFO_FULL	0= FIFO is not full. 1= FIFO is full. 0x0 **NOT_FULL** — FIFO is not full. 0x1 **FULL** — FIFO is full.
0 PRESENT	0= GPMI is not present in this product. 1= GPMI is present is in this product. 0x0 **UNAVAILABLE** — GPMI is not present in this product. 0x1 **AVAILABLE** — GPMI is present in this product.

15.4.12 GPMI Debug Information Register (HW_GPMI_DEBUG)

The GPMI debug information register provides a read back path for diagnostics to determine the current operating state of the GPMI controller.

Address: HW_GPMI_DEBUG – 8000_C000h base + C0h offset = 8000_C0C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WAIT_FOR_READY_END								DMA_SENSE								DMAREQ								CMD_END							
W	▨								▨								▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_DEBUG field descriptions

Field	Description
31–24 WAIT_FOR_ READY_END	Read Only view of the Wait_For_Ready End toggle signals to DMA. One per channel
23–16 DMA_SENSE	Read-only view of sense state of the 8 DMA channels. A value of 1 in any bit position indicates that a read and compare command failed or a timeout occured for the corresponding channel.
15–8 DMAREQ	Read-only view of DMA request line for 8 DMA channels. A toggle on any bit position indicates a DMA request for the corresponding channel.
7–0 CMD_END	Read Only view of the Command End toggle signals to DMA. One per channel

15.4.13 GPMI Version Register (HW_GPMI_VERSION)

This register reflects the version number for the GPMI.

Address: HW_GPMI_VERSION – 8000_C000h base + D0h offset = 8000_C0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_GPMI_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 16
20-BIT Correcting ECC Accelerator (BCH)

16.1 BCH Overview

The hardware ECC accelerator provides a forward error-correction function for improving the reliability of various storage media that may be attached to the device. For example, modern high-density NAND flash devices presume the existence of forward error-correction algorithms to correct some soft and/or hard bit errors within the device, allowing for higher device yields and, therefore, lower NAND device costs.

The Bose, Ray-Chaudhuri, Hocquenghem (BCH) Encoder and Decoder module is capable of correcting from 2 to 20 single bit errors within a block of data no larger than about 900 bytes (512 bytes is typical) in applications such as protecting data and resources stored on modern NAND flash devices. The correction level in the BCH block is programmable to provide flexibility for varying applications and configurations of flash page size. The design can be programmed to encode protection of 2, 4, 6, 8, 10, 12, 14, 16, 18, or 20 bit errors when writing flash and to correct the corresponding number of errors on decode. The correction level when decoding MUST be programmed to the same correction level as was used during the encode phase.

BCH-codes are a type of block-code, which implies that all error-correction is performed over a block of N-symbols. The BCH operation will be performed over GF(2^{13} = 8192), which is the Galois Field consisting of 8191 one-bit symbols. BCH-encoding (or encode for any block-code) can be performed by two algorithms: systematic encoding or multiplicative encoding. Systematic encoding is the process of reading all the symbols which constitute a block, dividing continuously these symbols by the generator polynomial for the GF(8192) and appending the resulting ***t*** parity symbols to the block to create a BCH codeword (where ***t*** is the number of correctable bits).

The BCH encode process creates *t* 13-bit parity symbols for each data block when the data is written to the flash device. The parity symbols are written to the flash device after the corresponding data block, and together these are collectively called the codeword. The codeword can be used during the decode process to correct errors that occur in either the data or parity blocks.

The BCH decoder processes code words in a 4-step fashion:

1. Syndrome Calculation (SC): This is the process of reading in all of the symbols of the codeword and continuously dividing by the generator polynomial for the field. 2**t* syndromes must be calculated for each codeword and inspection of the syndromes determines if there are errors: a non-zero set of syndromes indicates one or more errors. This process is implemented parallel hardware to minimize processing time since it must be done every time the decode is performed.
2. Key Equation Solver (KES): The syndromes represent 2t-linear equations with 2t-unknown variables. The process of solving these equations and selecting from the numerous solutions constitutes the KES module. When the KES block completes its operations, it generates an error locator polynomial (sigma) that is used in the proceeding block to determine the locations and values of the errors.
3. Chien Search (CS): This block takes input from the KES block and uses the Chien Algorithm for finding the locations of the errors based on the error locator polynomial. The method basically involves substituting all 8191 symbols from the GF(8192) into the locator polynomial. All evaluations that produce a zero solution indicate locations of the various errors. Since each located error corresponds to a single bit, the bit in the original data may be corrected by simply flipping the polarity of the incorrect location.
4. Correction: this block has to convert the symbol index and mask information to memory byte indexes and masks.

The BCH block, shown in the figure, was designed to operate in a pipelined fashion to maximize throughput. Aside from the initial latency to fill the pipeline stages, the BCH throughput is faster than the fastest GPMI read rate of 2 cycles/byte. Thus, the bottleneck in performing NAND reads and error corrections is the GPMI read rate. Current GPMI read rates are approximately 3 cycles/byte for the current generation of NAND flash. The CPU is not directly involved in generating parity symbols, checking for errors, or correcting them.

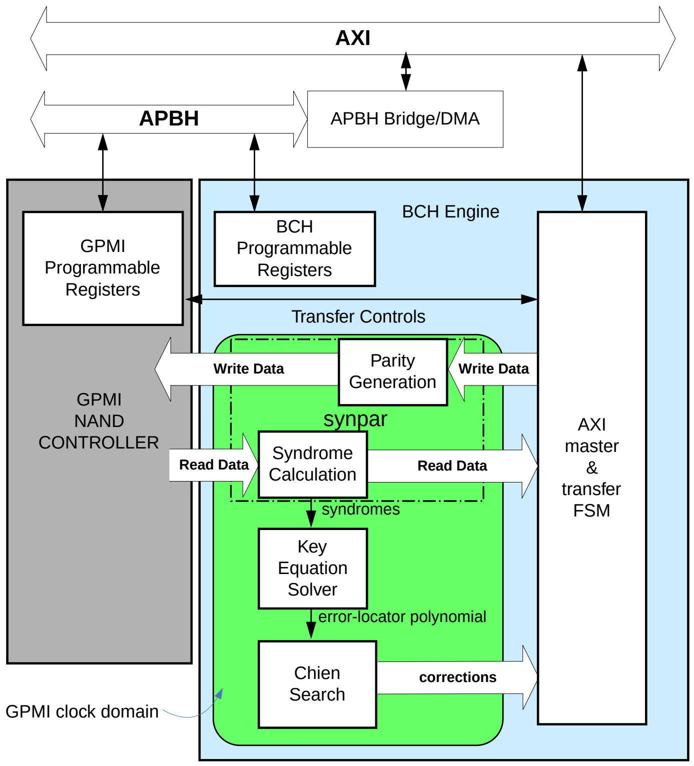


Figure 16-1. Hardware BCH Accelerator

16.2 Operation

Before performing any NAND flash read or write operations, software should first program the BCH's flash layout registers (see Flash Page Layout) to specify how data is to be formatted on the flash device. The BCH hardware allows full programmability over the flash page layout to enable users flexibility in balancing ECC correction levels and ever-changing flash page sizes.

To initiate a NAND Flash write, software will program a GPMI DMA operation. The DMA need only program the GPMI control registers (and handle the requisite flash addressing handshakes) since the BCH will handle all data operations using its AXI bus interface. The BCH will then send the data to the GPMI controller to be written to flash as it computes the parity symbols. At the end of each data block the BCH will insert the parity symbols into the data stream so that the GPMI sees only a continuous stream of data to be written.

NAND Flash read operations operate in a similar manner. As the GPMI controller reads the device, all data is sent to the BCH hardware for error detection/correction. The BCH controller writes all incoming read data to system memory and in parallel computes the syndromes used to detect bit errors. If errors are detected within a block, the BCH hardware activates the error correction logic to determine where bit errors have occurred and ultimately correct them in the data buffer in system memory. After an entire flash page has been read and corrected, the BCH will signal an interrupt to the CPU.

Figure 16-2 indicates how data read from the GPMI is operated on within the BCH hardware. As the BCH receives data from the GPMI (top row), it is written to memory by the BCH's Bus Interface Unit (BIU) (second row). For blocks requiring correction, the KES logic will be activated after the entire block has been received. Once the error locator polynomial has been computed, the corrections are determined by the Chien Search and fed back to the BIU, which performs a read, modify, write operation on the buffer in memory to correct the data.

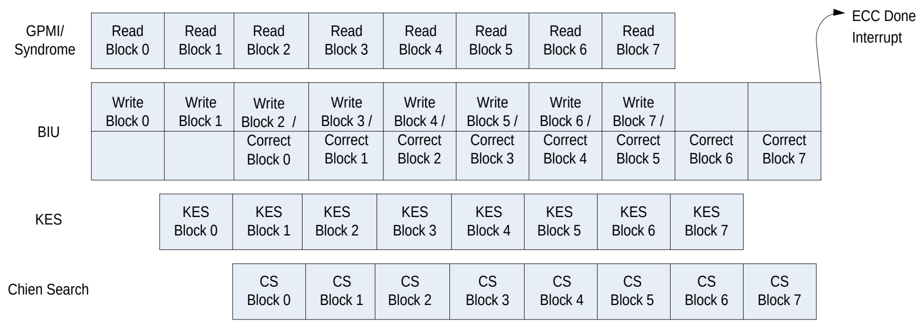


Figure 16-2. Block Pipeline while Reading Flash

16.2.1 BCH Limitations and Assumptions

- The BCH is programmable to support 2, 4, 6, 8, 10, 12, 14, 16, 18, and 20 bit error correction. ECC0 is supported as a passthrough, non-correcting mode.
- Data block sizes must be a multiple of 4 bytes and be aligned in system memory.
- The BCH supports a programmable number of metadata/auxiliary data bytes, from 0 to 255.
- Metadata will be written at the beginning of the flash page to facilitate fast access for filesystem operations.
- Metadata may be treated as an independent block for ECC purposes or combined with the first data block to conserve bits in the flash.
- The BCH does not support a partial page write (this can be accomplished by programming the BCH layout registers such that the BCH only sees a portion of the page).

- Flash read operations can read the entire page or the first block on the page.
- The BCH also supports a memory-to-memory mode of operation that does not require the use of DMA or the GPMI.

16.2.2 Flash Page Layout

The BCH supports a fully programmable flash page layout. The BCH maintains four independent layout registers that can describe four completely different NAND devices or layouts. When the BCH initiates an operation, it selects one of the layouts by using the chip select as an index into the HW_BCH_LAYOUTSELECT register the determines which layout should be used for the operation.

Three possible (generic) flash layout schemes are supported, as indicated in Figure 16-3. (In each case, the metadata size may also be programmed to 0 bytes). Metadata may either be combined with the first block of data or the size of the first data block can be programmed to 0 to allow the metadata to be protected by its own ECC parity bits.

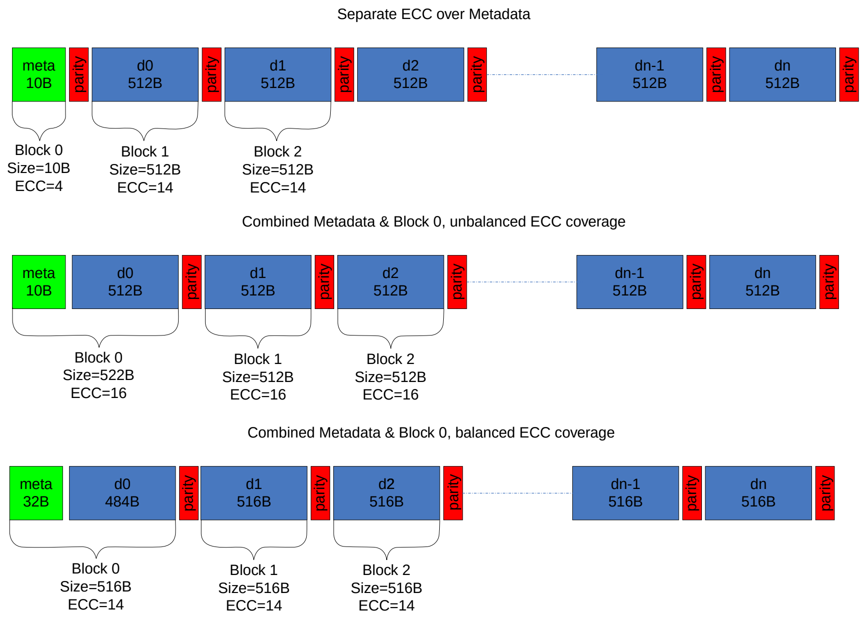


Figure 16-3. FLASH Page Layout Options

Each layout is determined by a pair of registers that define the following parameters:

- DATA0_SIZE: Indicates the number of data bytes in the first block on the page (this should not include parity or metadata bytes). This should be set to 0 when the metadata is to be covered separately with its own ECC. This must be a multiple of 4 bytes.

- ECC0: Indicates the ECC level to be used for the first block on the flash (data0+metadata).
- META_SIZE: Indicates the number of bytes (from 0-255) that are stored as metadata.
- NBLOCKS: Indicates the number of subsequent DATAN blocks on the flash, or the number of blocks following the DATA0 block.
- DATAN_SIZE: Indicates the number of data bytes in all subsequent data blocks. This MUST be a multiple of 4 bytes.
- ECCN: Indicates the ECC level to be used for the subsequent data blocks.
- PAGE_SIZE: Indicates the total number of bytes available per page on the physical flash device. This includes the spare area and is typically 4096+128, 4096+218, or 2048+64 bytes.

16.2.3 Determining the ECC layout for a device

Since the BCH is programmable, a system can trade off ECC levels for flash size and layout configurations. The following examples indicate how to determine a valid layout based on the required storage space and flash size. For all cases, the size of the parity will be 13*ECC level ***bits***-- so for ECC8, 13 bytes are required (per block).

16.2.3.1 4K+218 flash, 10 bytes metadata, 512 byte data blocks, separate metadata

In this case, we have 8 data blocks each consisting of 512 bytes. Since the flash has 218 spare bytes (1744 bits), first estimate an ECC level for the data blocks by first subtracting the number of metadata bytes from the spare bytes (218 – 10 = 208 bytes = 1664 bits) then dividing the number of bits by 8 (number of blocks) and then by 13 (bits per ECC level).

$$(218 - 10) \times 8 = \frac{1664}{13(8)} = 16$$

Therefore all the data blocks could be covered by ECC16 if the metadata had no parity. This isn't acceptable, so assume ECC14 for all the data blocks. Now calculate the number of free bits for the metadata parity as

$1664 - (14) \times 13 \times 8 = 208$

Therefore, 208 bits remain for metadata parity. Dividing by 13 (bits/ECC) gives 16, so the metadata can be covered with ECC16. The settings for this device would then be

Table 16-1. Settings for 4K+218 FLASH

Setting	Value
PAGE_SIZE	4096+218=4314=0x10DA
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	16=0x10
DATAN_SIZE	512=0x200
ECCN	14=0x0E
NBLOCKS	8

16.2.3.2 4K+128 flash, 10 bytes metadata, 512 byte data blocks, separate metadata

This flash will have 118 bytes available for ECC (after subtracting the metadata size), therefore, 994 bits. Dividing by 8*13 (number of blocks * ECC level) we get 9.07, therefore we can support ECC8 on the data blocks. The number of free spare bits becomes 944-8*8*13=944-832=112, divided by 13 = 8.6, therefore the metadata can be also covered by ECC8.

Table 16-2. Settings for 4K+128 FLASH

Setting	Value
PAGE_SIZE	4096+128=4224=0x1080
META_SIZE	10=0x0A
DATA0_SIZE	0
ECC0	8
DATAN_SIZE	512=0x200
ECCN	8
NBLOCKS	8

In this case, there will be additional unused spare bits, with the BCH will pad out with zeros.

16.2.4 Data Buffers in System Memory

While the data on the flash is interleaved with parity symbols, the BCH assumes that the data buffers in memory are contiguous. Metadata read from the flash will be stored to the location pointed to by the HW_GPMI_AUXILIARY register and data will be written to the address specified in the HW_GPMI_PAYLOAD register as is shown in Figure 16-4. Since the number of blocks on a flash page is programmable, the BCH also writes individual block correction status to the auxiliary pointer at the word-aligned address following the end of the metadata. Optionally, the computed syndromes may also be written to the auxiliary area if the DEBUGSYNDROME bit is set in the control register.

As blocks complete processing, the bus master will accumulate the status for each block and write it to the auxiliary data buffer following the metadata. The metadata area will be padded with 0's until the next word boundary and the status for blocks 0-3 will be written to the next word. The status for subsequent blocks will then be written to the buffer. The status for the first block (metadata block) is also stored in the STATUS_BLK0 register in the BCH_STATUS register. The completion codes for the blocks are indicated in the Table 16-3. Note that the definition of the bytes and their ordering in the auxiliary and payload storage areas are user defined. When this data is read back from the flash and put into memory, it will resemble the original buffer that was written out to the flash.

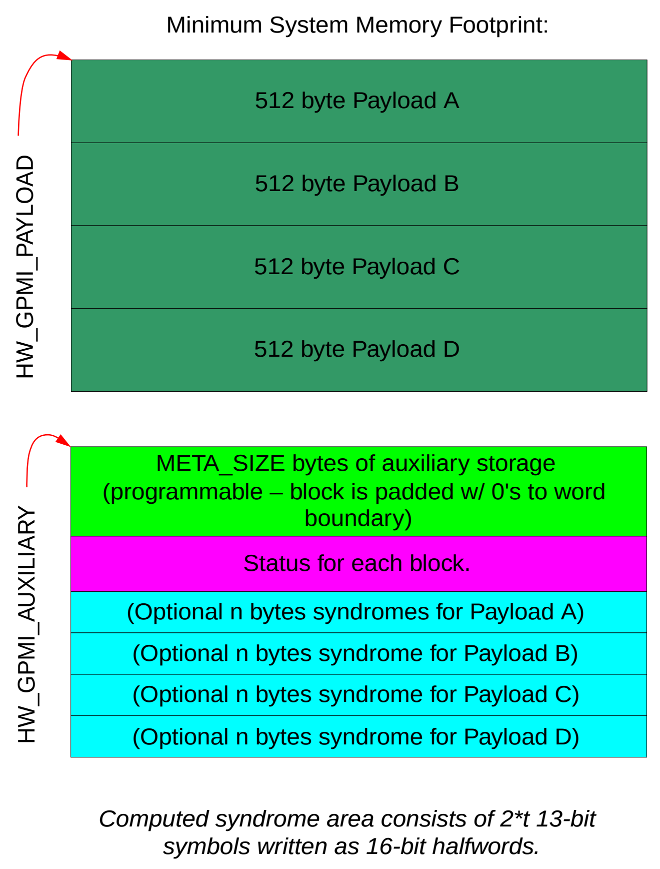
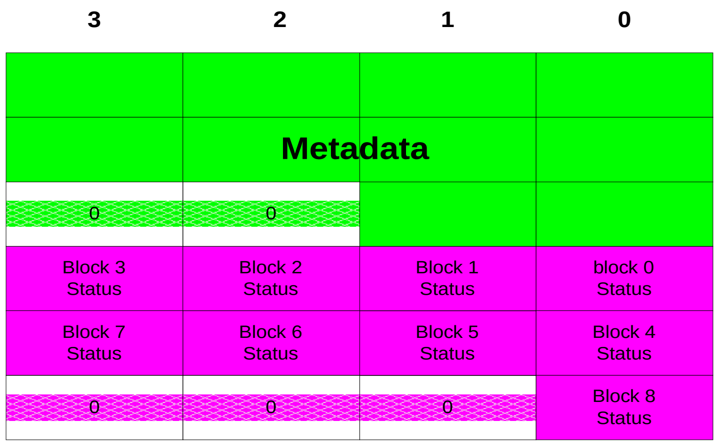

Minimum System Memory Footprint:
HW_GPMI_PAYLOAD
512 byte Payload A
512 byte Payload B
512 byte Payload C
512 byte Payload D
HW_GPMI_AUXILIARY
META_SIZE bytes of auxiliary storage (programmable – block is padded w/ 0's to word boundary)
Status for each block.
(Optional n bytes syndromes for Payload A)
(Optional n bytes syndrome for Payload B)
(Optional n bytes syndrome for Payload C)
(Optional n bytes syndrome for Payload D)
Computed syndrome area consists of 2*t 13-bit symbols written as 16-bit halfwords.

Figure 16-4. BCH Data Buffers in Memory

Table 16-3. Status Block Completion Codes

Code	Description
0xFF	Block is erased
0xFE	Block is uncorrectable
0x00	No errors found
0x01-0x14	Number of errors corrected

The following figure shows the layout of the bytes within the status field.



Status bytes are allocated based on the NBLOCKS programmed into the flash format register. The number of status bytes will be computed by the NBLOCKS+1. The status area will be padded with zeros to the next word boundary.

Syndrome data written for debug purposes will follow the end of the status block.

Figure 16-5. Memory-to-Memory Operations

16.3 Memory to Memory (Loopback) Operation

The BCH supports a memory-to-memory mode of operation where both the encoded and decoded buffers reside in system memory. This can be useful for applications where data must be protected by ECC, but the storage device does not reside on the GPMI bus.

The BCH operation in memory to memory mode is much simpler than in GPMI mode since DMAs are not required to manage the operation. Instead, software simply writes the HW_BCH_DATAPTR and HW_BCH_METAPTR with the addresses of the data and metadata (auxiliary) buffers and the HW_BCH_ENCODEPTR with the address of the buffer for encoded data. To initiate the operation, software simply sets the M2M_ENCODE and M2M_ENABLE bits in the control register. The BCH can be programmed to either issue an interrupt at the end of the operation or software may poll the status bits for completion.

Memory to memory decode operations work in a similar manner. The encoded data address is written to the HW_BCH_ENCODEPTR and the data and meta pointers are written to buffers that correspond to the desired decoded data addresses. To initiate a decode, software must set the M2M_ENCODE bit to 0 while writing the M2M_ENABLE bit. Note that the addresses written to the HW_BCH_DATAPTR, HW_BCH_METAPTR and HW_BCH_ENCODEPTR registers should always be aligned on a 4 byte boundary. In other words, the 2 lower bits of the address should always be written with zeros.

16.4 Programming the BCH/GPMI Interfaces

Programming the BCH for NAND operations consists largely of disabling the soft reset and clock bits (SFTRST and CLKGATE) from the HW_BCH_CTRL register and then programming the flash layout registers to correspond to the format of the attached NAND device(s). The HW_BCH_LAYOUTSELECT register should also be programmed to map the chip select of each attached device into one of the four layout registers.

The bulk of the programming is actually applied to the GPMI through PIO operations embedded in DMA command structures. The DMA will perform all the requisite handshaking with the GPMI interface to negotiate the address portion of the transfer, then the BCH will handle all the movement of data from memory to the GPMI (writes) or the GPMI to memory (reads). The BCH will direct all data blocks to the buffer pointed to by the PAYLOAD_BUFFER and the metadata will be written to the AUXILIARY_BUFFER. Both of these registers are located in the GPMI PIO data space and are communicated to the BCH hardware at the beginning of the transfer. Thus, the normal multi-NAND DMA based device interleaving is preserved, that is, four NANDs on four separate chip selects can be scheduled for read or write operations using the BCH. Whichever channel finishes its ready wait first and enters the DMA arbiter with its lock bit set owns the GPMI command interface and through it owns the BCH resources for the duration of its processing.

16.4.1 BCH Encoding for NAND Writes

The BCH encoder flowchart in Figure 16-6 shows the detailed steps involved in programming and using the BCH encoder. This flowchart shows how to use the BCH block with the GPMI.

To use the BCH encoder with the GPMI's DMA, create a DMA command chain containing ten descriptor structures, as shown in Figure 16-8 and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The ten descriptors perform the following tasks:

1. Disable the BCH block (in case it was enabled) and issue NAND write setup command byte (under CLE) and address bytes (under ALE).
2. Configure and enable the BCH and GPMI blocks to perform the NAND write.
3. Disable the BCH block and issue NAND write execute command byte (under CLE).
4. Wait for the NAND device to finish writing the data by watching the ready signal.
5. Check for NAND timeout through PSENSE.

6. Issue NAND status command byte (under CLE).
7. Read the status and compare against expected.
8. If status is incorrect or incomplete, branch to error handling descriptor chain.
9. Otherwise, write is complete and emit GPMI interrupt.

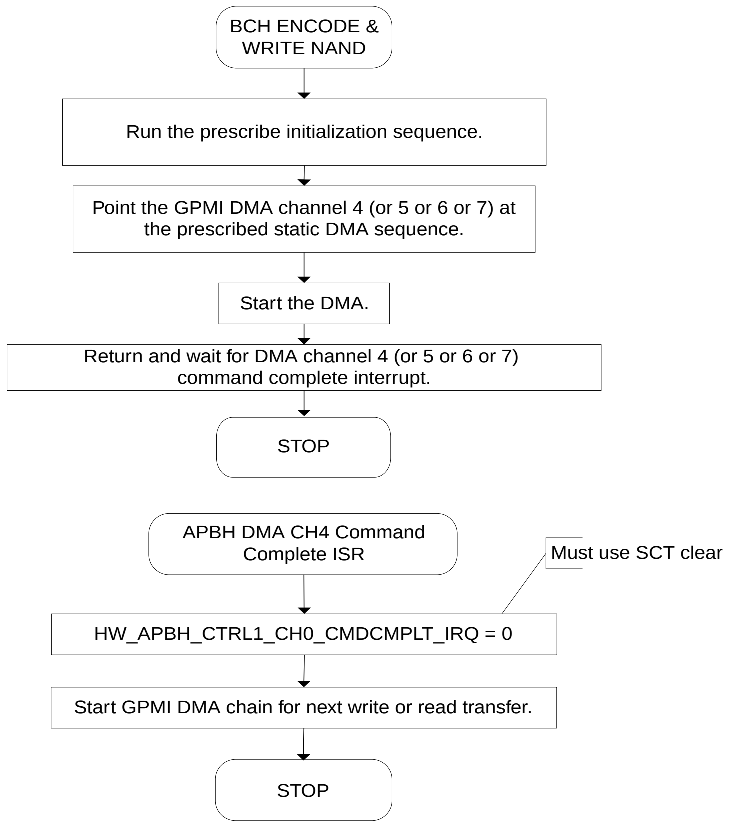
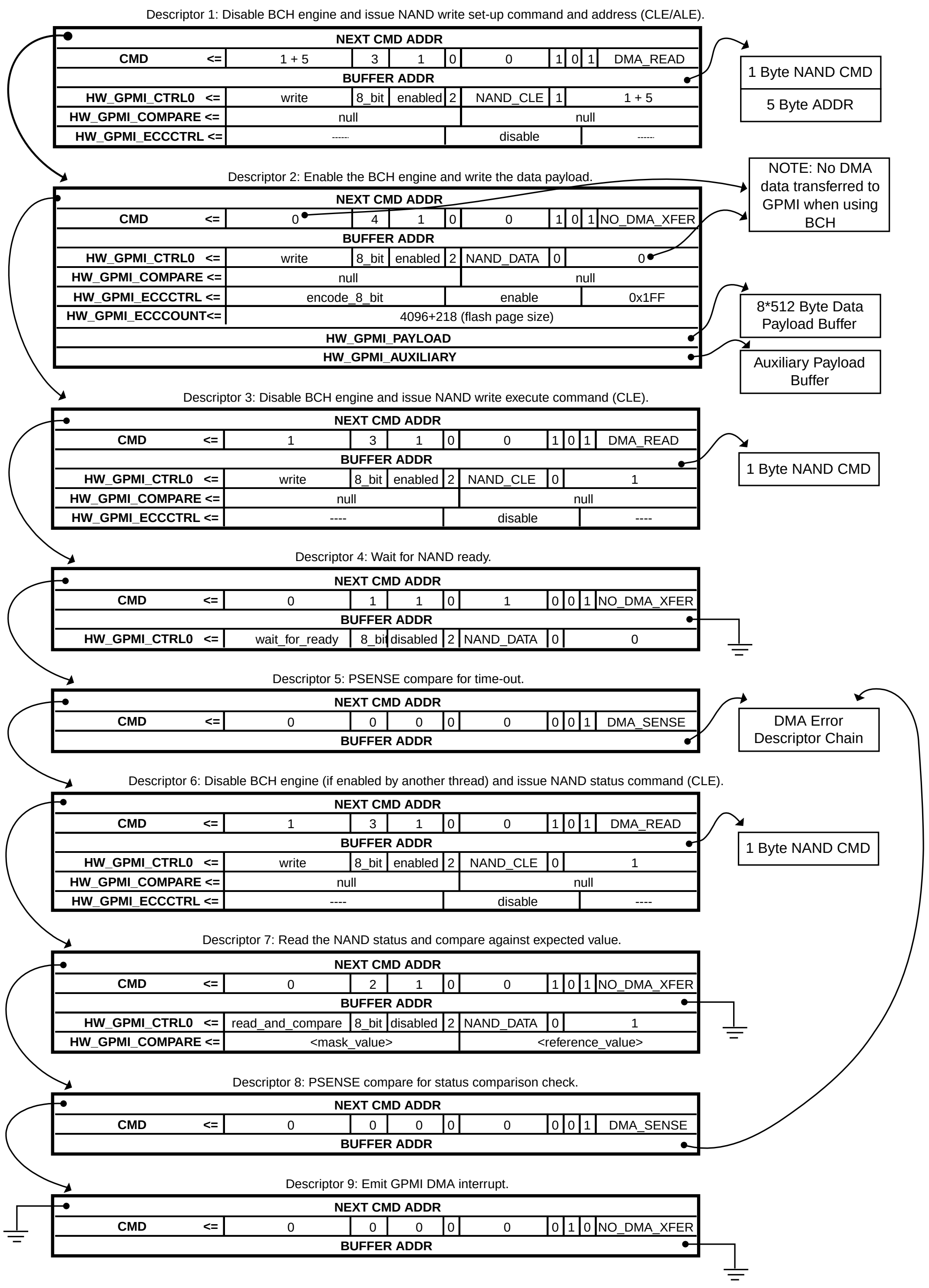

BCH ENCODE & WRITE NAND
Run the prescribe initialization sequence.
Point the GPMI DMA channel 4 (or 5 or 6 or 7) at the prescribed static DMA sequence.
Start the DMA.
Return and wait for DMA channel 4 (or 5 or 6 or 7) command complete interrupt.
STOP
APBH DMA CH4 Command Complete ISR
Must use SCT clear
HW_APBH_CTRL1_CH0_CMDCMPLT_IRQ = 0
Start GPMI DMA chain for next write or read transfer.
STOP

Figure 16-6. BCH Encode Flowchart

Descriptor Legend

NEXT CMD ADDR										
CMD	<=	xfer_count	cmdwords	wait4endcmd	semaphore	nandwait4ready	nandlock	irqoncmplt	chain	command
BUFFER ADDR										
HW_GPMI_CTRL0	<=	command_mode	word_length	lock_cs	CS	address	address_increment	xfer_count		
HW_GPMI_COMPARE	<=	mask				reference				
HW_GPMI_ECCCTRL	<=	ecc_cmd			enable_ecc				buffer_mask	
HW_GPMI_ECCCOUNT										
HW_GPMI_PAYLOAD										
HW_GPMI_AUXILIARY										

Figure 16-7. BCH DMA Descriptor Legend



Descriptor 1: Disable BCH engine and issue NAND write set-up command and address (CLE/ALE).
NEXT CMD ADDR
CMD <= | 1 + 5 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | DMA_READ
BUFFER ADDR
HW_GPMI_CTRL0 <= | write | 8_bit | enabled | 2 | NAND_CLE | 1 | 1 + 5
HW_GPMI_COMPARE <= | null | null
HW_GPMI_ECCCTRL <= | ------ | disable | ------
1 Byte NAND CMD
5 Byte ADDR
Descriptor 2: Enable the BCH engine and write the data payload.
NEXT CMD ADDR
CMD <= | 0 | 4 | 1 | 0 | 0 | 1 | 0 | 1 | NO_DMA_XFER
BUFFER ADDR
HW_GPMI_CTRL0 <= | write | 8_bit | enabled | 2 | NAND_DATA | 0 | 0
HW_GPMI_COMPARE <= | null | null
HW_GPMI_ECCCTRL <= | encode_8_bit | enable | 0x1FF
HW_GPMI_ECCCOUNT<= | 4096+218 (flash page size)
HW_GPMI_PAYLOAD
HW_GPMI_AUXILIARY
NOTE: No DMA data transferred to GPMI when using BCH
8*512 Byte Data Payload Buffer
Auxiliary Payload Buffer
Descriptor 3: Disable BCH engine and issue NAND write execute command (CLE).
NEXT CMD ADDR
CMD <= | 1 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | DMA_READ
BUFFER ADDR
HW_GPMI_CTRL0 <= | write | 8_bit | enabled | 2 | NAND_CLE | 0 | 1
HW_GPMI_COMPARE <= | null | null
HW_GPMI_ECCCTRL <= | ---- | disable | ----
1 Byte NAND CMD
Descriptor 4: Wait for NAND ready.
NEXT CMD ADDR
CMD <= | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | NO_DMA_XFER
BUFFER ADDR
HW_GPMI_CTRL0 <= | wait_for_ready | 8_bit | disabled | 2 | NAND_DATA | 0 | 0
Descriptor 5: PSENSE compare for time-out.
NEXT CMD ADDR
CMD <= | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | DMA_SENSE
BUFFER ADDR
DMA Error Descriptor Chain
Descriptor 6: Disable BCH engine (if enabled by another thread) and issue NAND status command (CLE).
NEXT CMD ADDR
CMD <= | 1 | 3 | 1 | 0 | 0 | 1 | 0 | 1 | DMA_READ
BUFFER ADDR
HW_GPMI_CTRL0 <= | write | 8_bit | enabled | 2 | NAND_CLE | 0 | 1
HW_GPMI_COMPARE <= | null | null
HW_GPMI_ECCCTRL <= | ---- | disable | ----
1 Byte NAND CMD
Descriptor 7: Read the NAND status and compare against expected value.
NEXT CMD ADDR
CMD <= | 0 | 2 | 1 | 0 | 0 | 1 | 0 | 1 | NO_DMA_XFER
BUFFER ADDR
HW_GPMI_CTRL0 <= | read_and_compare | 8_bit | disabled | 2 | NAND_DATA | 0 | 1
HW_GPMI_COMPARE <= | <mask_value> | <reference_value>
Descriptor 8: PSENSE compare for status comparison check.
NEXT CMD ADDR
CMD <= | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | DMA_SENSE
BUFFER ADDR
Descriptor 9: Emit GPMI DMA interrupt.
NEXT CMD ADDR
CMD <= | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | NO_DMA_XFER
BUFFER ADDR

Figure 16-8. BCH Encode DMA Descriptor Chain

16.4.1.1 DMA Structure Code Example

The following code sample illustrates the coding for one write transaction involving 4096 bytes of data payload (eight 512-byte blocks) and 10 bytes of auxiliary payload (also referred to as metadata) to a 4K NAND page sitting on GPMI CS2.

```
//----------------------------------------------------------------------------
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//----------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//----------------------------------------------------------------------------
// allocate 10 descriptors for doing a NAND ECC Write
//----------------------------------------------------------------------------
GENERIC_DESCRIPTOR write[10];
//----------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//----------------------------------------------------------------------------
unsigned int * dma_error_handler;
//----------------------------------------------------------------------------
// 8 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is write setup command
//   bytes 1-5 is the NAND address
//   byte 6 is write execute command
//   byte 7 is status command
//----------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[8];
//----------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//----------------------------------------------------------------------------
unsigned int write_payload_buffer[(4096/4)];
//----------------------------------------------------------------------------
// 65 byte meta-data to be written to NAND
// needs to be word aligned
//----------------------------------------------------------------------------
unsigned int write_aux_buffer[65];
//----------------------------------------------------------------------------
// Descriptor 1: issue NAND write setup command (CLE/ALE)
//----------------------------------------------------------------------------
write[0].dma_nxtcmdar = &write[1];                          // point to the next descriptor
write[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5)|   // 1 byte command, 5 byte address
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |   // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish before

                                                            //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels from

                                                            //    taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                 BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ); // read data from DMA, write to NAND
write[0].dma_bar = &nand_cmd_addr_buffer;          // byte 0 write setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
write[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                   BF_GPMI_CTRL0_CS              (2)      | // must correspond to NAND CS used

                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and address

                  BF_GPMI_CTRL0_XFER_COUNT       (1 + 5);    // 1 byte command, 5 byte address
write[0].gpmi_compare = NULL;                             // field not used but necessary to set
 eccctrl
write[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//----------------------------------------------------------------------------
// Descriptor 2: write the data payload (DATA)
//----------------------------------------------------------------------------
write[1].dma_nxtcmdar = &write[2];                        // point to the next descriptor
write[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT      (0)|  // NOTE: No DMA data transfer
                   BF_APBH_CHn_CMD_CMDWORDS        (4)|  // send 4 words to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD     (1)|  // Wait to end
                   BF_APBH_CHn_CMD_SEMAPHORE       (0)|
                   BF_APBH_CHn_CMD_NANDWAIT4READY  (0)|
                   BF_APBH_CHn_CMD_NANDLOCK        (1)|  // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT      (0)|
                   BF_APBH_CHn_CMD_CHAIN           (1)|  // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_NO_XFER);  // No data transferred
write[1].dma_bar = &write_payload_buffer;                      // pointer for the 4K byte data
 area
// 4 words sent to the GPMI
write[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                   BF_GPMI_CTRL0_CS                 (2)       | // must correspond to NAND CS
 used
                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)|
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);       // NOTE: this field contains

                                                                 //  the total amount
                              // DMA transferred to GPMI via DMA (0)!
write[1].gpmi_compare = NULL;                                 // field not used but necessary to

set eccctrl
write[1].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    ENCODE_8_BIT) | // specify t = 8 mode

                        BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       | // enable ECC module
                   BF_GPMI_ECCCTRL_BUFFER_MASK   (0x1FF);          // write all 8 data blocks

                                                                     //   and 1 aux block
write[1].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);      // specify number of bytes
                                                                //   written to NAND
write[1].gpmi_data_pointer = &write_payload_pointer;           // data buffer address
write[1].gpmi_aux_pointer  = &write_aux_pointer;               // metadata pointer
//----------------------------------------------------------------------------
// Descriptor 3: issue NAND write execute command (CLE)
//----------------------------------------------------------------------------
write[2].dma_nxtcmdar = &write[3];                          // point to the next descriptor
write[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1) |      // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3) |      // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |     // wait for command to finish before

                                                            //    continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0) |
                   BF_APBH_CHn_CMD_NANDLOCK      (1) |     // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                   BF_APBH_CHn_CMD_CHAIN         (1) |     // follow chain to next command
               BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);   // read data from DMA, write to NAND
write[2].dma_bar = &nand_cmd_addr_buffer[6];             // point to byte 6, write execute
command
// 3 words sent to the GPMI
write[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                  BF_GPMI_CTRL0_CS              (2)      | // must correspond to NAND CS used

                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
write[2].gpmi_compare = NULL;                         // field not used but necessary to set
eccctrl
write[2].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//----------------------------------------------------------------------------
// Descriptor 4: wait for ready (CLE)
//----------------------------------------------------------------------------
write[3].dma_nxtcmdar = &write[4];                       // point to the next descriptor

write[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0) |   // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (1) |   // send 1 word to the GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (1) |   // wait for command to finish before
                                                         //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0) |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(1) |   // wait for nand to be ready
                   BF_APBH_CHn_CMD_NANDLOCK      (0) |   // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0) |
                   BF_APBH_CHn_CMD_CHAIN         (1) |   // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);         // no dma transfer
write[3].dma_bar = NULL;                                               // field not used
// 1 word sent to the GPMI
write[3].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND ready

                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)         |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)      |
                BF_GPMI_CTRL0_CS              (2)         | // must correspond to NAND CS used

                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)     |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)            |
                      BF_GPMI_CTRL0_XFER_COUNT         (0);
//----------------------------------------------------------------------------
// Descriptor 5: psense compare (time out check)
//----------------------------------------------------------------------------
write[4].dma_nxtcmdar = &write[5];                            // point to the next descriptor
write[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[4].dma_bar = dma_error_handler;              // if sense check fails, branch to error
handler
//----------------------------------------------------------------------------
// Descriptor 6: issue NAND status command (CLE)
//----------------------------------------------------------------------------
write[5].dma_nxtcmdar = &write[6];                         // point to the next descriptor
write[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |  // 1 byte command
                   BF_APBH_CHn_CMD_CMDWORDS      (3)    |  // send 3 words to the GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |  // wait for command to finish before
```

```
continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |  // prevent other DMA channels from
taking over
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                   BF_APBH_CHn_CMD_CHAIN         (1)    |  // follow chain to next command
                BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to NAND
write[5].dma_bar = &nand_cmd_addr_buffer[7];               // point to byte 7, status command
write[5].gpmi_compare = NULL;                        // field not used but necessary to set
eccctrl
write[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
// 3 words sent to the GPMI
write[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                   BF_GPMI_CTRL0_CS               (2)     | // must correspond to NAND CS used

                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
//----------------------------------------------------------------------------
// Descriptor 7: read status and compare (DATA)
//----------------------------------------------------------------------------
write[6].dma_nxtcmdar = &write[7];                           // point to the next descriptor
write[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)  |    // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (2)  |    // send 2 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)  |    // wait for command to finish before

                                                           //      continuing
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)  |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)  |
                   BF_APBH_CHn_CMD_NANDLOCK      (1)  |    // maintain resource lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)  |
                   BF_APBH_CHn_CMD_CHAIN         (1)  |    // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
write[6].dma_bar = NULL;                                        // field not used
// 2 word sent to the GPMI
write[6].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ_AND_COMPARE) | // read from the
                                                                           //       NAND and
//                                                                       // compare to expect
                      BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                      BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                  BF_GPMI_CTRL0_CS             (2)       |   // must correspond to NAND CS used

                      BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                      BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                      BF_GPMI_CTRL0_XFER_COUNT         (1);
write[6].gpmi_compare = <MASK_AND_REFERENCE_VALUE>;  // NOTE: mask and reference values are
NAND
                                                            // SPECIFIC to evaluate the NAND
status
//----------------------------------------------------------------------------
// Descriptor 8: psense compare (time out check)
//----------------------------------------------------------------------------
write[7].dma_nxtcmdar = &write[8];                              // point to the next descriptor
write[7].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)        | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)        | // relinquish nand lock
                   BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                   BF_APBH_CHn_CMD_CHAIN         (1)        | // follow chain to next command
                   BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);  // perform a sense check
write[7].dma_bar = dma_error_handler;             // if sense check fails, branch to error
handler
//----------------------------------------------------------------------------
```

```
// Descriptor 9: emit GPMI interrupt
//----------------------------------------------------------------------------
write[8].dma_nxtcmdar = NULL;                                  // not used since this is last

descriptor
write[8].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)          | // no dma transfer
                   BF_APBH_CHn_CMD_CMDWORDS      (0)          | // no words sent to GPMI
                   BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)          | // do not wait to continue
                   BF_APBH_CHn_CMD_SEMAPHORE     (0)          |
                   BF_APBH_CHn_CMD_NANDWAIT4READY(0)          |
                   BF_APBH_CHn_CMD_NANDLOCK      (0)          |
                   BF_APBH_CHn_CMD_IRQONCMPLT    (1)          | // emit GPMI interrupt
                 BF_APBH_CHn_CMD_CHAIN        (0)         | // terminate DMA chain processing

                   BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);  // no dma transfer
```

16.4.1.2 Using the BCH Encoder

To use the BCH encoder, first turn off the module-wide soft reset bit in both the GPMI and BCH blocks before starting any DMA activity. Turning off the soft reset must take place by itself, prior to programming the rest of the control registers. Turn off the BCH bus master soft reset bit (bit 29). Turn off the clock gate bits.

Program the remainder of the GPMI, BCH and APBH DMA as follows:

```
// bring APBH out of reset
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_SFRST);
HW_APBH_CTRL0_CLR(BM_APBH_CTRL0_CLKGATE);
// bring BCH out of reset
HW_BCH_CTRL_CLR(BM_BCH_CTRL_SFTRST);
HW_BCH_CTRL_CLR(BM_BCH_CTRL_CLKGATE);
// bring gpmi out of reset
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
HW_GPMI_CTRL1_SET(BM_GPMI_CTRL1_DEV_RESET | // deassert reset
              BM_GPMI_CTRL1_BCH_MODE  ); // enable BCH mode
// enable pinctrl
HW_PINCTRL_CTRL_WR(0x00000000);
// enable GPMI through alt pin wiring
HW_PINCTRL_MUXSEL0_CLR(0xff000000);
HW_PINCTRL_MUXSEL0_SET(0xaa000000);
// to use the primary pins do the following
//    HW_PINCTRL_MUXSEL4_CLR(0xff000000);
//    HW_PINCTRL_MUXSEL4_SET(0x55000000);
// enable gpmi pins
HW_PINCTRL_MUXSEL0_CLR(0x0000ffff); // data bits
HW_PINCTRL_MUXSEL1_CLR(0x000fffff); // control bits
```

Note that for writing NANDs (ECC encoding), only GPMI DMA command complete interrupts are used. The BCH engine is used for writing to the NAND but may optionally produces an interrupt. From the sample code in DMA Structure Code Example:

- DMA descriptor 1 prepares the NAND for data write by using the GPMI to issue a write setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is disabled and not used for these commands.

- DMA descriptor 2 enables the BCH engine for encoding to begin the initial writing of the NAND data by specifying where the data and auxiliary payload are coming from in system memory.
- DMA descriptor 3 issues the write commit command byte under CLE to the NAND.
- DMA descriptor 4 waits for the NAND to complete the write commit/transfer by watching the NAND's ready line status. This descriptor relinquishes the NANDLOCK on the GPMI to enable the other DMA channels to initiate NAND transactions on different NAND CS lines.
- DMA descriptor 6 issues a NAND status command byte under "CLE" to check the status of the NAND device following the page write.
- DMA descriptor 7 reads back the NAND status and compares the status with an expected value. If there are differences, then the DMA processing engine follows an error-handling DMA descriptor path.
- DMA descriptor 8 disables the BCH engine and emits a GPMI interrupt to indicate that the NAND write has been completed.

16.4.2 BCH Decoding for NAND Reads

When a page is read from NAND flash, BCH syndromes will be computed and, if correctable errors are found, they will be corrected on a per block basis within the NAND page. This decoding process is fully overlapped with other NAND data reads and with CPU execution. The BCH decoder flowchart in Figure 16-9 shows the steps involved in programming the decoder. The hardware flow of reading and decoding a 4096-byte page is shown in Figure 16-10.

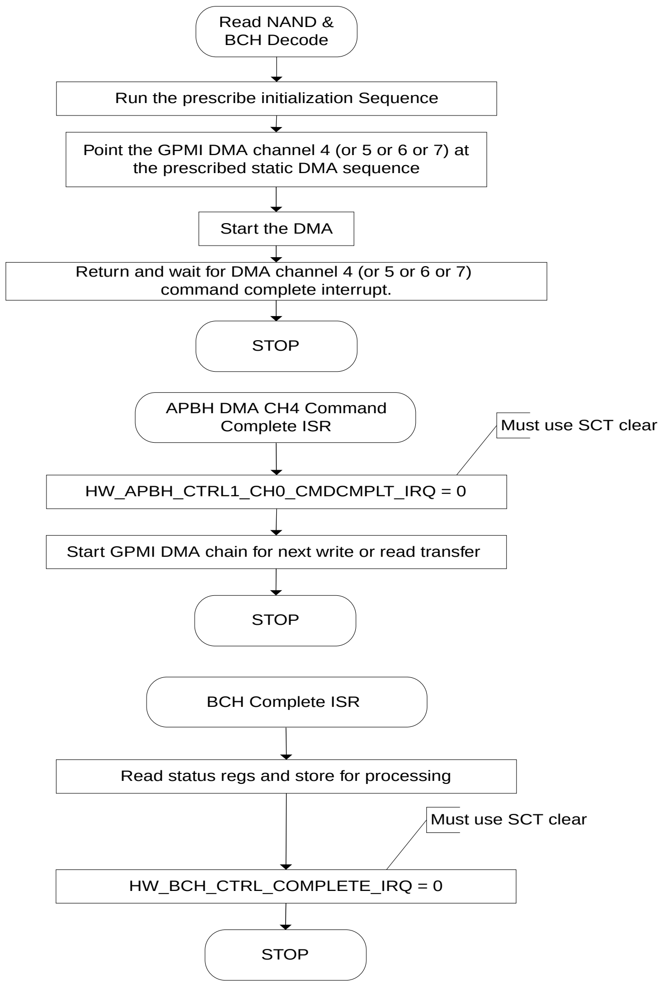

Read NAND & BCH Decode
Run the prescribe initialization Sequence
Point the GPMI DMA channel 4 (or 5 or 6 or 7) at the prescribed static DMA sequence
Start the DMA
Return and wait for DMA channel 4 (or 5 or 6 or 7) command complete interrupt.
STOP
APBH DMA CH4 Command Complete ISR
Must use SCT clear
HW_APBH_CTRL1_CH0_CMDCMPLT_IRQ = 0
Start GPMI DMA chain for next write or read transfer
STOP
BCH Complete ISR
Read status regs and store for processing
Must use SCT clear
HW_BCH_CTRL_COMPLETE_IRQ = 0
STOP

Figure 16-9. BCH Decode Flowchart

Conceptually, an APHB DMA Channel 4, 5, 6, or 7 command chain with seven command structures linked together is used to perform the BCH decode operation (as shown in Figure 16-10).

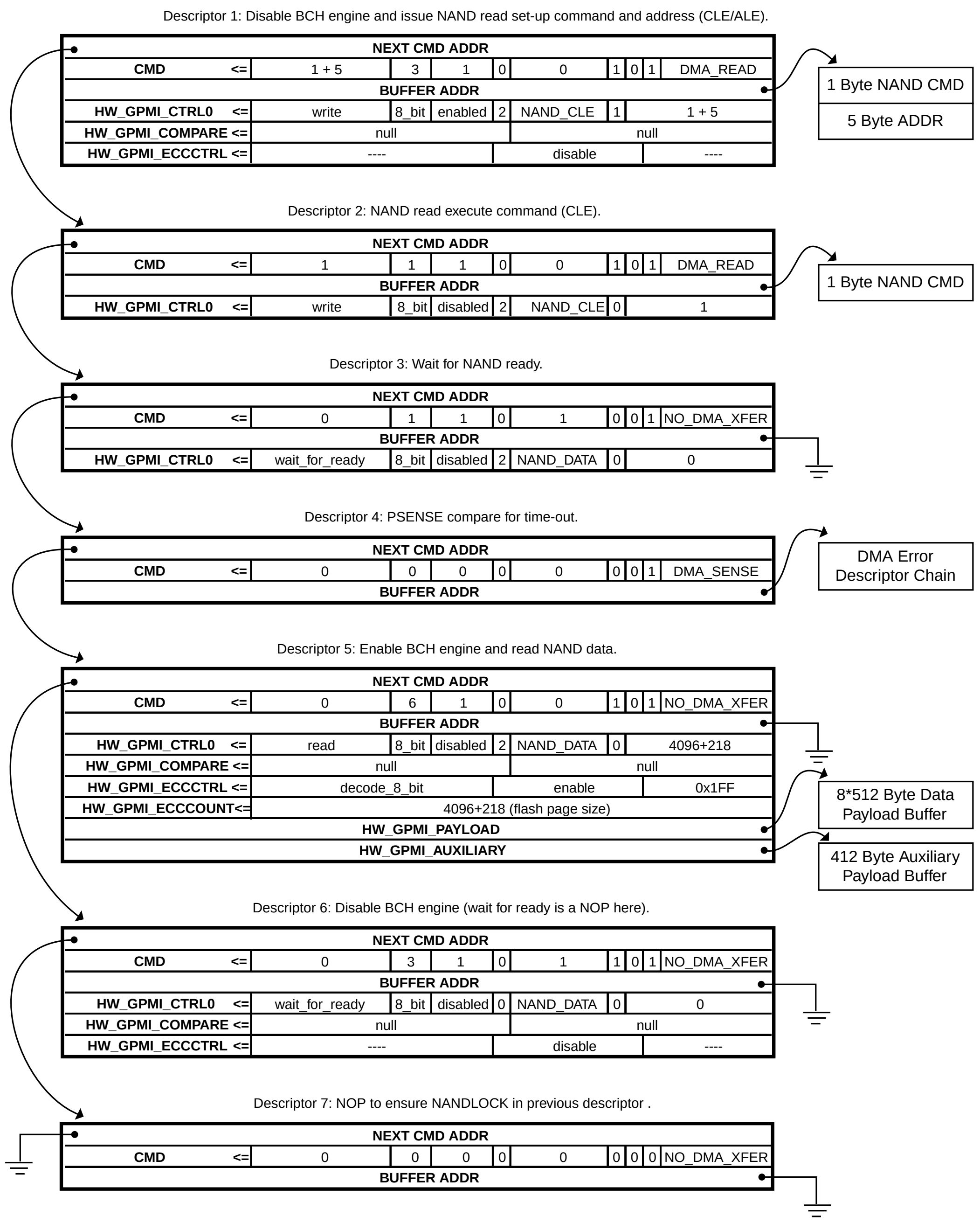
Note

The GPMI's DMA command structures controls the BCH decode operation.

To use the BCH decoder with the GPMI's DMA, create a DMA command chain containing seven descriptor structures, as shown in Figure 16-10 and detailed in the DMA structure code example that follows it in DMA Structure Code Example. The seven DMA descriptors perform the following tasks:

1. Issue NAND read setup command byte (under "CLE") and address bytes (under "ALE").
2. Issue NAND read execute command byte (under "CLE").

3. Wait for the NAND device to complete accessing the block data by watching the ready signal.
4. Check for NAND timeout through "PSENSE".
5. Configure and enable the BCH block and read the NAND block data.
6. Disable the BCH block.
7. Descriptor NOP to allow NANDLOCK in the previous descriptor to the thread-safe.


1 Byte NAND CMD
5 Byte ADDR
1 Byte NAND CMD
DMA Error Descriptor Chain
8*512 Byte Data Payload Buffer
412 Byte Auxiliary Payload Buffer

Descriptor 1: Disable BCH engine and issue NAND read set-up command and address (CLE/ALE).

NEXT CMD ADDR									
CMD <=	1 + 5	3	1	0	0	1	0	1	DMA_READ
BUFFER ADDR									
HW_GPMI_CTRL0 <=	write	8_bit	enabled	2	NAND_CLE	1	1 + 5		
HW_GPMI_COMPARE <=	null	null							
HW_GPMI_ECCCTRL <=	----	disable	----						

Descriptor 2: NAND read execute command (CLE).

NEXT CMD ADDR									
CMD <=	1	1	1	0	0	1	0	1	DMA_READ
BUFFER ADDR									
HW_GPMI_CTRL0 <=	write	8_bit	disabled	2	NAND_CLE	0	1		

Descriptor 3: Wait for NAND ready.

NEXT CMD ADDR									
CMD <=	0	1	1	0	1	0	0	1	NO_DMA_XFER
BUFFER ADDR									
HW_GPMI_CTRL0 <=	wait_for_ready	8_bit	disabled	2	NAND_DATA	0	0		

Descriptor 4: PSENSE compare for time-out.

NEXT CMD ADDR									
CMD <=	0	0	0	0	0	0	0	1	DMA_SENSE
BUFFER ADDR									

Descriptor 5: Enable BCH engine and read NAND data.

NEXT CMD ADDR									
CMD <=	0	6	1	0	0	1	0	1	NO_DMA_XFER
BUFFER ADDR									
HW_GPMI_CTRL0 <=	read	8_bit	disabled	2	NAND_DATA	0	4096+218		
HW_GPMI_COMPARE <=	null	null							
HW_GPMI_ECCCTRL <=	decode_8_bit	enable	0x1FF						
HW_GPMI_ECCCOUNT<=	4096+218 (flash page size)								
HW_GPMI_PAYLOAD									
HW_GPMI_AUXILIARY									

Descriptor 6: Disable BCH engine (wait for ready is a NOP here).

NEXT CMD ADDR									
CMD <=	0	3	1	0	1	1	0	1	NO_DMA_XFER
BUFFER ADDR									
HW_GPMI_CTRL0 <=	wait_for_ready	8_bit	disabled	0	NAND_DATA	0	0		
HW_GPMI_COMPARE <=	null	null							
HW_GPMI_ECCCTRL <=	----	disable	----						

Descriptor 7: NOP to ensure NANDLOCK in previous descriptor .

NEXT CMD ADDR									
CMD <=	0	0	0	0	0	0	0	0	NO_DMA_XFER
BUFFER ADDR									

Figure 16-10. BCH Decode DMA Descriptor Chain

16.4.2.1 DMA Structure Code Example

The following sample code illustrates the coding for one read transaction, consisting of a seven DMA command structure chain for reading all 4096 bytes of payload data (eight 512-byte blocks) and 65 bytes of metadata with the associative parity bytes (8 * (18) + 9) from a 4K NAND page sitting on GPMI CS2.

```
//-------------------------------------------------------------------------
```

```
// generic DMA/GPMI/ECC descriptor struct, order sensitive!
//------------------------------------------------------------------------------
typedef struct {
  // DMA related fields
  unsigned int dma_nxtcmdar;
  unsigned int dma_cmd;
  unsigned int dma_bar;
  // GPMI related fields
  unsigned int gpmi_ctrl0;
  unsigned int gpmi_compare;
  unsigned int gpmi_eccctrl;
  unsigned int gpmi_ecccount;
  unsigned int gpmi_data_ptr;
  unsigned int gpmi_aux_ptr;
} GENERIC_DESCRIPTOR;
//------------------------------------------------------------------------------
// allocate 7 descriptors for doing a NAND ECC Read
//------------------------------------------------------------------------------
GENERIC_DESCRIPTOR read[7];
//------------------------------------------------------------------------------
// DMA descriptor pointer to handle error conditions from psense checks
//------------------------------------------------------------------------------
unsigned int * dma_error_handler;
//------------------------------------------------------------------------------
// 7 byte NAND command and address buffer
// any alignment is ok, it is read by the GPMI DMA
//   byte 0 is read setup command
//   bytes 1-5 is the NAND address
//   byte 6 is read execute command
//------------------------------------------------------------------------------
unsigned char nand_cmd_addr_buffer[7];
//------------------------------------------------------------------------------
// 4096 byte payload buffer used for reads or writes
// needs to be word aligned
//------------------------------------------------------------------------------
unsigned int read_payload_buffer[(4096/4)];
//------------------------------------------------------------------------------
// 412 byte auxiliary buffer used for reads
// needs to be word aligned
//------------------------------------------------------------------------------
unsigned int read_aux_buffer[(412/4)];
//------------------------------------------------------------------------------
// Descriptor 1: issue NAND read setup command (CLE/ALE)
//------------------------------------------------------------------------------
read[0].dma_nxtcmdar = &read[1];                           // point to the next descriptor
read[0].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1 + 5) |  // 1 byte command, 5 byte address
                  BF_APBH_CHn_CMD_CMDWORDS      (3)     |  // send 3 words to the GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)     |  // wait for command to finish
                                                           //     before continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)     |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)     |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)     |  // prevent other DMA channels from
                                                           //    taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)     |
                  BF_APBH_CHn_CMD_CHAIN         (1)     |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);  // read data from DMA, write to
NAND
read[0].dma_bar = &nand_cmd_addr_buffer;            // byte 0 read setup, bytes 1 - 5 NAND
address
// 3 words sent to the GPMI
read[0].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      ENABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)       | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (1)       | // send command and address
                     BF_GPMI_CTRL0_XFER_COUNT         (1 + 5);    // 1 byte command, 5 byte
```

```
address
read[0].gpmi_compare = NULL;                          // field not used but necessary to set
eccctrl
read[0].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE); // disable the ECC block
//------------------------------------------------------------------------------
// Descriptor 2: issue NAND read execute command (CLE)
//------------------------------------------------------------------------------
read[1].dma_nxtcmdar = &read[2];                           // point to the next descriptor
read[1].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (1)    |   // 1 byte read command
                  BF_APBH_CHn_CMD_CMDWORDS      (1)    |   // send 1 word to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)    |   // wait for command to finish before

                                                           //    continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)    |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)    |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)    |   // prevent other DMA channels from
                                                           //     taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)    |
                  BF_APBH_CHn_CMD_CHAIN         (1)    |   // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_READ);     // read data from DMA, write to
 NAND
read[1].dma_bar = &nand_cmd_addr_buffer[6];               // point to byte 6, read execute
command
// 1 word sent to the GPMI
read[1].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WRITE)    | // write to the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)    |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED) |
                     BF_GPMI_CTRL0_CS                 (2)       | // must correspond to NAND
CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_CLE) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)       |
                     BF_GPMI_CTRL0_XFER_COUNT         (1);        // 1 byte command
//------------------------------------------------------------------------------
// Descriptor 3: wait for ready (DATA)
//------------------------------------------------------------------------------
read[2].dma_nxtcmdar = &read[3];                           // point to the next descriptor
read[2].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (1)      | // send 1 word to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish before

                                                           //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (0)      | // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);   // no dma transfer
read[2].dma_bar = NULL;                                         // field not used
// 1 word sent to the GPMI
read[2].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, WAIT_FOR_READY) | // wait for NAND ready

                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)          |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)       |
                     BF_GPMI_CTRL0_CS                 (2)             | // must correspond to
 NAND CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)      |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)             |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
//------------------------------------------------------------------------------
// Descriptor 4: psense compare (time out check)
//------------------------------------------------------------------------------
read[3].dma_nxtcmdar = &read[4];                               // point to the next descriptor
read[3].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)        |      // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)        |      // no words sent to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)        |      // do not wait to continue
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)        |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)        |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)        |
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)        |
                  BF_APBH_CHn_CMD_CHAIN         (1)        |      // follow chain to next
command
                  BV_FLD(APBH_CHn_CMD, COMMAND, DMA_SENSE);       // perform a sense check
read[3].dma_bar = dma_error_handler;                      // if sense check fails, branch to
error handler
//----------------------------------------------------------------------------
// Descriptor 5: read 4K page plus 65 byte meta-data Nand data
//               and send it to ECC block (DATA)
//----------------------------------------------------------------------------
read[4].dma_nxtcmdar = &read[5];                        // point to the next descriptor
read[4].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)   | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (6)   | // send 6 words to GPMI
                  BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)   | // wait for command to finish before
                                                        //       continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)   |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)   |
                  BF_APBH_CHn_CMD_NANDLOCK      (1)   | // prevent other DMA channels from
taking over
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)   | // ECC block generates BCH interrupt
                                                        //      on completion
                  BF_APBH_CHn_CMD_CHAIN         (1)   | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);     // no DMA transfer,
                                                                 // ECC block handles transfer
read[4].dma_bar = NULL;                                           // field not used
// 6 words sent to the GPMI
read[4].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)       |  // read from the NAND
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)      |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)   |
                    BF_GPMI_CTRL0_CS                (2)         |  // must correspond to NAND
 CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA)  |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)         |
                     BF_GPMI_CTRL0_XFER_COUNT         (4096+218);    // eight 512 byte data
blocks
                                                                     // metadata, and parity

read[4].gpmi_compare = NULL;                            // field not used but necessary to set
eccctrl
// GPMI ECCCTRL PIO This launches the 4K byte transfer through BCH's
// bus master. Setting the ECC_ENABLE bit redirects the data flow
// within the GPMI so that read data flows to the BCH engine instead
// of flowing to the GPMI's DMA channel.
read[4].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ECC_CMD,    DECODE_8_BIT) |    // specify t = 8
mode
                      BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, ENABLE)       |    // enable ECC module

                      BF_GPMI_ECCCTRL_BUFFER_MASK     (0X1FF);  // read all 8 data blocks and
 1 aux block
read[4].gpmi_ecccount = BF_GPMI_ECCCOUNT_COUNT(4096+218);        // specify number of bytes
                                                                 //        read from NAND
read[4].gpmi_data_ptr = &read_payload_buffer;                    // pointer for the 4K byte
                                                                 //    data area
read[4].gpmi_aux_ptr = &read_aux_buffer;                         // pointer for the 65 byte
aux area +
                                                                 // parity and syndrome bytes
 for both
                                                                // data and aux blocks.
//----------------------------------------------------------------------------
// Descriptor 6: disable ECC block
//----------------------------------------------------------------------------
read[5].dma_nxtcmdar = &read[6];                           // point to the next descriptor
read[5].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)      | // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (3)      | // send 3 words to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (1)      | // wait for command to finish before

                                                           //     continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)      |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(1)      | // wait for nand to be ready
                  BF_APBH_CHn_CMD_NANDLOCK      (1)      | // need nand lock to be
                                                           // thread safe while turn-off BCH
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)      |
                  BF_APBH_CHn_CMD_CHAIN         (1)      | // follow chain to next command
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[5].dma_bar = NULL;                                            // field not used
// 3 words sent to the GPMI
read[5].gpmi_ctrl0 = BV_FLD(GPMI_CTRL0, COMMAND_MODE, READ)      |
                     BV_FLD(GPMI_CTRL0, WORD_LENGTH,  8_BIT)     |
                     BV_FLD(GPMI_CTRL0, LOCK_CS,      DISABLED)  |
                     BF_GPMI_CTRL0_CS                 (2)        | // must correspond to NAND
 CS used
                     BV_FLD(GPMI_CTRL0, ADDRESS,      NAND_DATA) |
                     BF_GPMI_CTRL0_ADDRESS_INCREMENT  (0)        |
                     BF_GPMI_CTRL0_XFER_COUNT         (0);
read[5].gpmi_compare = NULL;                          // field not used but necessary to set
eccctrl
read[5].gpmi_eccctrl = BV_FLD(GPMI_ECCCTRL, ENABLE_ECC, DISABLE);  // disable the ECC block
//----------------------------------------------------------------------
// Descriptor 7: deassert nand lock
//----------------------------------------------------------------------
read[6].dma_nxtcmdar = NULL;                               // not used since this is last
descriptor
read[6].dma_cmd = BF_APBH_CHn_CMD_XFER_COUNT    (0)   |    // no dma transfer
                  BF_APBH_CHn_CMD_CMDWORDS      (0)   |    // no words sent to GPMI
                 BF_APBH_CHn_CMD_WAIT4ENDCMD   (0)   |    // wait for command to finish before

                                                           //        continuing
                  BF_APBH_CHn_CMD_SEMAPHORE     (0)   |
                  BF_APBH_CHn_CMD_NANDWAIT4READY(0)   |
                  BF_APBH_CHn_CMD_NANDLOCK      (0)   |    // relinquish nand lock
                  BF_APBH_CHn_CMD_IRQONCMPLT    (0)   |    // BCH engine generates interrupt
                  BF_APBH_CHn_CMD_CHAIN         (0)   |    // terminate DMA chain processing
                  BV_FLD(APBH_CHn_CMD, COMMAND, NO_DMA_XFER);      // no dma transfer
read[6].dma_bar = NULL;                                            // field not used
```

16.4.2.2 Using the Decoder

As illustrated in Figure 16-10 and the sample code in DMA Structure Code Example:

- DMA descriptor 1 prepares the NAND for data read by using the GPMI to issue a NAND read setup command byte under CLE, then sends a 5-byte address under ALE. The BCH engine is not used for these commands.
- DMA descriptor 2 issues a one-byte read execute command to the NAND device that triggers its read access. The NAND then goes not ready.
- DMA descriptor 3 performs a wait for ready operation allowing the DMA chain to remain dormant until the NAND device completes its read access time.
- DMA descriptor 5 handles the reading and error correction of the NAND data. This command's PIOs activate the BCH engine to write the read NAND data to system memory and to process it for any errors that need to be corrected. This DMA descriptor contains two PIO values that are system memory addresses pointing to the PAYLOAD data area and to the AUXILIARY data area. These addresses are used by the BCH engine's AHB master to move data into system memory and to correct it. While this

example is reading an entire 4K page—payload plus metadata—it is equally possible to read just one 512-byte payload block or just the uniquely protected metadata block in a single 7 DMA structure transfer.

- DMA descriptor 6 disables the BCH engine with the NANDLOCK asserted. This is necessary to ensure that the GPMI resource is not arbitrated to another DMA channel when multiple DMA channels are active concurrently.
- DMA descriptor 7 deasserts the NANDLOCK to free up the GPMI resource to another channel.

As the BCH block receives data from the GPMI:

- The decoder transforms the read NAND data block into a BCH code word and computes the codeword syndrome.
- If no errors are present, then the BCH block can immediately report back to firmware. This report is passed as the HW_BCH_CTRL_COMPLETE_IRQ interrupt status bit and the associated status registers in HW_BCH_STATUS0/1 registers.
- If an error is present, then the BCH block corrects the necessary data block or parity block bytes, if possible (not all errors are correctable).

As the BCH decoder reads the data and parity blocks, it records a special condition, i.e., that all of the bits of a payload data block or metadata block are one, including any associated parity bytes. The all-ones case for both parity and data indicates an erased block in the NAND device.

The HW_BCH_STATUS0 register contains a 4-bit field that indicates the final status of the auxiliary block. A value of 0x0 indicates no errors found for a block.

- A value of 1 to 20 inclusive indicates that many correctable errors were found and fixed.
- A value of 0xFE indicates uncorrectable errors detected on the block.
- A value of 0xFF indicates that the block was in the special ALL ONES state and is therefore considered to be an ERASED block.
- All other values are disallowed by the hardware design.

Recall that up to four NAND devices can have DMA chains in-flight at once, i.e. they can all be contending for access to the GPMI data bus. It is impossible to predict which NAND device will enter the BCH engine with a transfer first, because each chain includes a wait4ready command structure. As a result, firmware should look at the HW_BCH_STATUS0_COMPLETED_CE bit field to determine which block is being reported in the status register. There is also a 16-bit HANDLE field in the HW_GPMI_ECCCTRL register that is passed down the pipeline with each transaction. This handle field can be used to speed firmware's detection of which transaction is being reported.

These examples of reading and writing have focused on full page transfers of 4K page NAND devices. Other device configurations can be specified by changing the ECCOUNT field in the GPMI registers and reprogramming the BCH's HW_FLASHnLAYOUTm registers.

The BCH and GPMI blocks are designed to be very efficient at reading single 512-byte pages in one transaction. With no errors, the transaction takes less than 20 HCLKs longer than the time to read the raw data from the NAND.

To summarize, the APBH DMA command chain for a BCH decode operation is shown in Figure 16-10. Seven DMA command structures must be present for each NAND read transaction decoded by the BCH. The seven DMA command structures for multiple NAND read transaction blocks can be chained together to make larger units of work for the BCH, and each will produce an appropriate error report in the BCH PIO space. Multiple NAND devices can have such multiple chains scheduled. The results can come back out of order with respect to the multiple chains.

16.4.3 Interrupts

There are two interrupt sources used in processing BCH protected NAND read and write transfers. Since all BCH operations are initiated by GPMI DMA command structures, the DMA completion interrupt for the GPMI is an important ISR. Both of the flow charts of Figure 16-6 and Figure 16-9 show the GPMI DMA complete ISR skeleton. In both reads and writes, the GPMI DMA completion interrupt is used to schedule work ***INTO*** the error correction pipeline. As the front end processing completes, the DMA interrupt is generated and additional work, such as DMA chains, are passed to the GPMI DMA to keep it *fed*. For write operations, this is the only interrupt that is generated for processing the NAND write transfer.

For reads, however, two interrupts are needed. Every read is started by a GPMI DMA command chain and the front end queue is fed as described above. The back end of the read pipeline is drained by monitoring the BCH completion interrupt found int HW_BCH_CTRL_COMPLETE_IRQ.

An BCH transaction consists of reading or writing all of the blocks requested in the HW_GPMI_ECCCTRL_BUFFER_MASK bit field. As every read transaction completes, it posts the status of all of the blocks to the HW_BCH_STATUS0 and HW_BCH_STATUS1 registers and sets the completion interrupt. The five stages of the BCH read pipeline completes, one in the GPMI and four in the BCH, are independently stalled as they complete and try to deliver to the next stage in the data flow. Several of these stages can be skipped if no-errors are found or once an uncorrectable error is found in a block.

In any case, the final stage will stall if the status register is busy waiting for the CPU to take status register results. The hardware monitors the state of the HW_BCH_CTRL_COMPLETE_IRQ bit. If it is still set when the last pipeline stage is ready to post data, then the stage will stall. It follows that the next previous stage will stall when it is ready to hand off work to the final stage, and so on up the pipeline.

CAUTION

It is important that firmware read the STATUS0/1 results and save them before clearing the interrupt request bit. Otherwise, a transaction and its results could be completely lost.

16.5 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

16.6 Programmable Registers

BCH Hardware Register Format Summary

HW_BCH memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_A000	Hardware BCH ECC Accelerator Control Register (HW_BCH_CTRL)	32	R/W	00E0_0000_0000h	16.6.1/1288
8000_A010	Hardware ECC Accelerator Status Register 0 (HW_BCH_STATUS0)	32	R	0000_0010h	16.6.2/1290
8000_A020	Hardware ECC Accelerator Mode Register (HW_BCH_MODE)	32	R/W	0000_0000h	16.6.3/1292
8000_A030	Hardware BCH ECC Loopback Encode Buffer Register (HW_BCH_ENCODEPTR)	32	R/W	0000_0000h	16.6.4/1293
8000_A040	Hardware BCH ECC Loopback Data Buffer Register (HW_BCH_DATAPTR)	32	R/W	0000_0000h	16.6.5/1293
8000_A050	Hardware BCH ECC Loopback Metadata Buffer Register (HW_BCH_METAPTR)	32	R/W	0000_0000h	16.6.6/1294
8000_A070	Hardware ECC Accelerator Layout Select Register (HW_BCH_LAYOUTSELECT)	32	R/W	E4E4_E4E4h	16.6.7/1294

HW_BCH memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_A080	Hardware BCH ECC Flash 0 Layout 0 Register (HW_BCH_FLASH0LAYOUT0)	32	R/W	070A_8200h	16.6.8/1296
8000_A090	Hardware BCH ECC Flash 0 Layout 1 Register (HW_BCH_FLASH0LAYOUT1)	32	R/W	10DA_8200h	16.6.9/1297
8000_A0A0	Hardware BCH ECC Flash 1 Layout 0 Register (HW_BCH_FLASH1LAYOUT0)	32	R/W	070A_8200h	16.6.10/1298
8000_A0B0	Hardware BCH ECC Flash 1 Layout 1 Register (HW_BCH_FLASH1LAYOUT1)	32	R/W	10DA_8200h	16.6.11/1299
8000_A0C0	Hardware BCH ECC Flash 2 Layout 0 Register (HW_BCH_FLASH2LAYOUT0)	32	R/W	070A_8200h	16.6.12/1300
8000_A0D0	Hardware BCH ECC Flash 2 Layout 1 Register (HW_BCH_FLASH2LAYOUT1)	32	R/W	10DA_8200h	16.6.13/1302
8000_A0E0	Hardware BCH ECC Flash 3 Layout 0 Register (HW_BCH_FLASH3LAYOUT0)	32	R/W	070A_8200h	16.6.14/1303
8000_A0F0	Hardware BCH ECC Flash 3 Layout 1 Register (HW_BCH_FLASH3LAYOUT1)	32	R/W	10DA_8200h	16.6.15/1304
8000_A100	Hardware BCH ECC Debug Register0 (HW_BCH_DEBUG0)	32	R/W	0000_0000h	16.6.16/1305
8000_A110	KES Debug Read Register (HW_BCH_DBGKESREAD)	32	R	0000_0000h	16.6.17/1307
8000_A120	Chien Search Debug Read Register (HW_BCH_DBGCSFEREAD)	32	R	0000_0000h	16.6.18/1308
8000_A130	Syndrome Generator Debug Read Register (HW_BCH_DBGSYNDGENREAD)	32	R	0000_0000h	16.6.19/1308
8000_A140	Bus Master and ECC Controller Debug Read Register (HW_BCH_DBGAHBMREAD)	32	R	0000_0000h	16.6.20/1309
8000_A150	Block Name Register (HW_BCH_BLOCKNAME)	32	R	2048_4342h	16.6.21/1309
8000_A160	BCH Version Register (HW_BCH_VERSION)	32	R	0100_0000h	16.6.22/1310

16.6.1 Hardware BCH ECC Accelerator Control Register (HW_BCH_CTRL)

The BCH CTRL provides overall control of the hardware ECC accelerator

HW_BCH_CTRL: 0x000

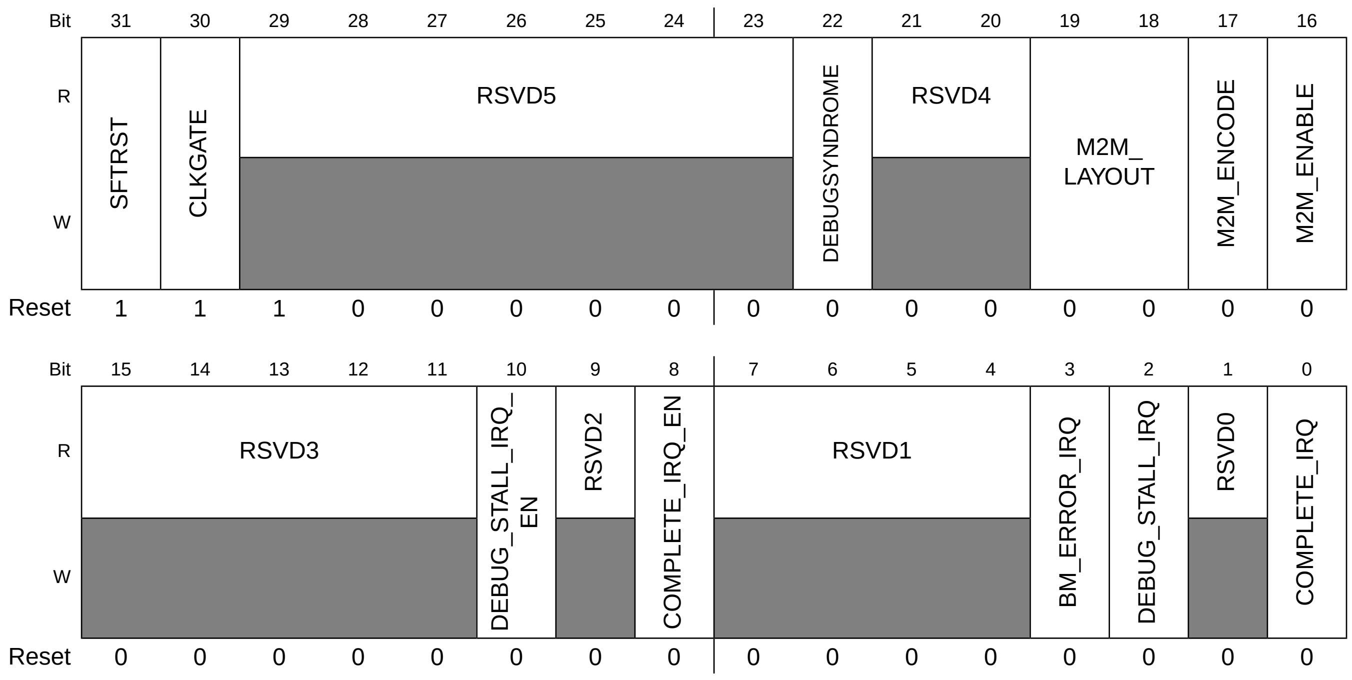
HW_BCH_CTRL_SET: 0x004

HW_BCH_CTRL_CLR: 0x008

HW_BCH_CTRL_TOG: 0x00C

EXAMPLE

Address: HW_BCH_CTRL – 8000_A000h base + 0h offset = 8000_A000h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
SFTRST
CLKGATE
RSVD5
DEBUGSYNDROME
RSVD4
M2M_ LAYOUT
M2M_ENCODE
M2M_ENABLE
Reset 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSVD3
DEBUG_STALL_IRQ_ EN
RSVD2
COMPLETE_IRQ_EN
RSVD1
BM_ERROR_IRQ
DEBUG_STALL_IRQ
RSVD0
COMPLETE_IRQ
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_BCH_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal BCH operation. Set this bit to one (default) to disable clocking with the BCH and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the BCH block to its default state. This bit resets all state machines except for the AHB master state machine 0x0 **RUN** — Allow BCH to operate normally. 0x1 **RESET** — Hold BCH in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow BCH to operate normally. 0x1 **NO_CLKS** — Do not clock BCH gates in order to minimize power consumption.
29–23 RSVD5	Reserved, always set this bit to zero.
22 DEBUGSYNDROME	(For debug purposes only). Enable write of computed syndromes to memory on BCH decode operations. Computed syndromes will be written to the auxiliary buffer after the status block. Syndromes will be written as padded 16-bit values.
21–20 RSVD4	Reserved, always set these bits to zero.
19–18 M2M_LAYOUT	Selects the flash page format for memory-to-memory operations.
17 M2M_ENCODE	Selects encode (parity generation) or decode (correction) mode for memory-to-memory operations.
16 M2M_ENABLE	NOTE! WRITING THIS BIT INITIATES A MEMORY-TO-MEMORY OPERATION. The BCH module must be inactive (not processing data from the GPMI) when this bit is set. The M2M_ENCODE and M2M_LAYOUT bits as well as the ENCODEPTR, DATAPTR, and METAPTR registers are used for memory-to-memory operations and must be correctly programmed before writing this bit.

HW_BCH_CTRL field descriptions (continued)

Field	Description
15–11 RSVD3	Reserved, always set these bits to zero.
10 DEBUG_STALL_ IRQ_EN	1 = interrupt on debug stall mode is enabled. The irq is raised on every block
9 RSVD2	Reserved, always set these bits to zero.
8 COMPLETE_IRQ_EN	1 = interrupt on completion of correction is enabled.
7–4 RSVD1	Reserved, always set these bits to zero.
3 BM_ERROR_IRQ	AHB Bus interface Error Interrupt Status. Write a one to the SCT clear address to clear the interrupt status bit.
2 DEBUG_STALL_IRQ	DEBUG STALL Interrupt Status. Write a one to the SCT clear address to clear the interrupt status bit.
1 RSVD0	Reserved, always set these bits to zero.
0 COMPLETE_IRQ	This bit indicates the state of the external interrupt line. Write a one to the SCT clear address to clear the interrupt status bit. NOTE: subsequent ECC completions will be held off as long as this bit is set. Be sure to read the data from HW_BCH_STATUS0,1 before clearing this interrupt bit.

16.6.2 Hardware ECC Accelerator Status Register 0 (HW_BCH_STATUS0)

The BCH STAT register provides visibility into the run-time status of the BCH and status information when processing is complete.

EXAMPLE

Address: HW_BCH_STATUS0 – 8000_A000h base + 10h offset = 8000_A010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HANDLE												COMPLETED_CE			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STATUS_BLK0								RSVD1			ALLONES	CORRECTED	UNCORRECTABLE	RSVD0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

HW_BCH_STATUS0 field descriptions

Field	Description
31–20 HANDLE	Software supplies a 12 bit handle for this transfer as part of the GPMI DMA PIO operation that started the transaction. That handle passes down the pipeline and ends up here at the time the BCH interrupt is signaled.
19–16 COMPLETED_CE	This is the chip enable number corresponding to the NAND device from which this data came.
15–8 STATUS_BLK0	Count of symbols in error during processing of first block of flash (metadata block). The number of errors reported will be in the range of 0 to the ECC correction level for block 0. 0x00 **ZERO** — No errors found on block. 0x01 **ERROR1** — One error found on block. 0x02 **ERROR2** — One errors found on block. 0x03 **ERROR3** — One errors found on block. 0x04 **ERROR4** — One errors found on block. 0xFE **UNCORRECTABLE** — Block exhibited uncorrectable errors. 0xFF **ERASED** — Page is erased.

HW_BCH_STATUS0 field descriptions (continued)

Field	Description
7–5 RSVD1	Reserved, always set these bits to zero.
4 ALLONES	1 = All data bits of this transaction are ONE.
3 CORRECTED	1 = At least one correctable error encountered during last processing cycle.
2 UNCORRECTABLE	1 = Uncorrectable error encountered during last processing cycle.
1–0 RSVD0	Reserved, always set these bits to zero.

16.6.3 Hardware ECC Accelerator Mode Register (HW_BCH_MODE)

The BCH MODE register provides additional mode controls.

Contains additional global mode controls for the BCH engine.

EXAMPLE

Address: HW_BCH_MODE – 8000_A000h base + 20h offset = 8000_A020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																								ERASE_THRESHOLD							
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_MODE field descriptions

Field	Description
31–8 RSVD	Reserved, always set these bits to zero.
7–0 ERASE_ THRESHOLD	This value indicates the maximum number of zero bits on a flash page for it to be considered erased. For SLC NAND devices, this value should be programmed to 0 (meaning that the entire page should consist of bytes of 0xFF. For MLC NAND devices, bit errors may occur on reads (even on blank pages), so this threshold can be used to tune the erased page checking algorithm.

16.6.4 Hardware BCH ECC Loopback Encode Buffer Register (HW_BCH_ENCODEPTR)

When performing memory to memory operations, indicates the address of the encode buffer. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register.

For memory to memory operations, this register is used as the pointer to the encoded data, which is an output when encoding and an input while decoding.

EXAMPLE

Address: HW_BCH_ENCODEPTR – 8000_A000h base + 30h offset = 8000_A030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_ENCODEPTR field descriptions

Field	Description
31–0 ADDR	Address pointer to encode buffer. This is the source for decode operations and the destination for encode operations. This value must be aligned on a 4 byte boundary.

16.6.5 Hardware BCH ECC Loopback Data Buffer Register (HW_BCH_DATAPTR)

When performing memory to memory operations, indicates the address of the data buffer.

For memory to memory operations, this register is used as the pointer to the data to encode or the destination buffer for decode operations.

EXAMPLE

Address: HW_BCH_DATAPTR – 8000_A000h base + 40h offset = 8000_A040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DATAPTR field descriptions

Field	Description
31–0 ADDR	Address pointer to data buffer. This is the source for encode operations and the destination for decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

16.6.6 Hardware BCH ECC Loopback Metadata Buffer Register (HW_BCH_METAPTR)

When performing memory to memory operations, indicates the address of the metadata buffer.

For memory to memory operations, this register is used as the pointer to the metadata to encode or the extracted metadata for decode operations.

EXAMPLE

Address: HW_BCH_METAPTR – 8000_A000h base + 50h offset = 8000_A050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_METAPTR field descriptions

Field	Description
31–0 ADDR	Address pointer to metadata buffer. This is the source for encode metadata read operations and the destination for metadata decode operations. This register should be programmed before writing a 1 to the M2M_ENABLE bit in the CTRL register. This value must be aligned on a 4 byte boundary.

16.6.7 Hardware ECC Accelerator Layout Select Register (HW_BCH_LAYOUTSELECT)

The BCH LAYOUTSELECT register provides a mapping of chip selects to layout registers.

When the BCH engine receives a request to process a data block from the GPMI interface, it will use this register to map the incoming chip select to one of the four possible flash layout registers

EXAMPLE

Address: HW_BCH_LAYOUTSELECT – 8000_A000h base + 70h offset = 8000_A070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CS15_ SELECT		CS14_ SELECT		CS13_ SELECT		CS12_ SELECT		CS11_ SELECT		CS10_ SELECT		CS9_ SELECT		CS8_ SELECT	
Re- set	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CS7_ SELECT		CS6_ SELECT		CS5_ SELECT		CS4_ SELECT		CS3_ SELECT		CS2_ SELECT		CS1_ SELECT		CS0_ SELECT	
Re- set	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	0

HW_BCH_LAYOUTSELECT field descriptions

Field	Description
31–30 CS15_SELECT	Selects which layout is used for chip select 15.
29–28 CS14_SELECT	Selects which layout is used for chip select 14.
27–26 CS13_SELECT	Selects which layout is used for chip select 13.
25–24 CS12_SELECT	Selects which layout is used for chip select 12.
23–22 CS11_SELECT	Selects which layout is used for chip select 11.
21–20 CS10_SELECT	Selects which layout is used for chip select 10.
19–18 CS9_SELECT	Selects which layout is used for chip select 9.
17–16 CS8_SELECT	Selects which layout is used for chip select 8.
15–14 CS7_SELECT	Selects which layout is used for chip select 7.
13–12 CS6_SELECT	Selects which layout is used for chip select 6.
11–10 CS5_SELECT	Selects which layout is used for chip select 5.
9–8 CS4_SELECT	Selects which layout is used for chip select 4.
7–6 CS3_SELECT	Selects which layout is used for chip select 3.
5–4 CS2_SELECT	Selects which layout is used for chip select 2.
3–2 CS1_SELECT	Selects which layout is used for chip select 1.
1–0 CS0_SELECT	Selects which layout is used for chip select 0.

16.6.8 Hardware BCH ECC Flash 0 Layout 0 Register (HW_BCH_FLASH0LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT1 register to control the format for the devices selecting layout 0 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
HW_BCH_FLASH0LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH0LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH0LAYOUT0 – 8000_A000h base + 80h offset = 8000_A080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NBLOCKS								META_SIZE								ECC0				DATA0_SIZE											
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH0LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–12 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed

HW_BCH_FLASH0LAYOUT0 field descriptions (continued)

Field	Description
	0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATA0_SIZE	Indicates the size of the data 0 block (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. If set to zero, the first block will only contain metadata.

16.6.9 Hardware BCH ECC Flash 0 Layout 1 Register (HW_BCH_FLASH0LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH0LAYOUT0 register to control the format for the device selecting layout 0 in the LAYOUTSELECT register.

EXAMPLE

```
HW_BCH_FLASH0LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH0LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH0LAYOUT1 – 8000_A000h base + 90h offset = 8000_A090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PAGE_SIZE																ECCN				DATAN_SIZE											
Re-set	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH0LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–12 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed

HW_BCH_FLASH0LAYOUT1 field descriptions (continued)

Field	Description
	0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

16.6.10 Hardware BCH ECC Flash 1 Layout 0 Register (HW_BCH_FLASH1LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH1LAYOUT1 register to control the format for the devices selecting layout 1 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
HW_BCH_FLASH1LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH1LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH1LAYOUT0 – 8000_A000h base + A0h offset = 8000_A0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NBLOCKS								META_SIZE								ECC0				DATA0_SIZE											
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH1LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–12 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATA0_SIZE	Indicates the size of the data 0 block (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. If set to zero, the first block will only contain metadata.

16.6.11 Hardware BCH ECC Flash 1 Layout 1 Register (HW_BCH_FLASH1LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH1LAYOUT0 register to control the format for the device selecting layout 1 in the LAYOUTSELECT register.

EXAMPLE

```
HW_BCH_FLASH1LAYOUT0_WR(0x020C8000);
```

```
HW_BCH_FLASH1LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH1LAYOUT1 – 8000_A000h base + B0h offset = 8000_A0B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PAGE_SIZE																ECCN				DATAN_SIZE											
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH1LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–12 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

16.6.12 Hardware BCH ECC Flash 2 Layout 0 Register (HW_BCH_FLASH2LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH2LAYOUT1 register to control the format for the devices selecting layout 2 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
HW_BCH_FLASH2LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH2LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH2LAYOUT0 – 8000_A000h base + C0h offset = 8000_A0C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NBLOCKS								META_SIZE								ECC0				DATA0_SIZE											
Re-set	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH2LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–12 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATA0_SIZE	Indicates the size of the data 0 block (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. If set to zero, the first block will only contain metadata.

16.6.13 Hardware BCH ECC Flash 2 Layout 1 Register (HW_BCH_FLASH2LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH2LAYOUT0 register to control the format for the device selecting layout 2 in the LAYOUTSELECT register.

EXAMPLE

```
HW_BCH_FLASH2LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH2LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH2LAYOUT1 – 8000_A000h base + D0h offset = 8000_A0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PAGE_SIZE																ECCN				DATAN_SIZE											
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH2LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–12 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed 0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

16.6.14 Hardware BCH ECC Flash 3 Layout 0 Register (HW_BCH_FLASH3LAYOUT0)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH3LAYOUT1 register to control the format for the devices selecting layout 3 in the LAYOUTSELECT register.

Each pair of layout registers describes one of four supported flash configurations. Software should program the LAYOUTSELECT register for each supported GPMI chip select to select from one of the fourlayout values. Each pair of registers contains settings that are used by the BCH block while reading/writing the flash page to control data, metadata, and flash page sizes as well as the ECC correction level. The first block written to flash can be programmed to have different ECC, metadata, and data sizes from subsequent data blocks on the device. In addition, the number of blocks stored on a page of flash is not fixed, but instead is determined by the number of bytes consumed by the initial (block 0) and subsequent data blocks. See the BCH programming reference manual for more information on setting up the flash layout registers.

EXAMPLE

```
HW_BCH_FLASH3LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH3LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH3LAYOUT0 – 8000_A000h base + E0h offset = 8000_A0E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NBLOCKS								META_SIZE								ECC0				DATA0_SIZE											
Reset	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH3LAYOUT0 field descriptions

Field	Description
31–24 NBLOCKS	Number of subsequent blocks on the flash page (excluding the data0 block). A value of 0 indicates that only the DATA0 block is present and a value of 8 indicates that 8 subsequent blocks are present for a total of 9 blocks on the flash (including the DATA0 block). Any values from 0 to 255 are supported by the hardware.
23–16 META_SIZE	Indicates the size of the metadata (in bytes) to be stored on a flash page. The BCH design support from 0 to 255 bytes for metadata -- if set to zero, no metadata will be stored. Metadata is stored before the associated data in block 0. If the DATA0_SIZE field is programmed to a zero, then metadata effectively be stored with its own parity. When both the metadata and data0 fields are programmed with non-zero values, the first block will contain both portions of data and will be covered by a single parity block.
15–12 ECC0	Indicates the ECC level for the first block on the flash page. The first block covers metadata plus the associated data from the DATA0_SIZE field. 0x0 **NONE** — No ECC to be performed

HW_BCH_FLASH3LAYOUT0 field descriptions (continued)

Field	Description
	0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATA0_SIZE	Indicates the size of the data 0 block (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. If set to zero, the first block will only contain metadata.

16.6.15 Hardware BCH ECC Flash 3 Layout 1 Register (HW_BCH_FLASH3LAYOUT1)

The flash format register contains a description of the logical layout of data on the flash device. This register is used in conjuction with the FLASH3LAYOUT0 register to control the format for the device selecting layout 3 in the LAYOUTSELECT register.

EXAMPLE

```
HW_BCH_FLASH3LAYOUT0_WR(0x020C8000);
HW_BCH_FLASH3LAYOUT1_WR(0x04408200);
```

Address: HW_BCH_FLASH3LAYOUT1 – 8000_A000h base + F0h offset = 8000_A0F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PAGE_SIZE																ECCN				DATAN_SIZE											
Reset	0	0	0	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

HW_BCH_FLASH3LAYOUT1 field descriptions

Field	Description
31–16 PAGE_SIZE	Indicates the total size of the flash page (in bytes). This should be set to the page size including spare area. The page size is programmable to accomodate different flash configurations that may be available in the future.
15–12 ECCN	Indicates the ECC level for the subsequent blocks on the flash page (blocks 1-n). Subsequent blocks only contain data (no metadata). 0x0 **NONE** — No ECC to be performed

HW_BCH_FLASH3LAYOUT1 field descriptions (continued)

Field	Description
	0x1 **ECC2** — ECC 2 to be performed 0x2 **ECC4** — ECC 4 to be performed 0x3 **ECC6** — ECC 6 to be performed 0x4 **ECC8** — ECC 8 to be performed 0x5 **ECC10** — ECC 10 to be performed 0x6 **ECC12** — ECC 12 to be performed 0x7 **ECC14** — ECC 14 to be performed 0x8 **ECC16** — ECC 16 to be performed 0x9 **ECC18** — ECC 18 to be performed 0xA **ECC20** — ECC 20 to be performed
11–0 DATAN_SIZE	Indicates the size of the subsequent data blocks (in bytes) to be stored on the flash page. The data size MUST be a multiple of four bytes. The size of subsequent data blocks does not have to match the data size for block 0, which is important when metadata is stored separately or for balancing the amount of data stored in each block.

16.6.16 Hardware BCH ECC Debug Register0 (HW_BCH_DEBUG0)

The hardware BCH accelerator internal state machines and signals can be seen in the ECC debug register.

HW_BCH_DEBUG0: 0x100

HW_BCH_DEBUG0_SET: 0x104

HW_BCH_DEBUG0_CLR: 0x108

HW_BCH_DEBUG0_TOG: 0x10C

The HW_BCH_DEBUG0 register provides access to various internal state information which might prove useful during hardware debug and validation.

EXAMPLE

```
// perform BIST operation
  HW_BCH_DEBUG0_SET(BM_BCH_DEBUG0_ROM_BIST_ENABLE);  // enable BIST operation

  // poll until BIST_DONE
  while( (HW_BCH_DEBUG0_RD() & BM_BCH_DEBUG0_ROM_BIST_COMPLETE) == 0 );

  i=HW_BCH_DBGKESREAD();
  if(HW_BCH_DBGKESREAD_RD() != 0x7AA3792F) {
    // BIST FAILED
    err++;
  }
HW_BCH_DEBUG0_CLR(BM_BCH_DEBUG0_ROM_BIST_ENABLE | BM_BCH_DEBUG0_ROM_BIST_COMPLETE);  // clear
 bist status
```

Address: HW_BCH_DEBUG0 – 8000_A000h base + 100h offset = 8000_A100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1					ROM_BIST_ ENABLE	ROM_BIST_ COMPLETE	KES_DEBUG_SYNDROME_SYMBOL								
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	KES_DEBUG_ SHIFT_SYND	KES_DEBUG_ PAYLOAD_FLAG	KES_DEBUG_ MODE4K	KES_DEBUG_ KICK	KES_STANDALONE	KES_DEBUG_ STEP	KES_DEBUG_ STALL	BM_KES_TEST_ BYPASS	RSVD0		DEBUG_REG_SELECT					
W									▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DEBUG0 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set these bits to zero.
26 ROM_BIST_ ENABLE	Software may initiate a ROM BIST operation by toggling this bit from a zero to a one. When the operation is complete, the ROM_BIST_COMPLETE bit will be set and the ROM's CRC value will be available in the DEBUG data register.
25 ROM_BIST_ COMPLETE	This bit will be set after a BIST operation completes, at which time the ROM CRC is available in the DBGKESREAD register. The CRC value will be cleared after the BIST_ENABLE bit is cleared.
24–16 KES_DEBUG_ SYNDROME_ SYMBOL	The 9 bit value in this bit field will be shifted into the syndrome register array at the input of the KES engine whenever HW_BCH_DEBUG0_KES_DEBUG_SHIFT_SYND is toggled. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
15 KES_DEBUG_ SHIFT_SYND	Toggling this bit causes the value in HW_BCH_DEBUG0_KES_SYNDROME_SYMBOL to be shift into the syndrome register array at the input to the KES engine. After shifting in 16 symbols, one can kick off both KES and CF cycles by toggling HW_BCH_DEBUG0_KES_DEBUG_KICK. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
14 KES_DEBUG_ PAYLOAD_FLAG	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input payload flag. 0x1 **DATA** — Payload is set for 512 byte data block. 0x1 **AUX** — Payload is set for 65 or 19 byte auxilliary block.
13 KES_DEBUG_ MODE4K	When running the stand alone debug mode on the error calculator, the state of this bit is presented to the KES engine as the input mode (4K or 2K pages). 0x1 **4k** — Mode is set for 4K NAND pages. 0x1 **2k** — Mode is set for 2K NAND pages.

HW_BCH_DEBUG0 field descriptions (continued)

Field	Description
12 KES_DEBUG_ KICK	Toggling causes KES engine FSM to start as if kick by the Bus Master. This allows stand alone testing of the KES and Chien Search engines. Be sure to set KES_BCH_DEBUG0_KES_STANDALONE mode to 1 before kicking.
11 KES_ STANDALONE	Set to one to cause the KES engine to suppress toggling the KES_BM_DONE signal to the bus master and to suppress toggling the CF_BM_DONE signal by the CF engine. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
10 KES_DEBUG_ STEP	Toggling this bit causes the KES FSM to skip passed the stall state if it is in DEBUG_STALL mode and it has completed processing a block.
9 KES_DEBUG_ STALL	Set to one to cause KES FSM to stall after notifying Chien search engine to start processing its block but before notifying the bus master that the KES computation is complete. This allows a diagnostic to stall the FSM after each blocks key equations are solved. This also has the effect of stalling the CSFE search engine so it's state can be examined after it finishes processing the KES stalled block. 0x0 **NORMAL** — KES FSM proceeds to next block supplied by bus master. 0x1 **WAIT** — KES FSM waits after current equations are solved and the search engine is started.
8 BM_KES_TEST_ BYPASS	1 = Point all synd_gen writes to dummy area at the end of the AUXILLIARY block so that diagnostics can preload all payload, parity bytes and computed syndrome bytes for test the KES engine. 0x0 **NORMAL** — Bus master address generator for synd_gen writes operates normally. 0x1 **TEST_MODE** — Bus master address generator always addresses last four bytes in Auxilliary block.
7–6 RSVD0	Reserved, always set these bits to zero.
5–0 DEBUG_REG_ SELECT	The value loaded in this bit field is used to select the internal register state view of KES engine or the Chien search engine.

16.6.17 KES Debug Read Register (HW_BCH_DBGKESREAD)

The hardware BCH ECC accelerator key equation solver internal state machines and signals can be seen in the ECC debug registers.

EXAMPLE

Address: HW_BCH_DBGKESREAD – 8000_A000h base + 110h offset = 8000_A110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUES																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DBGKESREAD field descriptions

Field	Description
31–0 VALUES	This register will return the ROM BIST CRC value after a BIST test.

16.6.18 Chien Search Debug Read Register (HW_BCH_DBGCSFEREAD)

The hardware BCH ECC accelerator Chien Search internal state machines and signals can be seen in the ECC debug registers.

EXAMPLE

Address: HW_BCH_DBGCSFEREAD – 8000_A000h base + 120h offset = 8000_A120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUES																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DBGCSFEREAD field descriptions

Field	Description
31–0 VALUES	Reserved

16.6.19 Syndrome Generator Debug Read Register (HW_BCH_DBGSYNDGENREAD)

The hardware BCH ECC accelerator syndrome generator internal state machines and signals can be seen in the ECC debug registers.

EXAMPLE

Address: HW_BCH_DBGSYNDGENREAD – 8000_A000h base + 130h offset = 8000_A130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUES																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DBGSYNDGENREAD field descriptions

Field	Description
31–0 VALUES	Reserved

16.6.20 Bus Master and ECC Controller Debug Read Register (HW_BCH_DBGAHBMREAD)

The hardware BCH ECC accelerator bus master and ecc controller internal state machines and signals can be seen in the ECC debug registers.

EXAMPLE

Address: HW_BCH_DBGAHBMREAD – 8000_A000h base + 140h offset = 8000_A140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUES																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_DBGAHBMREAD field descriptions

Field	Description
31–0 VALUES	Reserved

16.6.21 Block Name Register (HW_BCH_BLOCKNAME)

Read only view of the block name string BCH.

Fixed pattern read only value for test puposes. Can be read as an ASCII string with the zero termination coming from the first byte of the BLOCKVERSION register.

EXAMPLE

```
char *cp = ((char *)HW_BCH_BLOCKNAME_ADDR);   reads back "BCH ".
```

Address: HW_BCH_BLOCKNAME – 8000_A000h base + 150h offset = 8000_A150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NAME																															
W																																
Reset	0	0	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	1	1	0	1	0	0	0	0	1	0

HW_BCH_BLOCKNAME field descriptions

Field	Description
31–0 NAME	Should be the ASCII characters BCH (0x20, H, C, B).

16.6.22 BCH Version Register (HW_BCH_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_BCH_VERSION.B.MAJOR != 1) Error();
```

Address: HW_BCH_VERSION – 8000_A000h base + 160h offset = 8000_A160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_BCH_VERSION field descriptions

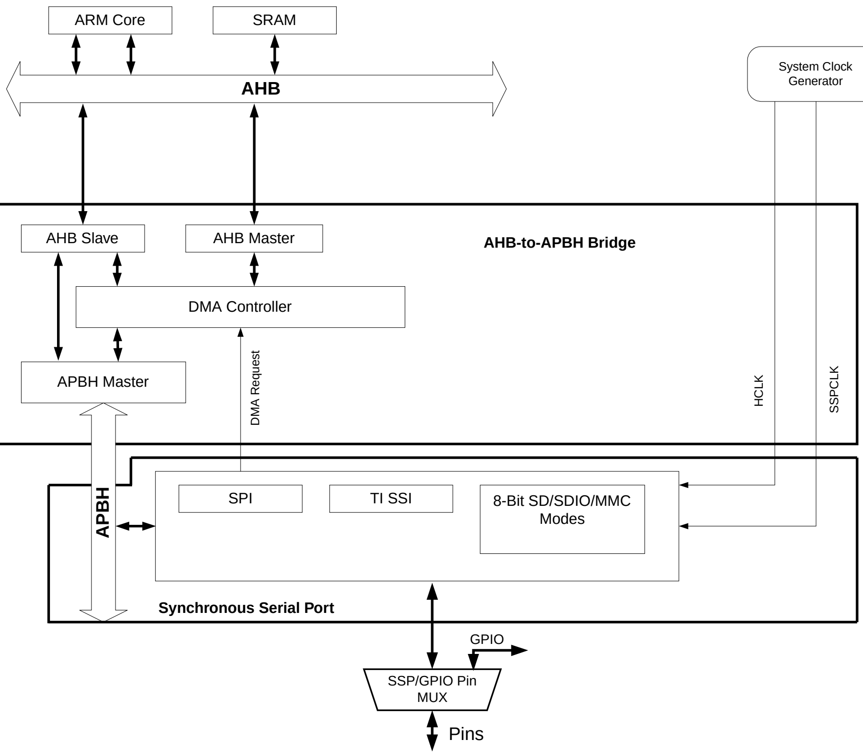
Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 17
Synchronous Serial Ports (SSP)

17.1 Overview

The Synchronous Serial Port is a host controller which provides a flexible interface for inter-IC and removable media control and communication. The SSP supports MMC/SD/SDIO 1-bit, 4-bit, and 8-bit modes; SPI master and slave modes; and Texas Instruments SSI mode. The SSP is fully compliant with the eMMC 4.4 standard. Specifically, it supports Power-on and alternate boot mode and is designed to perform both SDR and DDR operations at the maxinum speed of 52 MHz. In SPI mode, the SSP has enhancements to support 1-bit legacy MMC cards, and SPI master dual (2-bit) and quad (4-bit) read modes are also supported. The SSP has a dedicated DMA channel in the bridge and can also be controlled directly by the CPU through PIO registers.Figure 17-1 illustrates one of the four SSP ports included on the i.MX28. Each SSP is independent of the other and can have separate SSPCLK frequencies.

Figure 17-1. Synchronous Serial Port Block Diagram


ARM Core
SRAM
AHB
System Clock Generator
AHB Slave
AHB Master
AHB-to-APBH Bridge
DMA Controller
APBH Master
DMA Request
HCLK
SSPCLK
APBH
SPI
TI SSI
8-Bit SD/SDIO/MMC Modes
Synchronous Serial Port
GPIO
SSP/GPIO Pin MUX
Pins

17.2 External Pins

#id-54331/id-33763 lists the SSP pin placements for all supported modes.

Table 17-1. SSP Pin Matrix

Pin Name	MOTOROLASPI Mode	WinBONDSPI Mode	TI SSIMode	SD/SDIO/MMC/ Modes
SSP_SCK	SCK	CLK	CLK	CLK
SSP_CMD	MOSI	DI (IO0)	MOSI	CMD
SSP_DATA0	MISO	DO (IO1)	MISO	DATA0
SSP_DATA1		WPn (IO2)		DATA1/IRQ
SSP_DATA2		HOLDn (IO3)		DATA2
SSP_DATA3	SSn0	SSn0	SSn	DATA3
SSP_DATA4	SSn1	SSn1		DATA4
SSP_DATA5	SSn2	SSn2		DATA5
SSP_DATA6				DATA6
SSP_DATA7				DATA7
SSP_DETECT				CARD_DETECT

The pin control interface on the i.MX28 provides all digital pins with selectable output drive strengths. In addition, all SSP data pins have selectable 47-KΩ pullup resistors, and SSP command pins have 10-KΩ pullups. Configuring the SSP_CMD pad to connect to the internal 10-KΩ pullup is recommended for SD/SDIO/MMC modes during the Card_ID phase. After the Card_ID phase, the 10-KΩ pullup should be disabled, and the weaker external 47-KΩ pullup takes over. The SSP_DATA pads also can be configured to connect to an internal 47-KΩ pullup, which is required for SD/SDIO/MMC modes.

17.3 Bit Rate Generation

The serial bit rate is derived by dividing down the internal clock SSPCLK. The clock is first divided by an even prescale value, CLOCK_DIVIDE, from 2 to 254, which is programmed in HW_SSP_TIMING. The clock is further divided by a value from 1 to 256, which is 1 + CLOCK_RATE, where CLOCK_RATE is the value programmed in HW_SSP_TIMING.

The frequency of the output signal bit clock SSP_SCK is defined as follows:

$$SSP\_SCK = \frac{SSPCLK}{CLOCK\ DIVIDE \times (1 + CLOCK\_RATE)}$$

For example, if SSPCLK is 3.6864 MHz, and CLOCK_DIVIDE = 2, then SSP_SCK has a frequency range from 7.2 KHz to 1.8432 MHz. See Clock Generation and Control (CLKCTRL) Overview for more clock details.

17.4 Frame Format for SPI and SSI

Each data frame is between 4 and 16 bits long, depending on the size of data programmed, and is transmitted starting with the MSB. Two basic frame types can be selected:

- Motorola SPI
- Texas Instruments Synchronous Serial Interface (SSI)

For both formats, the serial clock (SSP_SCK) is held inactive while the SSP is idle and transitions at the programmed frequency only during active transmissions or reception of data. The idle state of SSP_SCK is used to provide a receive time-out indication, which occurs when FIFO still contains data after a time-out period.

For Motorola SPI frame format, the serial frame (SSn) pin is active low and is asserted (pulled down) during the entire transmission of the frame.

For Texas Instruments synchronous serial interface (SSI) frame format, the SSn pin is pulsed for one serial clock period starting at its rising edge, prior to the transmission of each frame. For this frame format, both the SSP and the off-chip slave device drive their output on data on the rising edge of SSP_SCK, and latch data from the other device on the falling edge.

The SSP master supports up to three combinations of SPI and SSI slave devices connected. Three SSn pins are provided but only one can be active at a time.

17.5 Motorola SPI Mode

The SPI mode is used for general inter-component communication and legacy 1-bit MMC cards.

17.5.1 SPI DMA Mode

The SPI bus is inherently a full-duplex bidirectional interface. However, as most applications only require half-duplex data transmission, the i.MX28 has a single DMA channel for the SSP. It can be configured for either transmit or receive. In DMA receive mode, the SPI continuously repeats the word written to its data register. In DMA transmit mode, the SPI ignores the incoming data.

17.5.2 Motorola SPI Frame Format

The Motorola SPI interface is a four-wire interface where the SSn signal behaves as a slave select. The main feature of the Motorola SPI format is that the inactive state and phase of SSP_SCK signal are programmable through the polarity and phase bits within the HW_SSP_CTRL1.

17.5.2.1 Clock Polarity

- When the clock polarity control bit is low, it produces a steady-state low value on the SSP_SCK pin.
- When the clock polarity control bit is high, a steady-state high value is placed on the SSP_SCK pin when data is not being transferred.

17.5.2.2 Clock Phase

The phase control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted, by either allowing or not allowing a clock transition before the first data-capture edge.

- When the phase control bit is low, data is captured on the first clock-edge transition.
- When the clock phase control bit is high, data is captured on the second clock-edge transition.

17.5.3 Motorola SPI Format with Polarity=0, Phase=0

Single and continuous transmission signal sequences for Motorola SPI format with POLARITY=0, PHASE=0 are shown in Figure 17-2 and Figure 17-3.

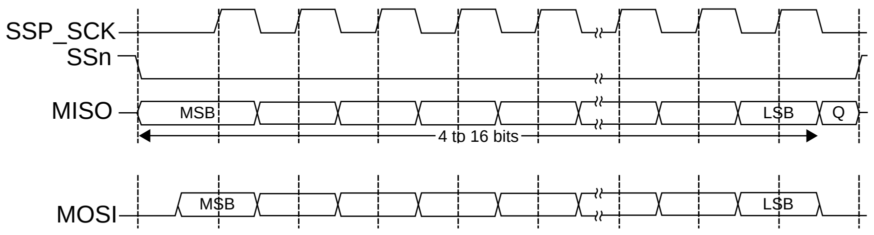


Figure 17-2. Motorola SPI Frame Format (Single Transfer) with POLARITY=0 and PHASE=0

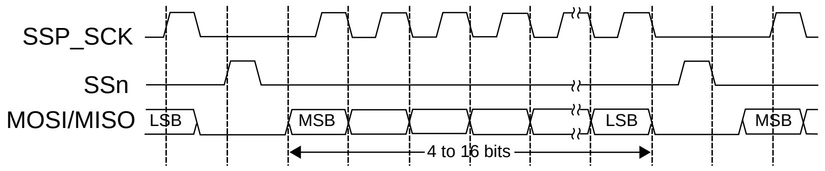


Figure 17-3. Motorola SPI Frame Format with POLARITY=0 and PHASE=0

In this configuration, during idle periods:

- The SSP_SCK signal is forced low.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, SSP_SCK is an output.
- When the SSP is configured as a slave, SSP_SCK is an input.

If the SSP is enabled and there is valid data within the FIFO, the start of the transmission is signified by the SSn master signal being low. This causes slave data to be enabled onto the MISO input line of the master, and enables the master MOSI output pad.

One-half SSP_SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SSP_SCK master clock pin goes high after one further half SSP_SCK period.

The data is now captured on the rising and propagated on the falling edges of the SSP_SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSn signal must be pulsed high between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the PHASE bit is logic 0. Therefore, the master device must raise the SSn pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSn pin is returned to its idle state one SSP_SCK period after the last bit has been captured.

17.5.4 Motorola SPI Format with Polarity=0, Phase=1

The transfer signal sequence for Motorola SPI format with POLARITY=0 and PHASE=1 is shown in Figure 17-4, which covers both single and continuous transfers.

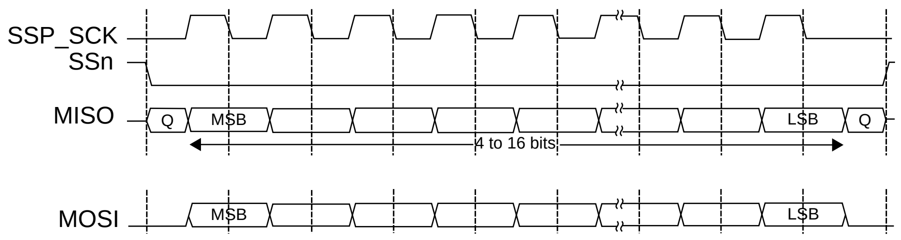


Figure 17-4. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=0 and PHASE=1

In this configuration, during idle periods:

- The SSP_SCK signal is forced low.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the FIFO, the start of the transmission is signified by the SSn master signal being low. After a further one-half SSP_SCK period, both master and slave valid data are enabled with a rising-edge transition.

Data is then captured on the falling edges and propagated on the rising edges of the SSP_SCK signal.

In the case of a single word transfer, after all bits have been transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

For continuous back-to-back transfers, SSPFSOUT (the SSn pin in master mode) is held low between successive data words and termination is the same as that of a single word transfer.

17.5.5 Motorola SPI Format with Polarity=1, Phase=0

Single and continuous transmission signal sequences for Motorola SPI format with POLARITY=1 and PHASE=0 are shown in Figure 17-5 and Figure 17-6.

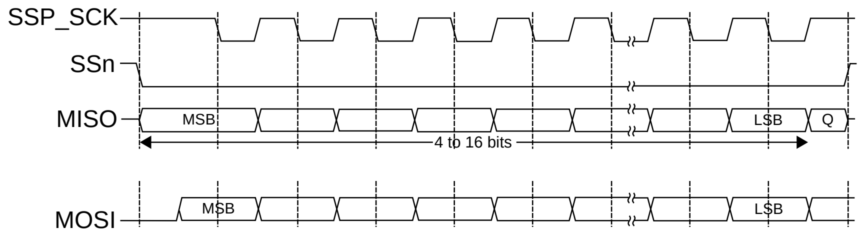


Figure 17-5. Motorola SPI Frame Format (Single Transfer) with POLARITY=1 and PHASE=0

Note

In Figure 17-5, Q is an undefined signal.

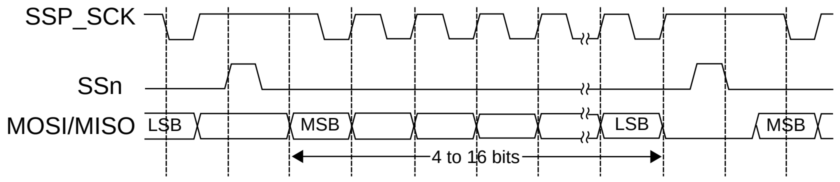


Figure 17-6. Motorola SPI Frame Format (Continuous Transfer) with POLARITY=1 and PHASE=0

In this configuration, during idle periods:

- The SSP_SCK signal is forced high.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the FIFO, the start of transmission is signified by the SSn master signal being driven low, which causes slave data to be immediately transferred onto the MISO line of the master, and enabling the master MOSI output pad.

One half-period later, valid master data is transferred to the MOSI line. Now that both master and slave data have been set, the SSP_SCK master clock pin becomes low after one further half SSP_SCK period. This means that data is captured on the falling edges and propagated on the rising edges of the SSP_SCK signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSn signal must be pulsed high between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the PHASE bit is logic 0. Therefore, the master device must raise SSPSFSSIN (the SSn pin in slave mode) of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSn pin is returned to its idle state one SSP_SCK period after the last bit has been captured.

17.5.6 Motorola SPI Format with Polarity=1, Phase=1

The transfer signal sequence for Motorola SPI format with POLARITY=1 and PHASE=1 is shown in Figure 17-7, which covers both single and continuous transfers.

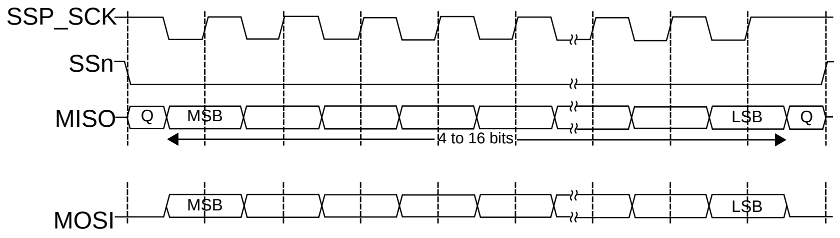


Figure 17-7. Motorola SPI Frame Format with POLARITY=1 and PHASE=1

Note

In Figure 17-7, Q is an undefined signal.

In this configuration, during idle periods:

- The SSP_SCK signal is forced high.
- SSn is forced high.
- The Transmit data line MOSI is arbitrarily forced low.
- When the SSP is configured as a master, the SSP_SCK pad is an output.
- When the SSP is configured as a slave, the SSP_SCK is an input.

If the SSP is enabled and there is valid data within the FIFO, the start of transmission is signified by the SSn master signal being driven low, and MOSI output is enabled. After a further one-half SSP_SCK period, both master and slave are enabled onto their respective transmission lines. At the same time, the SSP_SCK is enabled with a falling edge transition. Data is then captured on the rising edge and propagated on the falling edges of the SSP_SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSn line is returned to its idle high state one SSP_SCK period after the last bit has been captured.

For continuous back-to-back transmissions, the SSn pin remains in its active low state, until the final bit of the last word has been captured, and then returns to its idle state as described above.

For continuous back-to-back transfers, the SSn pin is held low between successive data words and termination is same as that of a single word transfer.

17.6 Winbond SPI Mode

The Winbond SPI mode is similar to Motorola's SPI mode when POLARITY = PHASE, where data is sampled on the rising edge. In addition to serial 1-bit reads and writes, 2-bit (dual) and 4-bit (quad) reads are supported. Only 8-bit word length is supported for dual and quad read modes. See Figure 17-8. The numbers in the IO signal waveform correspond to the bit position in the byte being transferred.

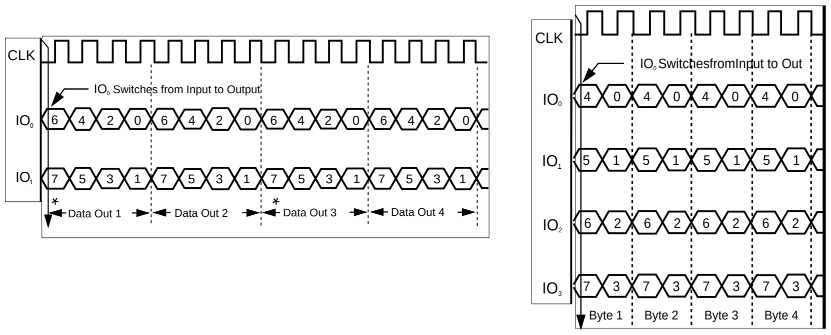


Figure 17-8. Fast Read Dual and Quad Output Diagram

17.7 Texas Instruments Synchronous Serial Interface (SSI) Mode

Figure 17-9 shows the Texas Instruments synchronous serial frame format for a single transmitted frame.

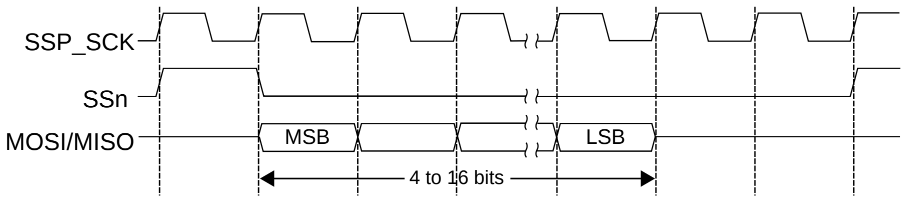


Figure 17-9. Texas Instruments Synchronous Serial Frame Format (Single Transfer)

In this mode, SSP_SCK and SSn are forced low, and the transmit data line MOSI is three-stated whenever the SSP is idle. Once the bottom entry of the FIFO contains data, SSn is pulsed high for one SSP_SCK period. The value to be transmitted is also transferred from the FIFO to the serial shift register of the transmit logic. On the next rising edge of SSP_SCK, the MSB of the 4-to-16-bit data frame is shifted out on the SSPRXD pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each SSP_SCK. The received data is transferred from the serial shifter to the FIFO on the first rising edge of SSP_SCK after the LSB has been latched.

Figure 17-10 shows the Texas Instrument synchronous serial frame format when back-to-back frames are transmitted.

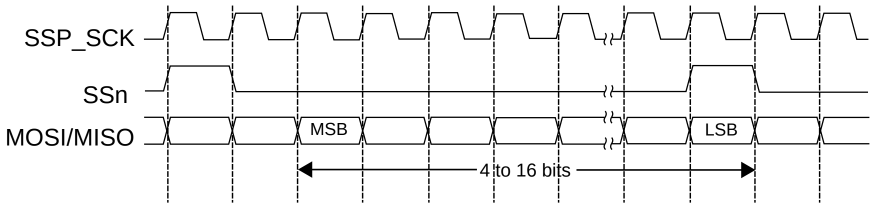


Figure 17-10. Texas Instruments Synchronous Serial Frame Format (Continuous Transfer)

17.8 SD/SDIO/MMC Mode

This mode is used to provide high performance with SD, SDIO, MMC, and high-speed (4-bit and 8-bit) MMC cards. When running 4-bit and 8-bit modes, the SSP has the capability of SDR and DDR operations. Power-on and alternate boot operations are also supported.

SD/MMC mode supports simultaneous command and data transfers. Commands are sent to the card and responses are returned to the host on the CMD line. Register data, such as card information, is sent as a command response and is therefore on the CMD line. Block data read from or written to the card's flash is transferred on the DAT line(s). The SSP also supports the SDIO IRQ.

The SSP's SD/MMC controller can automatically perform a single block read/write or card register operation with a single PIO setup and RUN. For example, the SD/MMC controller can perform these steps with a single write to the PIO registers:

- Send command to the card.
- Receive response from the card.
- Check response for errors (and assert a CPU IRQ if there is an error).
- Wait for the DAT line(s) to be ready to transfer data (while counting for time-out).
- Transfer multiple blocks of data to/from the card.
- Check the CRC or CRC status of received/sent data (and assert IRQ if there is an error).

The SD/MMC controller is generally used with the DMA. Each DMA descriptor is set up the SD/MMC controller to perform a single complex operation as exemplified above. Multiple DMA descriptors can be chained to perform multiple card block transfers without CPU intervention. A single DMA descriptor can also perform multiple card block transfers.

17.8.1 SD/MMC Command/Response Transfer

SD/MMC commands are written to the HW_SSP_CMDn registers and sent on the CMD line. Command tokens consist of a start bit (0), a source bit (1), the actual command, which is padded to 38 bits, a 7-bit CRC and a stop bit (1). The command token format is shown in Table 17-2.

Table 17-2. SD/MMC Command/Response Transfer

Line	Start	Source	Data	CRC	End
CMD	0	1 (Host)	38-bit Command	CRC7	1

SD/MMC cards transmit command words with the most significant bit first. After the card receives the command, it checks for CRC errors or invalid commands. If an error occurs, the card withholds the usual response to the command.

After transmitting the end bit, the SSP releases the CMD line to the high-impedance state. A pullup resistor on the CMD node keeps it at the 1 state until the response packet is received. The slave waits to issue a reply until the SCK line is clocking again.

After the SSP sends an SD/MMC command, it optionally starts looking for a response from the card. It waits for the CMD line to go low, indicating the start of the response token. Once the SSP has received the Start and Source bits, it begins shifting the response content into the receive shift register. The SSP calculates the CRC7 of the incoming data.

If the card fails to start sending an expected response packet within 64 SCK cycles, then an error occurrs; the command may be invalid or have a bad CRC. After the SSP detects a time-out, it stops any DMA request activity and sets the RESP_TIMEOUT flag. If RESP_TIMEOUT_IRQ_EN is set, then a CPU IRQ is asserted.

The SSP calculates the CRC of the received response and compares it to the CRC received from the card. If they do not match, then the SSP sets the RESP_ERR status flag. If RESP_ERR_IRQ_EN is set, then a CPU IRQ is asserted on a command response CRC mismatch.

The SSP can also compare the 32-bit card status word, known as response R1, against a reference to check for errors. If CHECK_RESP in HW_SSP_CTRL0 is set, then the SSP XORs the response with the XOR field in the HW_SSP_COMPREF register. It then masks the results with the MASK field in the HW_SSP_COMPMASK register. If there are any differences between the masked response and the reference, then an error will occur. The CPU asserts the RESP_ERR status flag. If RESP_ERR_IRQ_EN is set, then the RESP_ERR_IRQ is asserted. In the ISR, the CPU can read the status word to see which error flags are set.

The regular and long response tokens are shown in Table 17-3 and Table 17-4:

Table 17-3. SD/MMC Command Regular Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	38-bit Response	CRC7	1

Table 17-4. SD/MMC Command Regular Long Response Token

Line	Start	Source	Data	CRC	End
CMD	0	0 (Card)	117-bit response	CRC16	1

17.8.2 SD/MMC Data Block Transfer

Block data is transferred on the DATA0 pin. In 1-bit I/O mode, the block data is formatted as shown in Table 17-5. Block data transfers typically have 512 bytes of payload, plus a 16-bit CRC, a Start bit, and an End bit. The block size is programmable with the XFER_COUNT field in the HW_SSP_XFER_SIZE register. In SD/MMC mode, WORD_LENGTH in the HW_SSP_CTRL1 register field should always be set to 8 bits. Data is always sent Most Significant Bit of the Least Significant Byte first.

The SSP is designed to support block transfer modes only. Streaming modes may not be supported. Figure 17-11 shows a flowchart of SD/MMC block read and write transfers.

In block write mode, the card holds the DATA0 line low while it is busy. SSP must wait for the DATA0 line to be high for one clock cycle before starting to write to a block.

In block read mode, the card begins sending the data when it is ready. The first bit transmitted by the card is a Start bit 0. Prior to the 0 Start Bit, the DATA0 bus is high. After the Start bit is received, data is shifted in. The SSP bus width is set using the BUS_WIDTH bit in the HW_SSP_CTRL0 register.

In 1-bit bus mode, the block data is formatted as shown in Table 17-5.

Table 17-5. SD/MMC Data Block Transfer 1-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA0	0	Data Bit 7 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

In 4-bit I/O mode, the block data is formatted as shown in Table 17-6.

Table 17-6. SD/MMC Data Block Transfer 4-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA3	0	Data Bit 7 Byte 0	...	Data Bit 3 Byte 511	CRC16	1
DATA2	0	Data Bit 6 Byte 0	...	Data Bit 2 Byte 511	CRC16	1
DATA1	0	Data Bit 5 Byte 0	...	Data Bit 1 Byte 511	CRC16	1
DATA0	0	Data Bit 4 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

In 8-bit bus mode, the block data is formatted as shown in Table 17-7.

Table 17-7. SD/MMC Data Block Transfer 8-Bit Bus Mode

Line	Start	Data	Data	Data	CRC	End
DATA7	0	Data Bit 7 Byte 0	...	Data Bit 7 Byte 511	CRC16	1
DATA6	0	Data Bit 6 Byte 0	...	Data Bit 6 Byte 511	CRC16	1
DATA5	0	Data Bit 5 Byte 0	...	Data Bit 5 Byte 511	CRC16	1
DATA4	0	Data Bit 4 Byte 0	...	Data Bit 4 Byte 511	CRC16	1
DATA3	0	Data Bit 3 Byte 0	...	Data Bit 3 Byte 511	CRC16	1
DATA2	0	Data Bit 2Byte 0	...	Data Bit 2 Byte 511	CRC16	1
DATA1	0	Data Bit 1 Byte 0	...	Data Bit 1 Byte 511	CRC16	1
DATA0	0	Data Bit 0 Byte 0	...	Data Bit 0 Byte 511	CRC16	1

17.8.2.1 SD/MMC Multiple Block Transfers

The SSP supports SD/MMC multiple block transfers. The CPU or DMA will configure the SD/MMC controller to issue a Multi-Block Read or Write command. If DMA is used, then the first descriptor issues the multi-block read/write command and receives/sends the first block (512 bytes) of data. Subsequent DMA descriptors only receive/send blocks of data and do not issue new SD/MMC commands. If the card is configured for an open-ended multi-block transfer, then the last DMA descriptor needs to issue a STOP command to the card. Multiple blocks can also be transferred with a single DMA descriptor.

After each block of data has been transferred, the SSP sends/receives the CRC and checks the CRC or the CRC token. If the CRC is correct, then the SSP signals the DMA that it is completed.

The SSP supports transferring multiple SD/MMC blocks per DMA descriptor. The SSP state machine needs to know the number of blocks and the size of a block. When HW_SSP_BLOCK_SIZE_BLOCK_COUNT is non-zero, the actual block size is:

0x1 shiftleft BLOCK_SIZE

For example, setting a value of 9 will result in a block size of 512 bytes. When BLOCK_COUNT is 0, BLOCK_SIZE is ignored and the bit field HW_SSP_XFER_SIZE_XFER_COUNT represents the single block size or the number of byte to transfer. This must satisfy the equation:

HW_SSP_XFER_SIZE_XFER_COUNT = (0x1 shiftleft BLOCK_SIZE) x (BLOCK_COUNT+1)

for BLOCK_COUNT greater than 0.

17.8.2.2 eMMC DDR operation

When performing a single block or multiple block transfer, the SSP has an option of using DDR operation by setting DBL_DATA_RATE_EN bit in HW_SSP_CMD0 register. When performing a DDR operation, the timing relationship between SCK and DATA can be programmed to suit different needs. The highly recommended settings are in the following (but not necessary):

- To set TXCLK_DELAY_TYPE of HW_SSP_DDR_CTRL register allows SCK to have a 1/4 SCK delay and to switch approximately at the center of each TX DATA period.
- Turn on the DLL by setting ENABLE bit of HW_SSP_DLL_CTRL registerr. This generates a precisive clock delay relative to SCK (input version), which is used to sample RX DATA. Program the value of 4'd7 to DLV_DLY_TARGET field of HW_SSP_DLL_CTRL register, and this specifies the clock delay to be 1/4 of SCK period.

To boost the performance of SSP running in DDR mode and at the maximum frequency, the DMA interface allows a burst of 4 or 8 32-bit APB transfers per DMA request.

17.8.2.3 SD/MMC Block Transfer CRC Protection

The block of data transferred over the data bus is protected by CRC16. For reads, the SSP calculates the CRC of the incoming data and compares it to the CRC16 reference that is provided by the card at the end of the block. In DDR mode, two sets of CRC16s are calculated for both the rising edge sampled data and the falling edge sampled data. If any CRC mismatch occurs, then the block asserts the DATA_CRC_ERR status flag. If DATA_CRC_IRQ_EN is set, then a CPU IRQ is asserted.

For block write operations, the card determines if a CRC error has occurred. After the SSP has sent a block of data, it transmits the reference CRC16. In DDR mode, two sets of reference CRC16s are transmitted. The card compares that(those) to its calculated CRC16(s). The card then sends a CRC status token on the DATA bus. It sends a positive status ('010') if the transfer was good, and a negative status ('101') if the CRC16(s) did not match. If the SSP receives a CRC bad token, it sets the DATA_CRC_ERROR in the HW_SSP_STATUS register, and then it indicates it to the CPU if DATA_CRC_IRQ_EN is set.

17.8.3 eMMC Boot Operation

In boot operation mode, the SSP can read boot data from the slave (MMC device) by keeping SSP_CMD line low using PRIM_BOOT_OP_EN bit, or sending CMD0 with argument = 0xFFFFFFFA, before issuing CMD1. User can terminate the boot operation by setting SOFT_TERMINATE bit in HW_SSP_CMD0 register when using primary boot method, or send a CMD0 command when using the alternate boot method. Boot acknowledge is also supported by setting BOOT_ACK_EN bit in HW_SSP_CMD0 register. The boot mode supports DDR operations as specified in eMMC 4.4 standard.

17.8.4 SDIO Interrupts

The SSP supports SDIO interrupts. When the SSP is in SD/MMC mode and the SDIO_IRQ bit in the HW_SSP_CTRL0 register is set, the SSP looks for interrupts on DATA1 during the valid IRQ periods. The valid IRQ periods are defined in the SDIO specification. If the card asserts an interrupt and SDIO_IRQ_EN is set, then the SSP sets the SDIO_IRQ status bit and asserts a CPU IRQ. Other than detecting when card IRQs are valid, the SDIO IRQ function operates independently from the rest of the SSP. After the CPU receives an IRQ, it should monitor the SSP and DMA status to determine when it should send commands to the SDIO card to handle the interrupt.

17.8.5 SD/MMC Mode Error Handling

There are several errors that can occur during the SD/MMC operation. These errors can be caused by normal unexpected events, such as having a card removed or unusual events such as a card failure. The detected error cases are listed below. Please note that in all the cases below, a CPU IRQ is only asserted if DATA_CRC_IRQ_EN is set in HW_SSP_CTRL1 register.

- **Data Receive CRC Error**—Detected by the SSP after a block receive. If this occurs, the SSP will not indicate to DMA that the transfer is complete. It will set the DATA_CRC_ERR status flag and assert a CPU IRQ. The ISR should reset the SSP DMA channel and instruct the DMA to re-try the block read operation.
- **Data Transmit CRC Error**—Transmit CRC error token is received from the SD/MMC card on the DAT line after a block transmit. If this occurs, the SSP will not indicate to DMA that the transfer is complete. It will set the DATA_CRC_ERR status flag and assert a CPU IRQ. The ISR should reset the SSP DMA channel and instruct the DMA to re-try the block write operation.

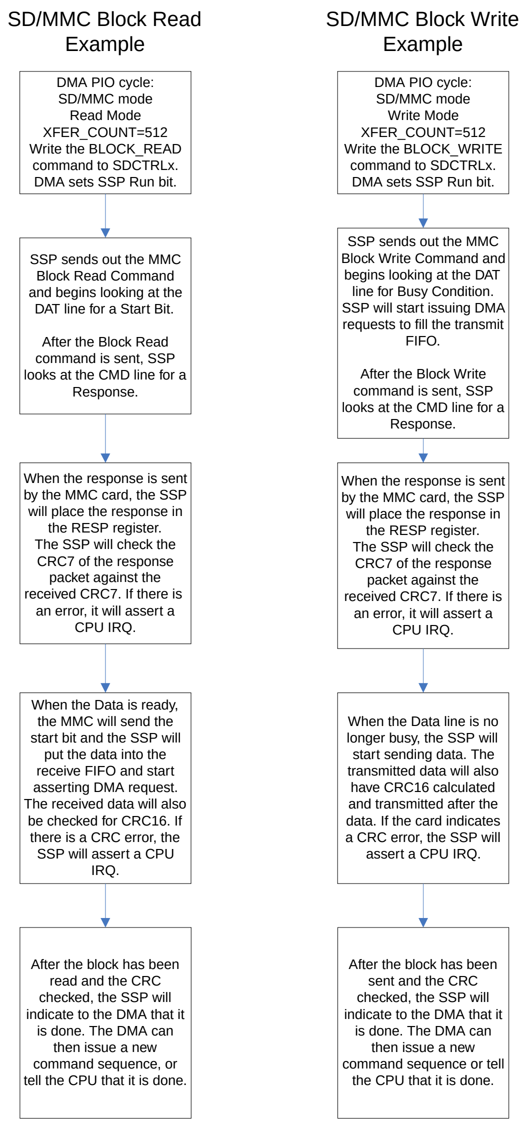

SD/MMC Block Read Example
DMA PIO cycle: SD/MMC mode Read Mode XFER_COUNT=512 Write the BLOCK_READ command to SDCTRLx. DMA sets SSP Run bit.
SSP sends out the MMC Block Read Command and begins looking at the DAT line for a Start Bit.
After the Block Read command is sent, SSP looks at the CMD line for a Response.
When the response is sent by the MMC card, the SSP will place the response in the RESP register. The SSP will check the CRC7 of the response packet against the received CRC7. If there is an error, it will assert a CPU IRQ.
When the Data is ready, the MMC will send the start bit and the SSP will put the data into the receive FIFO and start asserting DMA request. The received data will also be checked for CRC16. If there is a CRC error, the SSP will assert a CPU IRQ.
After the block has been read and the CRC checked, the SSP will indicate to the DMA that it is done. The DMA can then issue a new command sequence, or tell the CPU that it is done.
SD/MMC Block Write Example
DMA PIO cycle: SD/MMC mode Write Mode XFER_COUNT=512 Write the BLOCK_WRITE command to SDCTRLx. DMA sets SSP Run bit.
SSP sends out the MMC Block Write Command and begins looking at the DAT line for Busy Condition. SSP will start issuing DMA requests to fill the transmit FIFO.
After the Block Write command is sent, SSP looks at the CMD line for a Response.
When the response is sent by the MMC card, the SSP will place the response in the RESP register. The SSP will check the CRC7 of the response packet against the received CRC7. If there is an error, it will assert a CPU IRQ.
When the Data line is no longer busy, the SSP will start sending data. The transmitted data will also have CRC16 calculated and transmitted after the data. If the card indicates a CRC error, the SSP will assert a CPU IRQ.
After the block has been sent and the CRC checked, the SSP will indicate to the DMA that it is done. The DMA can then issue a new command sequence or tell the CPU that it is done.

Figure 17-11. SD/MMC Block Transfer Flowchart

- **Data Time-Out Error**—The SSP TIMEOUT counter is used to detect a time-out condition during data write or read operations. The time-out counts any time that the SSP is waiting on a busy DAT bus. For read operations, the DAT line(s) indicate busy before the card sends the start bit. For write operations, the DAT line(s) may indicate busy after the block has been sent to the card. If the time-out counter expires before the DAT line(s) become ready, the SSP stops any DMA requests, sets the DATA_TIMEOUT status flag, and asserts a CPU IRQ. The ISR should check the status register to see that a data time-out has occurred. It can then reset the DMA channel and the SSP to re-try the operation.

- **DMA Overflow/Underflow**—The SSP should stop SCK if the FIFO is full or the FIFO is empty during data transfer. So, a DMA underflow or overflow should not occur. However, if it does due to some unforeseen problem, the FIFO_OVRFLW or FIFO_UNDRFLW status bit is set in the SSP Status Register and asserts a CPU IRQ.
- **Command Response Error**—The SD/MMC card returns a R1 status response after most commands. The SSP can compare the R1 response against a mask/reference pair. If any of the enabled bits are set, then an error will occur. The SSP stops requesting any DMAs, sets the RESP_ERR status flag, and asserts a CPU IRQ. The CPU can read the SSP Status Register to see the RESP_ERR flag and read the HW_SSP_SDRESP0 register to get the actual response from the SD/MMC card. That response contains the specific error information. Once the error is understood, the CPU can reset the DMA channel and the SSP and re-try the operation or take some other action to recover or inform the user of a non-recoverable error.
- **Command Response Time-Out**—If an expected response is not received within 64 SCK cycles, then the command response has timed out. If this occurs, the SSP stops any DMA requests, stops transferring data to the card, sets the RESP_TIME-OUT status flag, and asserts the RESP_TIME-OUT_IRQ. The ISR should read the status register to find that a command response time-out has occurred. It can then decide to reset the DMA channel and SSP and re-try the operation.

17.8.6 SD/MMC Clock Control

- When SD/MMC block is idle, the serial clock (SCK) toggling will be based on the value of CONT_CLKING_EN and SLOW_CLKING_EN. See HW_SSP_CMD0 register description.
- SCK runs any time that RUN is set and a data or command is active or pending. If a command has been sent and a response is expected, then SCK continues to run until the response is received. If a data operation is active or if the DAT line is busy, then SCK runs.
- If CONT_CLKING_EN=0, SCK stops running if received command response status R1 indicates an error.
- If CONT_CLKING_EN=0, SCK stops running if a data operation has timed out or a CRC error has occurred.
- If CONT_CLKING_EN=0, SCK stops running after all pending commands and data operations have completed. SCK restarts when a new command or data operation has been requested.

17.9 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

17.10 Programmable Registers

SSP Hardware Register Format Summary

SSP0 base address is 0x80010000; SSP1 base address is 0x80012000; SSP2 base address is 0x80014000; SSP3 base address is 0x80016000

HW_SSP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_0000	SSP Control Register 0 (HW_SSP_CTRL0)	32	R/W	C000_0000h	17.10.1/1331
8001_0010	SD/MMC Command Register 0 (HW_SSP_CMD0)	32	R/W	0000_0000h	17.10.2/1333
8001_0020	SD/MMC Command Register 1 (HW_SSP_CMD1)	32	R/W	0000_0000h	17.10.3/1336
8001_0030	Transfer Count Register (HW_SSP_XFER_SIZE)	32	R/W	0000_0001h	17.10.4/1336
8001_0040	SD/MMC BLOCK SIZE and COUNT Register (HW_SSP_BLOCK_SIZE)	32	R/W	0000_0000h	17.10.5/1337
8001_0050	SD/MMC Compare Reference (HW_SSP_COMPREF)	32	R/W	0000_0000h	17.10.6/1337
8001_0060	SD/MMC compare mask (HW_SSP_COMPMASK)	32	R/W	0000_0000h	17.10.7/1338
8001_0070	SSP Timing Register (HW_SSP_TIMING)	32	R/W	0000_0000h	17.10.8/1338
8001_0080	SSP Control Register 1 (HW_SSP_CTRL1)	32	R/W	0000_0080h	17.10.9/1339
8001_0090	SSP Data Register (HW_SSP_DATA)	32	R/W	0000_0000h	17.10.10/1342
8001_00A0	SD/MMC Card Response Register 0 (HW_SSP_SDRESP0)	32	R	0000_0000h	17.10.11/1342
8001_00B0	SD/MMC Card Response Register 1 (HW_SSP_SDRESP1)	32	R	0000_0000h	17.10.12/1342
8001_00C0	SD/MMC Card Response Register 2 (HW_SSP_SDRESP2)	32	R	0000_0000h	17.10.13/1343
8001_00D0	SD/MMC Card Response Register 3 (HW_SSP_SDRESP3)	32	R	0000_0000h	17.10.14/1343
8001_00E0	SD/MMC Double Data Rate Control Register (HW_SSP_DDR_CTRL)	32	R/W	0000_0000h	17.10.15/1344
8001_00F0	SD/MMC DLL Control Register (HW_SSP_DLL_CTRL)	32	R	0000_0000h	17.10.16/1345
8001_0100	SSP Status Register (HW_SSP_STATUS)	32	R	E000_0020h	17.10.17/1346
8001_0110	SD/MMC DLL Status Register (HW_SSP_DLL_STS)	32	R	0000_0000h	17.10.18/1348
8001_0120	SSP Debug Register (HW_SSP_DEBUG)	32	R	0000_0000h	17.10.19/1349

HW_SSP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_0130	SSP Version Register (HW_SSP_VERSION)	32	R	0400_0000h	17.10.20/1350

17.10.1 SSP Control Register 0 (HW_SSP_CTRL0)

SSP Control Register 0

HW_SSP_CTRL0: 0x000

HW_SSP_CTRL0_SET: 0x004

HW_SSP_CTRL0_CLR: 0x008

HW_SSP_CTRL0_TOG: 0x00C

This is the most frequently changed fields.

Address: HW_SSP_CTRL0 – 8001_0000h base + 0h offset = 8001_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	SFTRST	CLKGATE	RUN	SDIO_IRQ_CHECK	LOCK_CS	IGNORE_CRC	READ	DATA_XFER	BUS_WIDTH		WAIT_FOR_IRQ	WAIT_FOR_CMD	LONG_RESP	CHECK_RESP	GET_RESP	ENABLE
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_CTRL0 field descriptions

Field	Description
31 SFTRST	SSP Reset - 0: SSP is not Reset. 1: SSP Held in Reset. After Reset, all registers are returned to their reset state. This will not work if the CLKGATE bit is already set to \'1\'. CLKGATE must be cleared to \'0\' before issuing a soft reset. Also the SSPCLK must be running for this to work properly.
30 CLKGATE	Gate SSP Clocks - 0: SSP Clocks not gated. 1: SSP Clocks are gated. Set this to save power while the SSP is not actively being used. Configuration state is kept while the clock is gated.
29 RUN	SSP Run. 0: SSP is not running. 1: SSP is running. Automatically set during DMA operation.

HW_SSP_CTRL0 field descriptions (continued)

Field	Description
28 SDIO_IRQ_ CHECK	In SD/MMC mode: SDIO IRQ: 1= Enable checking for SDIO Card IRQ.
27 LOCK_CS	In SPI mode: This affects the SSn output. When set to 1, SSn will be asserted throughout the current command. SSn will remain asserted after the command if IGNORE_CRC = 0. SSn will deassert at the end of the next command that has IGNORE_CRC = 1. In SD/MMC mode: 0= Look for a CRC status token from the card on DATA0 after a block write. 1= Ignore the CRC status response on DATA0 after a write operation. Note that the SD/MMC function should be used when performing MMC BUSTEST_W operation.
26 IGNORE_CRC	Ignore CRC - In SD/MMC. In SPI/SSI modes: When set to 1, deassert the chip select (SSn) pin after the command is executed.
25 READ	Read Mode - When this and DATA_XFER are set, the SSP will read data from the device. If this is not set, then the SSP will write data to the device.
24 DATA_XFER	Data Transfer Mode - When set, transfer XFER_COUNT bytes of data. When not set, the SSP will not transfer any data (command or Wait for IRQ only).
23–22 BUS_WIDTH	Data Bus Width - SD/MMC modes supports all widths, SSI mode supports only 1-bit bus width. In SPI mode 1, 2, and 4-bit bus widths are supported. In SPI mode the Data Bus Width field is redefined: 0 means 1-bit; 1 means 2-bit; 2 means 4-bit. 0x0 **ONE_BIT** — SD/MMC data bus is 1-bit wide 0x1 **FOUR_BIT** — SD/MMC data bus is 4-bits wide 0x2 **EIGHT_BIT** — SD/MMC data bus is 8-bits wide
21 WAIT_FOR_IRQ	Wait for IRQ In SD/MMC mode this signal means wait for MMC ready before sending command. (MMC is busy when databit0 is low.) In SPI/SSI mode this bit and WAIT_FOR_CMD bit select which SSn output to assert: (WAIT_FOR_IRQ,WAIT_FOR_CMD) = b00: SSn0 will assert during access (SSn2, SSn1 inactive); b01: SSn1 will assert during access (SSn2, SSn0 inactive); b10: SSn2 will assert during access (SSn1, SSn0 inactive).
20 WAIT_FOR_CMD	Wait for Data Done - SD/MMC - 0: Send commands immediately after they are written. 1: Wait to send command until after the CRC-checking phase of a data transfer has completed successfully. This delays sending a command until a block of data is transferred. This can be used to send a STOP command during an SD/MMC multi-block read. In SD/MMC mode this signal means wait for MMC ready before sending command. (MMC is busy when databit0 is low.) In SPI/SSI mode this bit and WAIT_FOR_IRQ bit select which SSn output to assert: (WAIT_FOR_IRQ,WAIT_FOR_CMD) = b00: SSn0 will assert during access (SSn2, SSn1 inactive); b01: SSn1 will assert during access (SSn2, SSn0 inactive); b10: SSn2 will assert during access (SSn1, SSn0 inactive).

HW_SSP_CTRL0 field descriptions (continued)

Field	Description
19 LONG_RESP	Get Long Response - SD/MMC - 0: The card response will be short. 1: The card will provide a 136-bit response. Only valid if GET_RESP is set. A long response cannot be checked using CHECK_RESP.
18 CHECK_RESP	Check Response - SD/MMC. If this bit is set, the SSP will XOR the result with the REFERENCE field and then mask the incoming status word with the MASK field in the COMPARE register. If there is a mismatch, then the SSP will set the RESP_ERR status bit, and, if enabled, the RESP_ERR_IRQ. This should not be used with LONG_RESP.
17 GET_RESP	Get Response - SD/MMC - 0: Do not wait for a response from the card. 1: This command should receive a response from the card.
16 ENABLE	Command Transmit Enable - SD/MMC - 0: Commands are not enabled. 1: Data in Command registers will be sent. This is normally enabled in SD/MMC.
15–0 RSVD0	Reserved

17.10.2 SD/MMC Command Register 0 (HW_SSP_CMD0)

SD/MMC Command Index and control register

HW_SSP_CMD0: 0x010

HW_SSP_CMD0_SET: 0x014

HW_SSP_CMD0_CLR: 0x018

HW_SSP_CMD0_TOG: 0x01C

This is the command code for the SD/MMC devices. See device specification for details. For eMMC Alternate boot operation, Command and RX data phases must be executed together (HW_SSP_CTRL0_ENABLE=1 and HW_SSP_CTRL0_DATA_XFER=1) so that SSP does not miss the boot ack or boot data phases.

Address: HW_SSP_CMD0 – 8001_0000h base + 10h offset = 8001_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0					SOFT_TERMINATE	DBL_DATA_RATE_EN	PRIM_BOOT_OP_EN	BOOT_ACK_EN	SLOW_CLKING_EN	CONT_CLKING_EN	APPEND_8CYC	RSVD1[19:16]			
W	▨	▨	▨	▨	▨								▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:8]								CMD							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_CMD0 field descriptions

Field	Description
31–27 RSVD0	Reserved
26 SOFT_ TERMINATE	Setting this bit to 1 causes the current operation to terminate itself under normal condition. This signal is used as edge-sensitive, so in order to create a subsequent termanation, SOFT_TERMINATE must be taken low and then asserted again.
25 DBL_DATA_ RATE_EN	This bit enables the Double Data Rate operation. The DDR operation only applies in 4-bit or 8-bit data transfers.
24 PRIM_BOOT_ OP_EN	Enable the primary method of Boot Operation where the SSP_CMD output is driven low for the entire boot operation. In primary Boot Operation mode, program the SSP to read one or more 512 byte blocks of data, and ensure command is dissabled (HW_SSP_CTRL0_ENABLE=0). Do not set this bit for the Alternate Boot Operation mode; instead, program the SSP to send command zero (with no response) and read one or more 512 byte blocks of boot data. Enabling the BOOT_ACK is optional.
23 BOOT_ACK_EN	Enable Boot Acknowledge reception from the slave during primary or alternate boot operation. For eMMC Alternate boot operation, Command and RX data phases must be executed together (HW_SSP_CTRL0_ENABLE=1 and HW_SSP_CTRL0_DATA_XFER=1) so that SSP does not miss the boot ack or boot data phases. This bit must be zero for non-boot operations.
22 SLOW_CLKING_ EN	Enable Continuous clocking on SCK to occur at a frequency eight times slower that when actively transferring command and data. This field is ignored when CONT_CLKING is zero.
21 CONT_CLKING_ EN	Set this bit to enable Continous clocking of SCK when no SD/MMC command/reaponse or data transfer is active. This is used in SD/MMC mode. When set to zero, SCK is idle when no transfer is taking place.
20 APPEND_8CYC	Append 8 SCK cycles. This is used in SD/MMC mode. When set to one, the SCK will toggle for 8 more cycles before going idle. When set to zero SCK will toggle up to 4 cycles before going idle. This should be set to one at the end of a single or multiple block transfer.
19–8 RSVD1	Reserved
7–0 CMD	SD/MMC Command Index (uses 5:0) to be sent to card. This is also SPI/SSI control word[7:0] for RX transfers. 0x00 **MMC_GO_IDLE_STATE** — 0x01 **MMC_SEND_OP_COND** — 0x02 **MMC_ALL_SEND_CID** — 0x03 **MMC_SET_RELATIVE_ADDR** — 0x04 **MMC_SET_DSR** —

HW_SSP_CMD0 field descriptions (continued)

Field	Description
	0x05 **MMC_RESERVED_5** — 0x06 **MMC_SWITCH** — 0x07 **MMC_SELECT_DESELECT_CARD** — 0x08 **MMC_SEND_EXT_CSD** — 0x09 **MMC_SEND_CSD** — 0x0A **MMC_SEND_CID** — 0x0B **MMC_READ_DAT_UNTIL_STOP** — 0x0C **MMC_STOP_TRANSMISSION** — 0x0D **MMC_SEND_STATUS** — 0x0E **MMC_BUSTEST_R** — 0x0F **MMC_GO_INACTIVE_STATE** — 0x10 **MMC_SET_BLOCKLEN** — 0x11 **MMC_READ_SINGLE_BLOCK** — 0x12 **MMC_READ_MULTIPLE_BLOCK** — 0x13 **MMC_BUSTEST_W** — 0x14 **MMC_WRITE_DAT_UNTIL_STOP** — 0x17 **MMC_SET_BLOCK_COUNT** — 0x18 **MMC_WRITE_BLOCK** — 0x19 **MMC_WRITE_MULTIPLE_BLOCK** — 0x1A **MMC_PROGRAM_CID** — 0x1B **MMC_PROGRAM_CSD** — 0x1C **MMC_SET_WRITE_PROT** — 0x1D **MMC_CLR_WRITE_PROT** — 0x1E **MMC_SEND_WRITE_PROT** — 0x23 **MMC_ERASE_GROUP_START** — 0x24 **MMC_ERASE_GROUP_END** — 0x26 **MMC_ERASE** — 0x27 **MMC_FAST_IO** — 0x28 **MMC_GO_IRQ_STATE** — 0x2A **MMC_LOCK_UNLOCK** — 0x37 **MMC_APP_CMD** — 0x38 **MMC_GEN_CMD** — 0x00 **SD_GO_IDLE_STATE** — 0x02 **SD_ALL_SEND_CID** — 0x03 **SD_SEND_RELATIVE_ADDR** — 0x04 **SD_SET_DSR** — 0x05 **SD_IO_SEND_OP_COND** — 0x07 **SD_SELECT_DESELECT_CARD** — 0x09 **SD_SEND_CSD** — 0x0A **SD_SEND_CID** — 0x0C **SD_STOP_TRANSMISSION** — 0x0D **SD_SEND_STATUS** — 0x0F **SD_GO_INACTIVE_STATE** — 0x10 **SD_SET_BLOCKLEN** — 0x11 **SD_READ_SINGLE_BLOCK** — 0x12 **SD_READ_MULTIPLE_BLOCK** — 0x18 **SD_WRITE_BLOCK** —

HW_SSP_CMD0 field descriptions (continued)

Field	Description
	0x19 **SD_WRITE_MULTIPLE_BLOCK** — 0x1B **SD_PROGRAM_CSD** — 0x1C **SD_SET_WRITE_PROT** — 0x1D **SD_CLR_WRITE_PROT** — 0x1E **SD_SEND_WRITE_PROT** — 0x20 **SD_ERASE_WR_BLK_START** — 0x21 **SD_ERASE_WR_BLK_END** — 0x23 **SD_ERASE_GROUP_START** — 0x24 **SD_ERASE_GROUP_END** — 0x26 **SD_ERASE** — 0x2A **SD_LOCK_UNLOCK** — 0x34 **SD_IO_RW_DIRECT** — 0x35 **SD_IO_RW_EXTENDED** — 0x37 **SD_APP_CMD** — 0x38 **SD_GEN_CMD** —

17.10.3 SD/MMC Command Register 1 (HW_SSP_CMD1)

SD/MMC Command Argument Register

Address: HW_SSP_CMD1 – 8001_0000h base + 20h offset = 8001_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_ARG																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_CMD1 field descriptions

Field	Description
31–0 CMD_ARG	SD/MMC Command Argument. See SSP_CTRL0_DATA_XFER bit description.

17.10.4 Transfer Count Register (HW_SSP_XFER_SIZE)

Transfer Count Register.

Address: HW_SSP_XFER_SIZE – 8001_0000h base + 30h offset = 8001_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XFER_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_SSP_XFER_SIZE field descriptions

Field	Description
31–0 XFER_COUNT	Number of words to transfer, as referenced in WORD_LENGTH in HW_SSP_CTRL1. The run bit and DMA request will clear after this many words have been transferred. In SD/MMC, this should be a multiple of the block size.

17.10.5 SD/MMC BLOCK SIZE and COUNT Register (HW_SSP_BLOCK_SIZE)

SD/SDIO/MMC Multiple Block Size and Count Register.

Address: HW_SSP_BLOCK_SIZE – 8001_0000h base + 40h offset = 8001_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0				BLOCK_COUNT																								BLOCK_SIZE			
W	▨	▨	▨	▨																												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_BLOCK_SIZE field descriptions

Field	Description
31–28 RSVD0	Reserved
27–4 BLOCK_COUNT	SD/MMC block count. This value one less than the number of blocks to transfer. For example setting a value of one will transfer two blocks. This must satisfy equation HW_SSP_XFER_SIZE_XFER_COUNT = (1 shiftleft BLOCK_SIZE) x (BLOCK_COUNT+1) for BLOCK_COUNT greater than zero. This is also SPI/SSI control word[15:8] for RX transfers.
3–0 BLOCK_SIZE	SD/MMC block size encode. When BLOCK_COUNT is nonzero, the actual block size is (1 shiftleft BLOCK_SIZE). For example setting a value of 9 will result in a block size of 512 bytes. When BLOCK_COUNT is zero, BLOCK_SIZE is ignored and HW_SSP_XFER_SIZE_XFER_COUNT represents the single block size or the number of byte to transfer. This must satisfy equation HW_SSP_XFER_SIZE_XFER_COUNT = (1 shiftleft BLOCK_SIZE) x (BLOCK_COUNT+1) for BLOCK_COUNT greater than zero.

17.10.6 SD/MMC Compare Reference (HW_SSP_COMPREF)

SD/MMC status can be compared with a reference value.

Address: HW_SSP_COMPREF – 8001_0000h base + 50h offset = 8001_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	REFERENCE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_COMPREF field descriptions

Field	Description
31–0 REFERENCE	SD/MMC Compare mode reference. If CHECK_RESP is set, the response will be XOR'd with this value. The results will be masked by the MASK bitfield. If there are any differences, then the SSP will indicate an error state to the DMA.

17.10.7 SD/MMC compare mask (HW_SSP_COMPMASK)

A mask allows the comparison of one or more bit fields in the reference value.

Address: HW_SSP_COMPMASK – 8001_0000h base + 60h offset = 8001_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_COMPMASK field descriptions

Field	Description
31–0 MASK	SD/MMC Compare mode Mask. If CHECK_RESP is set, the response is compared to REFERENCE, and the results are masked by this bitfield.

17.10.8 SSP Timing Register (HW_SSP_TIMING)

SSP Timing Config Register

The Timeout field and the clock dividers are contained in this register.

Address: HW_SSP_TIMING – 8001_0000h base + 70h offset = 8001_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TIMEOUT																CLOCK_DIVIDE								CLOCK_RATE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_TIMING field descriptions

Field	Description
31–16 TIMEOUT	Timeout counter. This specifies the number of SCK cycles multiplied by 4096, to wait before asserting the DATA TIMEOUT IRQ. It is used during timeout is used for data transfer/write operations in SD/MMC modes.
15–8 CLOCK_DIVIDE	Clock Pre-Divider. CLOCK_DIVIDE must be an even value from 2 to 254.
7–0 CLOCK_RATE	Serial Clock Rate. The value CLOCK_RATE is used to generate the transmit and receive bit rate of the SSP. The bit rate is SSPCLK / (CLOCK_DIVIDE x (1+ CLOCK_RATE)). CLOCK_RATE is a value from 0 to 255.

17.10.9 SSP Control Register 1 (HW_SSP_CTRL1)

Control Register 1.

HW_SSP_CTRL1: 0x080

HW_SSP_CTRL1_SET: 0x084

HW_SSP_CTRL1_CLR: 0x088

HW_SSP_CTRL1_TOG: 0x08C

This contains interrupt status and enable fields.

Address: HW_SSP_CTRL1 – 8001_0000h base + 80h offset = 8001_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SDIO_IRQ	SDIO_IRQ_EN	RESP_ERR_IRQ	RESP_ERR_IRQ_EN	RESP_TIMEOUT_IRQ	RESP_TIMEOUT_IRQ_EN	DATA_TIMEOUT_IRQ	DATA_TIMEOUT_IRQ_EN	DATA_CRC_IRQ	DATA_CRC_IRQ_EN	FIFO_UNDERRUN_IRQ	FIFO_UNDERRUN_EN	RSVD2	RSVD1	RECV_TIMEOUT_IRQ	RECV_TIMEOUT_IRQ_EN
W													▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIFO_OVERRUN_IRQ	FIFO_OVERRUN_IRQ_EN	DMA_ENABLE	RSVD0	SLAVE_OUT_DISABLE	PHASE	POLARITY	SLAVE_MODE	WORD_LENGTH				SSP_MODE			
W				▨												
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

HW_SSP_CTRL1 field descriptions

Field	Description
31 SDIO_IRQ	If this is set, an SDIO card interrupt has occurred and an IRQ, if enabled, has been sent to the ICOLL. Write a one to the SCT Clear address to reset this interrupt request status bit.
30 SDIO_IRQ_EN	SDIO Card Interrupt IRQ Enable. 0: SDIO card IRQs masked. 1: SDIO card IRQs will be sent to the ICOLL.
29 RESP_ERR_IRQ	When the CHECK_RESP bit in CTRL0 is set, if an unexpected response (or response CRC) is received from the card, this bit will be set. Write a one to the SCT Clear address to reset this interrupt request status bit.
28 RESP_ERR_ IRQ_EN	SD/MMC Card Error IRQ Enable. 0: Card Error IRQ is Masked. 1: Card Error IRQ is enabled. When set to 1, if an SD/MMC card indicates a card error (bit is set in both the SD/MMC Error Mask and R1 Card Status response), then a CPU IRQ will be asserted.
27 RESP_ TIMEOUT_IRQ	If this is set, a command response timeout has occurred, and an IRQ, if enabled, has been sent to the IRQ Collector. This is used for SD/MMC response timeout. Write a one to the SCT Clear address to reset this interrupt request status bit.
26 RESP_ TIMEOUT_IRQ_ EN	SD/MMC Card Command Respone Timeout Error IRQ Enable. 0: Response Timeout IRQ is Masked. 1: Response Timeout IRQ is enabled. When set to 1, if an SD/MMC card does not respond to a command within 64 cycles, then this CPU IRQ will be asserted.
25 DATA_ TIMEOUT_IRQ	Data Transmit/Receive Timeout Error IRQ. If the timeout counter expires before the DAT bus is ready for write or sends read data, then a data timeout has occurred. Only Valid For SD/MMC Modes. Write a one to the SCT Clear address to reset this interrupt request status bit.
24 DATA_ TIMEOUT_IRQ_ EN	Data Transmit/Receive Timeout Error IRQ Enable. If the timeout counter expires before the DAT bus is ready for write or sends read data, then a data timeout has occurred. Only Valid For SD/MMC Modes.
23 DATA_CRC_IRQ	Data Transmit/Receive CRC Error IRQ. Only valid for SD/MMC Modes. Write a one to the SCT Clear address to reset this interrupt request status bit.
22 DATA_CRC_ IRQ_EN	Data Transmit/Receive CRC Error IRQ Enable. Only valid for SD/MMC Modes.
21 FIFO_ UNDERRUN_IRQ	FIFO Underrun Interrupt. If the FIFO is read when it is empty this bit will be set. Write a one to the SCT Clear address to reset this interrupt request status bit.
20 FIFO_ UNDERRUN_EN	FIFO Underrun IRQ Enable. If set and the FIFO_UNDERRUN_IRQ bit is asserted, an IRQ will be generated.
19 RSVD2	Reserved
18 RSVD1	Reserved
17 RECV_ TIMEOUT_IRQ	Data Timeout Interrupt. If enabled and the FIFO is not empty, an IRQ will be generated if 128 HCLK Cycles Pass before the DATA register is read. This is supported for SPI modes only (Modes 0,1,2). Write a one to the SCT Clear address to reset this interrupt request status bit.

HW_SSP_CTRL1 field descriptions (continued)

Field	Description
16 RECV_ TIMEOUT_IRQ_ EN	Receive Timeout. If set and the FIFO is not empty, an IRQ will be generated if 128 HCLK Cycles Pass before the DATA register is read.
15 FIFO_ OVERRUN_IRQ	FIFO Overrun Interrupt. Indicated that the FIFO has been written to while full. Write a one to the SCT Clear address to reset this interrupt request status bit.
14 FIFO_ OVERRUN_IRQ_ EN	FIFO Overrun Interrupt Enable. If set, an IRQ will be generated if the FIFO is written to while full.
13 DMA_ENABLE	DMA Enable. This signal enables DMA request and DMA Command End signals to be asserted.
12 RSVD0	Reserved
11 SLAVE_OUT_ DISABLE	Slave Output Disable. 0: SSP can drive MISO in Slave Mode. 1: SSP does not drive MISO in slave mode.
10 PHASE	Serial Clock Phase. For SPI mode only.
9 POLARITY	Serial Clock Polarity. In SD/MMC mode, 0: Command and TX data change after rising edge of SCK. In SPI mode, 0: Steady-state '0' on SCK when data is not being transferred. 1: Steady-state '1' on SCK when data is not being transferred.
8 SLAVE_MODE	Slave Mode. 0: SSP is in Master Mode. 1: SSP is in Slave Mode. Set to zero for SD/MMC modes.
7–4 WORD_LENGTH	Word Length in bits per word. 0x0 to 0x2 are Reserved and Undefined. 0x3 is 4-bits per word...0xF is 16-bits per word. Always use 8 bits per word in SD/MMC Modes. 0x0 **RESERVED0** — 0x0 is Reserved and Undefined 0x1 **RESERVED1** — 0x1 is Reserved and Undefined 0x2 **RESERVED2** — 0x2 is Reserved and Undefined 0x3 **FOUR_BITS** — use 4-bits per word 0x7 **EIGHT_BITS** — use 8-bits per word 0xF **SIXTEEN_BITS** — use 16-bits per word
3–0 SSP_MODE	Operating Mode. 0x0 = Motorola and Winbond SPI Mode, 0x1 = TI syncronous serial mode, 0x3 = SD/MMC Card. All other values are undefined. Before changing ssp_mode, a softreset must be issued to clear the FIFO. 0x0 **SPI** — Motorola and Winbond SPI mode 0x1 **SSI** — Texas Instruments SSI mode 0x3 **SD_MMC** — SD/MMC mode —

17.10.10 SSP Data Register (HW_SSP_DATA)

The HW_SSP_DATA register allows user access to the SSP FIFO.

This register is the gateway for data transfer to and from the attached device.

Address: HW_SSP_DATA – 8001_0000h base + 90h offset = 8001_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_DATA field descriptions

Field	Description
31–0 DATA	Data Register. Holds one, two, three, or four words, depending on the WORD_LENGTH. If WORD_LENGTH is not 8, 16, or 32, the words are padded to those lengths. Data is right justified. When the run bit is set reads will cause the FIFO read pointer to increment and writes will cause the FIFO write pointer to increment.

17.10.11 SD/MMC Card Response Register 0 (HW_SSP_SDRESP0)

SD/SDIO/MMC Card Response Register 0.

Address: HW_SSP_SDRESP0 – 8001_0000h base + A0h offset = 8001_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESP0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_SDRESP0 field descriptions

Field	Description
31–0 RESP0	SD/MMC Response Status Bits[31:0]. See SSP_CTRL0_DATA_XFER bit description.

17.10.12 SD/MMC Card Response Register 1 (HW_SSP_SDRESP1)

SD/SDIO/MMC Card Response Register 1.

Address: HW_SSP_SDRESP1 – 8001_0000h base + B0h offset = 8001_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESP1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_SDRESP1 field descriptions

Field	Description
31–0 RESP1	SD/MMC Long Response [63:32]

17.10.13 SD/MMC Card Response Register 2 (HW_SSP_SDRESP2)

SD/SDIO/MMC Card Response Register 2.

Address: HW_SSP_SDRESP2 – 8001_0000h base + C0h offset = 8001_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESP2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_SDRESP2 field descriptions

Field	Description
31–0 RESP2	SD/MMC Long Response [95:64]

17.10.14 SD/MMC Card Response Register 3 (HW_SSP_SDRESP3)

SD/SDIO/MMC Card Response Register 3.

Address: HW_SSP_SDRESP3 – 8001_0000h base + D0h offset = 8001_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESP3																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_SDRESP3 field descriptions

Field	Description
31–0 RESP3	SD/MMC Long Response [127:96]

17.10.15 SD/MMC Double Data Rate Control Register (HW_SSP_DDR_CTRL)

SD/MMC Double Data Rate Control Register.

This register provides programmability in DDR mode for data output timing and data formats.

Address: HW_SSP_DDR_CTRL – 8001_0000h base + E0h offset = 8001_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA_BURST_TYPE		RSVD0[29:16]													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0[15:2]														NIBBLE_POS	TXCLK_DELAY_TYPE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_DDR_CTRL field descriptions

Field	Description
31–30 DMA_BURST_TYPE	The field controls the number of APB transfers per DMA request. 2'b00: select one APB transfer per DMA request 2'b01: select 4 APB transfers per DMA request. 2'b10: select 8 APB transfers per DMA request. 2'b11: reserved
29–2 RSVD0	Reserved
1 NIBBLE_POS	This bit only applies in MMC 4-bit DDR transfers. 0: The two high nibbles of two odd and even bytes are sent/received on a cycle(both edges), then the low nibbles of the odd and even bytes are sent/received on the next cycle. 1: Two nibbles of every bytes are sent/received on a cycle.
0 TXCLK_DELAY_TYPE	This field specifies two delay methods of delaying SCK related to SSP TX data to serve the purpose of obtaining better TX data setup time. Set this bit to 0 to choose the pre-set gate delay, which is approximately 5ns; set this bit to 1 to choose the SCK-coupled delay, which is 1/4 of SCK period.

17.10.16 SD/MMC DLL Control Register (HW_SSP_DLL_CTRL)

SD/MMC Delay Loop Lock Control Register.

This register provides programmability in DDR mode for data input timing and data formats.

Address: HW_SSP_DLL_CTRL – 8001_0000h base + F0h offset = 8001_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REF_UPDATE_INT				SLV_UPDATE_INT								RSVD1			
W													▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLV_OVERRIDE_VAL						SLV_ OVERRIDE	RSVD0	GATE_ UPDATE	SLV_DLY_TARGET				SLV_ FORCE_UPD	RESET	ENABLE
W								▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_DLL_CTRL field descriptions

Field	Description
31–28 REF_UPDATE_ INT	This field allows the user to add additional delay cycles to the DLL control loop (reference delay line control). By default, the DLL control loop shall update every two SSPCLK cycles. Programming this field results in a DLL control loop update interval of (2 + REF_UPDATE_INT) * SSPCLK. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that may effect the delay (such as voltage and temperature)
27–20 SLV_UPDATE_ INT	Setting a value greater than 0 in this field, shall over-ride the default slave delay-line update interval of 256 SSPCLK cycles. A value of 0 results in an update interval of 256 SSPCLK cycles (default setting). A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line will also update automatically when the reference DLL transitions to a locked state (from an un-locked state).
19–16 RSVD1	Reserved
15–10 SLV_ OVERRIDE_VAL	When SLV_OVERRIDE=1 This field is used to select 1 of 64 physical taps manually. A value of 0 selects tap 1, and a value of 0x3f selects tap 64.
9 SLV_OVERRIDE	Set this bit to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to tbe enabled using the ENABLE bit. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE=0
8 RSVD0	Reserved
7 GATE_UPDATE	Setting this bit to 1, forces the slave delay line not update

HW_SSP_DLL_CTRL field descriptions (continued)

Field	Description
6–3 SLV_DLY_ TARGET	The delay target for the SSP read clock is can be programmed in 1/16th increments of an SSPCLK half-period. So the input read-clock can be delayed relative input data from (SSPCLK/2)/16 to SSPCLK/2.
2 SLV_FORCE_ UPD	Setting this bit to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line shall update automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered).
1 RESET	Setting this bit to 1 force a reset on DLL. This will cause the DLL to lose lock and re-calibrate to detect an SSPCLK half period phase shift. This signal is used by the DLL as edge-sensitive, so in order to create a subsequent reset, RESET must be taken low and then asserted again.
0 ENABLE	Set this bit to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

17.10.17 SSP Status Register (HW_SSP_STATUS)

SSP Read Only Status Registers.

Various SSP status fields are provided in this register.

Address: HW_SSP_STATUS – 8001_0000h base + 100h offset = 8001_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	RSVD4	SD_PRESENT	CARD_DETECT	RSVD3					DMABURST	DMASENSE	DMATERM	DMAREQ	DMAEND	SDIO_IRQ	RESP_CRC_ERR
W																
Reset	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESP_ERR	RESP_TIMEOUT	DATA_CRC_ERR	TIMEOUT	RECV_TIMEOUT_STAT	RSVD2	FIFO_OVRFLW	FIFO_FULL	RSVD1		FIFO_EMPTY	FIFO_UNDRFLW	CMD_BUSY	DATA_BUSY	RSVD0	BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

HW_SSP_STATUS field descriptions

Field	Description
31 PRESENT	SSP Present Bit. 0: SSP is not present in this product. 1: SSP is present.
30 RSVD4	Reserved
29 SD_PRESENT	SD/MMC Controller Present bit. 0: SD/MMC controller is not present in this product. 1: SD/MMC controller is present.
28 CARD_DETECT	Reflects the state of the SSP_DETECT input pin.
27–23 RSVD3	Reserved
22 DMABURST	Reflects the state of the ssp_dmaburst output port.
21 DMASENSE	Reflects the state of the ssp_dmasense output port. It indicates a DMA error (Timeout or CRC) when asserted high at the end of a DMA command.
20 DMATERM	Reflects the state of the ssp_dmaterm output port. This is a toggle signal.
19 DMAREQ	Reflects the state of the ssp_dmareq output port. This is a toggle signal.
18 DMAEND	Reflects the state of the ssp_dmaend output port. This is a toggle signal.
17 SDIO_IRQ	SDIO IRQ has been detected.
16 RESP_CRC_ ERR	SD/MMC Response failed CRC check. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
15 RESP_ERR	SD/MMC Card Responsed to Command with an Error Condition. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
14 RESP_TIMEOUT	SD/MMC Card Expected Command Response not received within 64 CLK cycles. This indicates a card error, bad command, or command that failed CRC check. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
13 DATA_CRC_ ERR	Data CRC Error. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
12 TIMEOUT	SD/MMC - timeout counter expired before data bus was ready. This bit is cleared with soft reset or the rising edge of HW_SSP_CTRL0_RUN.
11 RECV_ TIMEOUT_STAT	Raw Receive Timeout Status. Indicates that no read has occurred to non-empty receive data FIFO for 128 cycles
10 RSVD2	Reserved
9 FIFO_OVRFLW	FIFO Overflow Interrupt.

HW_SSP_STATUS field descriptions (continued)

Field	Description
8 FIFO_FULL	FIFO FULL
7–6 RSVD1	Reserved
5 FIFO_EMPTY	FIFO Empty.
4 FIFO_UNDRFLW	FIFO Underflow has occurred.
3 CMD_BUSY	SD/MMC command controller is busy sending a command or receiving a response
2 DATA_BUSY	SD/MMC command controller is busy transferring data.
1 RSVD0	Reserved
0 BUSY	SSP State Machines are Busy.

17.10.18 SD/MMC DLL Status Register (HW_SSP_DLL_STS)

SSP Read Only Status Registers.

SSP DLL status fields are provided in this register.

Address: HW_SSP_DLL_STS – 8001_0000h base + 110h offset = 8001_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0[15:14]		REF_SEL						SLV_SEL						REF_ LOCK	SLV_ LOCK
W	▨		▨						▨						▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_DLL_STS field descriptions

Field	Description
31–14 RSVD0	Reserved

HW_SSP_DLL_STS field descriptions (continued)

Field	Description
13–8 REF_SEL	Reference delay line select status.
7–2 SLV_SEL	Slave delay line select status
1 REF_LOCK	Reference DLL lock status. This signifies that the DLL has detected and locked to a half-phase SSPCLK shift, allowing the slave delay-line to perform programmed clock delays.
0 SLV_LOCK	Slave delay-line lock status. This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value.

17.10.19 SSP Debug Register (HW_SSP_DEBUG)

SSP Read Only Debug Registers.

Debug Register provide access to State machines.

Address: HW_SSP_DEBUG – 8001_0000h base + 120h offset = 8001_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATACRC_ERR				DATA_ STALL	DAT_SM			RSVD1				CMD_ OE	DMA_SM		
W	▨				▨	▨			▨				▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MMC_SM				CMD_SM		SSP_ CMD	SSP_ RESP	SSP_RXD							
W	▨				▨		▨	▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_DEBUG field descriptions

Field	Description
31–28 DATACRC_ERR	Data CRC error
27 DATA_STALL	MMC mode: FIFO transfer not ready
26–24 DAT_SM	MMC dataxfer state machine 0x0 **DSM_IDLE** — 0x2 **DSM_WORD** — 0x3 **DSM_CRC1** — 0x4 **DSM_CRC2** — 0x5 **DSM_END** —

HW_SSP_DEBUG field descriptions (continued)

Field	Description
23–20 RSVD1	Reserved
19 CMD_OE	Enable for SSP_CMD
18–16 DMA_SM	DMA state machine 0x0 **DMA_IDLE** — 0x1 **DMA_DMAREQ** — 0x2 **DMA_DMAACK** — 0x3 **DMA_STALL** — 0x4 **DMA_BUSY** — 0x5 **DMA_DONE** — 0x6 **DMA_COUNT** —
15–12 MMC_SM	MMC_state machine 0x0 **MMC_IDLE** — 0x1 **MMC_CMD** — 0x2 **MMC_TRC** — 0x3 **MMC_RESP** — 0x4 **MMC_RPRX** — 0x5 **MMC_TX** — 0x6 **MMC_CTOK** — 0x7 **MMC_RX** — 0x8 **MMC_CCS** — 0x9 **MMC_PUP** — 0xA **MMC_WAIT** —
11–10 CMD_SM	MMC command_state machine 0x0 **CSM_IDLE** — 0x1 **CSM_INDEX** — 0x2 **CSM_ARG** — 0x3 **CSM_CRC** —
9 SSP_CMD	SSP_CMD
8 SSP_RESP	SSP_RESP
7–0 SSP_RXD	SSP_RXD.

17.10.20 SSP Version Register (HW_SSP_VERSION)

This register reflects the version number for the SSP.

Address: HW_SSP_VERSION – 8001_0000h base + 130h offset = 8001_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SSP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 18
Boundary Scan Interface

18.1 Boundary Scan Interface

This chapter describes the boundary scan feature that has been implemented on the i.MX28. This feature is provided to enable board level testing.

There are five pins on the device which are used to implement the IEEE 1149.1 boundary scan protocol. In order to make use of the boundary scan functionality, an extra pin called DEBUG must be tied to low. The following table shows the pin information that is used to implement boundary scan.

Table 18-1. Boundary Scan Pin Functions

Name of Pin	Function	Direction
TCK	This is the clock that is used to serially shift data in to the part.	Input
TMS	This signal is used to select the test mode.	Input
TDI	This is the serial data that is used to shift in data into the boundary scan register. It is also be used to shift in instructions into the instruction register and data into the I/O configuration register.	Input
TRST_N	This is the reset signal for the block.	Input
TDO	This is the output of the boundary scan chain.	Output
DEBUG	This is the selection for the JTAG interface. DEBUG=0: JTAG interface works for boundary scan. DEBUG=1: JTAG interface works for ARM debugging. This chapter does not cover ARM Debugging.	Input

There are six instructions implemented in the boundary scan design. The following table describes these instructions:

Table 18-2. Boundary Scan Instruction Set

Instruction	Code	Description
EXTEST	0000	This instruction places the IC into an external boundary-test mode and selects the boundary-scan register to be connected between TDI and TDO. During this instruction, the boundary-scan register is accessed to drive test data off-chip through the boundary outputs and receive test data off-chip through the boundary inputs.
SAMPLE/PRELOAD	0001	This instruction takes the data on the I/O pad and latches it into the boundary scan register.
BYPASS	1111	This instruction is used to bypass the chip during board testing.
IDCODE	0010	This instruction allows a code to be serially read from the component. The Chip level JTAG ID Code of i.MX28 is 0x0882401D.
HIGHZ	0011	This instruction places the component in a state, in which all of its system logic outputs are placed in an inactive drive state.
IOCFG	0101	This user-defined instruction is used to program the voltage level of pins.

There is a special instruction which is called IOCFG that is implemented to load the I/O configuration register. The I/O configuration register is used to program the voltage levels of pins. It has 126 bits, corresponding to the 126 GPIO pins in the device. If there is a 0 in a certain bit location, then the GPIO corresponding to that location will be programmed to a 1.8 V level. The default value, which is 1, corresponds to a pin being programmed to a 3.3 V level. This register is provided so that the pins that are connected to 1.8 V peripherals can be programmed to the correct voltage level before running the board level tests.

The customer will be provided a BSDL file which is in a format that is used by the board tester. This file will detail the sequence of the pad connection in the boundary scan chain, and the properties of the pads. This is used by the board level tester to develop the tests for connectivity of the chip.

Chapter 19
Digital Control (DIGCTL) and On-Chip RAM

19.1 Overview

The digital control block provides overall control of various items within the top digital block of the chip, including:

- Default first-level page table (DFLPT) controls
- HCLK performance counter
- Free-running microseconds counter
- Entropy control
- BIST controls for ARM Core and On-Chip RAM
- Chip Revision register
- USB loop back congtrol
- Other miscellaneous controls

The detailed diagram is shown below.

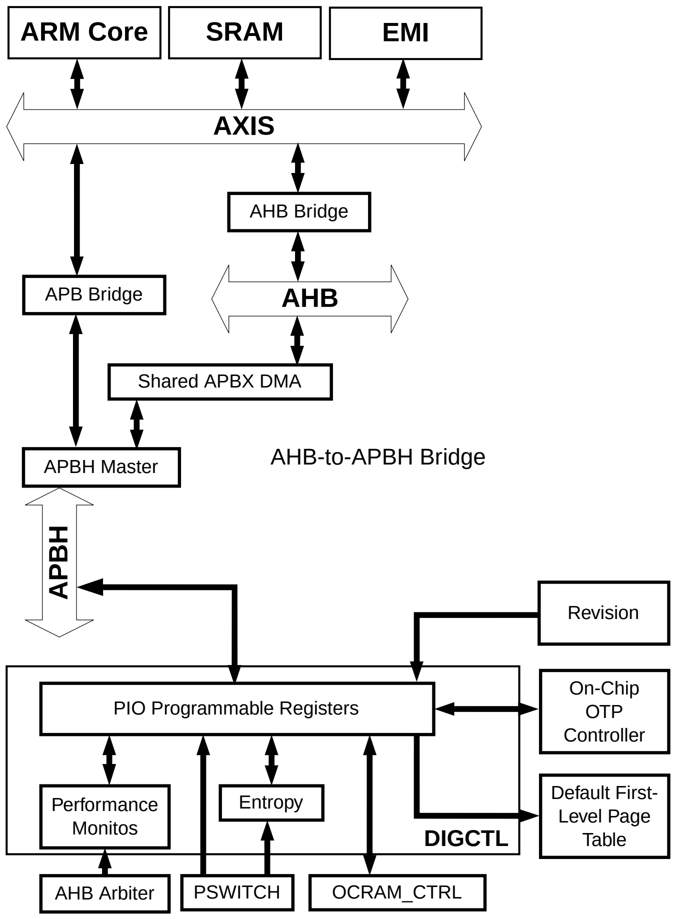


Figure 19-1. Digital Control (DIGCTL) Block Diagram

19.2 SRAM Controls

The on-chip 128 KB RAM is comprised of four physical banks of 8Kx32-bit each. The architecture requires a 4-port, word interleaved topology to maximize the performance in a multi-layer bus system. A diagrammatic representation of the word-interleaving is shown below.

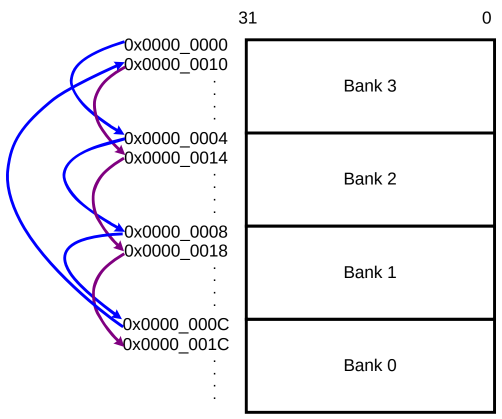


Figure 19-2. On-Chip RAM Partitioning

A 32-bit AHB address is arranged as shown below.

Table 19-1. On-Chip RAM Address Bits

AHB ADDR BITS	USAGE	DESCRIPTION
16:4	Address	Selects one of 8 K words in a bank.
3:2	Bank Address	Selects 1 of 4 physical 32 KB banks.
1:0	Byte Address	Selects/masks out specific bytes within a word.

The on-chip RAM and controller work at full CLK_H frequency without additional wait states. There is no memory repair for the SRAM.

19.3 Miscellaneous Controls

The digital control block also contains a number of other miscellaneous functions, as detailed in this section.

19.3.1 Performance Monitoring

The digital control block contains several registers for system bus performance monitoring, including HW_DIGCTL_HCLKCOUNT, which counts HCLK rising edges. This register counts at a variable rate as HW_CLKCTRL_HBUS_AUTO_SLOW_ MODE is enabled.

In addition, there exists a performance monitoring register for each AHB layer (L0–L3). The HW_DIGCTL_L(n)_AHB_DATA_STALLED and HW_DIGCTL_L(n)_AHB_ACTIVE_CYCLES registers can be used to measure AHB bus utilization. The Stalled register counts all cycles in which any device has an outstanding and unfulfilled bus operation in flight. The Active Cycles register counts the number of data transfer cycles. Subtract cycles from stalls to determine under utilized bus cycles. These counters can be used to tune the performance of the HCLK frequency for specific activities. In addition, these monitors can be forced to focus on specific masters (which connect to that layer). See the HW_DIGCTL_AHB_STATS_SELECT bit description for details.

19.3.2 High-Entropy PRN Seed

A 32-bit entropy register begins running a pseudo-random number algorithm from the time reset is removed until the PSWITCH is released by the user. This high-entropy value can be used as the seed for other pseudo-random number generators.

19.3.3 Write-Once Register

A 32-bit write-once register holds a runtime-derived locked seed. Once written, it cannot be changed until the next chip wide reset event. The contents of this register are frequently derived from the entropy register.

19.3.4 Microseconds Counter

A 32-bit free-running microseconds counter provides fine-grain real-time control. Its period is determined by dividing the 24.0-MHz crystal oscillator by 24. Therefore, its frequency does not change as HCLK, XCLK, and the processor clock frequency are changed.

19.4 Programmable Registers

DIGCTL Register Format Summary

HW_DIGCTL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_C000	DIGCTL Control Register (HW_DIGCTL_CTRL)	32	R/W	0001_0004h	19.4.1/1361
8001_C010	DIGCTL Status Register (HW_DIGCTL_STATUS)	32	R	FF00_0000h	19.4.2/1364
8001_C020	Free-Running HCLK Counter Register (HW_DIGCTL_HCLKCOUNT)	32	R	0000_0000h	19.4.3/1366
8001_C030	On-Chip RAM Control Register (HW_DIGCTL_RAMCTRL)	32	R/W	0000_0000h	19.4.4/1367
8001_C040	EMI Status Register (HW_DIGCTL_EMI_STATUS)	32	R	0000_0000h	19.4.5/1368
8001_C050	On-Chip Memories Read Margin Register (HW_DIGCTL_READ_MARGIN)	32	R/W	0000_0002h	19.4.6/1369

HW_DIGCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_C060	Software Write-Once Register (HW_DIGCTL_WRITEONCE)	32	R/W	A5A5_A5A5h	19.4.7/1369
8001_C070	BIST Control Register (HW_DIGCTL_BIST_CTL)	32	R/W	4000_0000h	19.4.8/1370
8001_C080	DIGCTL Status Register (HW_DIGCTL_BIST_STATUS)	32	R	0000_0000h	19.4.9/1372
8001_C090	Entropy Register (HW_DIGCTL_ENTROPY)	32	R	0000_0000h	19.4.10/1375
8001_C0A0	Entropy Latched Register (HW_DIGCTL_ENTROPY_LATCHED)	32	R	0000_0000h	19.4.11/1376
8001_C0C0	Digital Control Microseconds Counter Register (HW_DIGCTL_MICROSECONDS)	32	R/W	0000_0000h	19.4.12/1376
8001_C0D0	Digital Control Debug Read Test Register (HW_DIGCTL_DBGRD)	32	R	789A_BCDEh	19.4.13/1377
8001_C0E0	Digital Control Debug Register (HW_DIGCTL_DBG)	32	R	8765_4321h	19.4.14/1377
8001_C100	USB LOOP BACK (HW_DIGCTL_USB_LOOPBACK)	32	R/W	0000_0000h	19.4.15/1378
8001_C110	SRAM Status Register 0 (HW_DIGCTL_OCRAM_STATUS0)	32	R	0000_0000h	19.4.16/1380
8001_C120	SRAM Status Register 1 (HW_DIGCTL_OCRAM_STATUS1)	32	R	0000_0000h	19.4.17/1381
8001_C130	SRAM Status Register 2 (HW_DIGCTL_OCRAM_STATUS2)	32	R	0000_0000h	19.4.18/1381
8001_C140	SRAM Status Register 3 (HW_DIGCTL_OCRAM_STATUS3)	32	R	0000_0000h	19.4.19/1382
8001_C150	SRAM Status Register 4 (HW_DIGCTL_OCRAM_STATUS4)	32	R	0000_0000h	19.4.20/1383
8001_C160	SRAM Status Register 5 (HW_DIGCTL_OCRAM_STATUS5)	32	R	0000_0000h	19.4.21/1384
8001_C170	SRAM Status Register 6 (HW_DIGCTL_OCRAM_STATUS6)	32	R	0000_0000h	19.4.22/1384
8001_C180	SRAM Status Register 7 (HW_DIGCTL_OCRAM_STATUS7)	32	R	0000_0000h	19.4.23/1385
8001_C190	SRAM Status Register 8 (HW_DIGCTL_OCRAM_STATUS8)	32	R	0000_0000h	19.4.24/1386
8001_C1A0	SRAM Status Register 9 (HW_DIGCTL_OCRAM_STATUS9)	32	R	0000_0000h	19.4.25/1386
8001_C1B0	SRAM Status Register 10 (HW_DIGCTL_OCRAM_STATUS10)	32	R	0000_0000h	19.4.26/1387
8001_C1C0	SRAM Status Register 11 (HW_DIGCTL_OCRAM_STATUS11)	32	R	0000_0000h	19.4.27/1388
8001_C1D0	SRAM Status Register 12 (HW_DIGCTL_OCRAM_STATUS12)	32	R	0000_0000h	19.4.28/1389
8001_C1E0	SRAM Status Register 13 (HW_DIGCTL_OCRAM_STATUS13)	32	R	0000_0000h	19.4.29/1390
8001_C280	Digital Control Scratch Register 0 (HW_DIGCTL_SCRATCH0)	32	R/W	0000_0000h	19.4.30/1391
8001_C290	Digital Control Scratch Register 1 (HW_DIGCTL_SCRATCH1)	32	R/W	0000_0000h	19.4.31/1392
8001_C2A0	Digital Control ARM Cache Register (HW_DIGCTL_ARMCACHE)	32	R/W	0000_0000h	19.4.32/1392
8001_C2B0	Debug Trap Control and Status for AHB Layer 0 and 3 (HW_DIGCTL_DEBUG_TRAP)	32	R/W	0000_0000h	19.4.33/1393

HW_DIGCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_C2C0	Debug Trap Range Low Address for AHB Layer 0 (HW_DIGCTL_DEBUG_TRAP_L0_ADDR_LOW)	32	R/W	0000_0000h	19.4.34/1395
8001_C2D0	Debug Trap Range High Address for AHB Layer 0 (HW_DIGCTL_DEBUG_TRAP_L0_ADDR_HIGH)	32	R/W	0000_0000h	19.4.35/1396
8001_C2E0	Debug Trap Range Low Address for AHB Layer 3 (HW_DIGCTL_DEBUG_TRAP_L3_ADDR_LOW)	32	R/W	0000_0000h	19.4.36/1396
8001_C2F0	Debug Trap Range High Address for AHB Layer 3 (HW_DIGCTL_DEBUG_TRAP_L3_ADDR_HIGH)	32	R/W	0000_0000h	19.4.37/1397
8001_C300	Freescale Copyright Identifier Register (HW_DIGCTL_FSL)	32	R	6565_7246h	19.4.38/1397
8001_C310	Digital Control Chip Revision Register (HW_DIGCTL_CHIPID)	32	R	2800_0000h	19.4.39/1398
8001_C330	AHB Statistics Control Register (HW_DIGCTL_AHB_STATS_SELECT)	32	R/W	0000_0000h	19.4.40/1399
8001_C370	AHB Layer 1 Transfer Count Register (HW_DIGCTL_L1_AHB_ACTIVE_CYCLES)	32	R/W	0000_0000h	19.4.41/1400
8001_C380	AHB Layer 1 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L1_AHB_DATA_STALLED)	32	R/W	0000_0000h	19.4.42/1401
8001_C390	AHB Layer 1 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L1_AHB_DATA_CYCLES)	32	R/W	0000_0000h	19.4.43/1401
8001_C3A0	AHB Layer 2 Transfer Count Register (HW_DIGCTL_L2_AHB_ACTIVE_CYCLES)	32	R/W	0000_0000h	19.4.44/1402
8001_C3B0	AHB Layer 2 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L2_AHB_DATA_STALLED)	32	R/W	0000_0000h	19.4.45/1403
8001_C3C0	AHB Layer 2 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L2_AHB_DATA_CYCLES)	32	R/W	0000_0000h	19.4.46/1403
8001_C3D0	AHB Layer 3 Transfer Count Register (HW_DIGCTL_L3_AHB_ACTIVE_CYCLES)	32	R/W	0000_0000h	19.4.47/1404
8001_C3E0	AHB Layer 3 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L3_AHB_DATA_STALLED)	32	R/W	0000_0000h	19.4.48/1405
8001_C3F0	AHB Layer 3 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L3_AHB_DATA_CYCLES)	32	R/W	0000_0000h	19.4.49/1406
8001_C500	Default First Level Page Table Movable PTE Locator 0 (HW_DIGCTL_MPTE0_LOC)	32	R/W	0000_0000h	19.4.50/1406
8001_C510	Default First-Level Page Table Movable PTE Locator 1 (HW_DIGCTL_MPTE1_LOC)	32	R/W	0000_0001h	19.4.51/1407
8001_C520	Default First-Level Page Table Movable PTE Locator 2 (HW_DIGCTL_MPTE2_LOC)	32	R/W	0000_0002h	19.4.52/1408
8001_C530	Default First-Level Page Table Movable PTE Locator 3 (HW_DIGCTL_MPTE3_LOC)	32	R/W	0000_0003h	19.4.53/1409
8001_C540	Default First-Level Page Table Movable PTE Locator 4 (HW_DIGCTL_MPTE4_LOC)	32	R/W	0000_0004h	19.4.54/1410

HW_DIGCTL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8001_C550	Default First-Level Page Table Movable PTE Locator 5 (HW_DIGCTL_MPTE5_LOC)	32	R/W	0000_0005h	19.4.55/1411
8001_C560	Default First-Level Page Table Movable PTE Locator 6 (HW_DIGCTL_MPTE6_LOC)	32	R/W	0000_0006h	19.4.56/1411
8001_C570	Default First-Level Page Table Movable PTE Locator 7 (HW_DIGCTL_MPTE7_LOC)	32	R/W	0000_0007h	19.4.57/1412
8001_C580	Default First-Level Page Table Movable PTE Locator 8 (HW_DIGCTL_MPTE8_LOC)	32	R/W	0000_0008h	19.4.58/1413
8001_C590	Default First-Level Page Table Movable PTE Locator 9 (HW_DIGCTL_MPTE9_LOC)	32	R/W	0000_0009h	19.4.59/1414
8001_C5A0	Default First-Level Page Table Movable PTE Locator 10 (HW_DIGCTL_MPTE10_LOC)	32	R/W	0000_000Ah	19.4.60/1415
8001_C5B0	Default First-Level Page Table Movable PTE Locator 11 (HW_DIGCTL_MPTE11_LOC)	32	R/W	0000_000Bh	19.4.61/1416
8001_C5C0	Default First-Level Page Table Movable PTE Locator 12 (HW_DIGCTL_MPTE12_LOC)	32	R/W	0000_000Ch	19.4.62/1416
8001_C5D0	Default First-Level Page Table Movable PTE Locator 13 (HW_DIGCTL_MPTE13_LOC)	32	R/W	0000_000Dh	19.4.63/1417
8001_C5E0	Default First-Level Page Table Movable PTE Locator 14 (HW_DIGCTL_MPTE14_LOC)	32	R/W	0000_000Eh	19.4.64/1418
8001_C5F0	Default First-Level Page Table Movable PTE Locator 15 (HW_DIGCTL_MPTE15_LOC)	32	R/W	0000_000Fh	19.4.65/1419

19.4.1 DIGCTL Control Register (HW_DIGCTL_CTRL)

The DIGCTL Control Register provides overall control of various functions throughout the digital portion of the chip.

HW_DIGCTL_CTRL: 0x000

HW_DIGCTL_CTRL_SET: 0x004

HW_DIGCTL_CTRL_CLR: 0x008

HW_DIGCTL_CTRL_TOG: 0x00C

EXAMPLE

```
HW_DIGCTL_CTRL_CLR(BM_DIGCTL_CTRL_USB_CLKGATE);  // enable USB clock
```

Address: HW_DIGCTL_CTRL – 8001_C000h base + 0h offset = 8001_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3	XTAL24M_GATE	RSVD2					USB1_ OVERCURRENT_ ENABLE	USB0_ OVERCURRENT_ ENABLE	USB1_ OVERCURRENT_POL	USB0_ OVERCURRENT_POL	USB1_ TESTMODE	USB0_ TESTMODE	ANALOG_ TESTMODE	DIGITAL_ TESTMODE	USB1_CLKGATE
W	▨		▨	▨	▨	▨	▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SAIF_LOOPBACK	DUART_ LOOPBACK	AUART01_ LOOPBACK	RSVD1	SAIF_ CLKMUX_ SEL		RSVD0						DEBUG_DISABLE	USB0_CLKGATE	JTAG_SHIELD	LATCH_ ENTROPY
W				▨			▨	▨	▨	▨	▨	▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_DIGCTL_CTRL field descriptions

Field	Description
31 RSVD3	Reserved.
30 XTAL24M_GATE	If set to 1, disable the Digital Control Microseconds counter, STRB1MHZ. If set to 0, enable the Digital Control Microseconds counter..
29–25 RSVD2	Reserved.
24 USB1_ OVERCURRENT_ ENABLE	0 - Disable the overcurrent detection logic (default) 1 - Enable the overcurrent detection logic
23 USB0_ OVERCURRENT_ ENABLE	0 - Disable the overcurrent detection logic (default) 1 - Enable the overcurrent detection logic
22 USB1_ OVERCURRENT_ POL	0 - The external Over Current indicator signal is high active.(default) 1 - The external Over Current indicator signal is low active.
21 USB0_ OVERCURRENT_ POL	0 - The external Over Current indicator signal is high active.(default) 1 - The external Over Current indicator signal is low active.
20 USB1_ TESTMODE	Set this bit to get into USB1 test mode
19 USB0_ TESTMODE	Set this bit to get into USB0 test mode

HW_DIGCTL_CTRL field descriptions (continued)

Field	Description
18 ANALOG_ TESTMODE	Set this bit to get into analog test mode
17 DIGITAL_ TESTMODE	Set this bit to get into digital test mode
16 USB1_CLKGATE	This bit must be cleared to 0 for normal operation of the USB controller. When set to 1, it gates off the clocks to the USB controller. USB_CLKGATE can be set during suspend to gate the USB clock during suspend. If this is gated, then the USB controller Reset Received bit (HW_USBCTRL_USBSTS_URI) should be not be polled for reset during suspend; use the HW_USBPHY_CTRL_RESUME_IRQ bit instead. This bit can be hardware auto cleared when wakeup from suspend happens 0x0 **RUN** — Allow USB to operate normally. 0x1 **NO_CLKS** — Do not clock USB gates in order to minimize power consumption.
15 SAIF_LOOPBACK	Set this bit to one to loop SAIF0 to SAIF1 and SAIF1 to SAIF0. To use SAIF loopback the user must configure one SAIF for transmit and the other for receive. Because this bit connects SAIF0's output to SAIF1's input, and SAIF1's output to SAIF0's input it doesn't matter which of the two ports is configured for TX and the other for RX, either configuration will produce an internal TX to RX loopback. 0x0 **NORMAL** — No loopback. 0x1 **LOOPIT** — Loop SAIF0 and SAIF1 back to each other.
14 DUART_ LOOPBACK	Set this bit to one to loop the debug UART's RX and TX signals back on themselves in a null modem configuration. 0x0 **NORMAL** — No loopback. 0x1 **LOOPIT** — Loop debug UART TX and RX together.
13 AUART01_ LOOPBACK	Set this bit to one to loop application UARTs 0 and 1 back on themselves in a null modem configuration. 0x0 **NORMAL** — No loopback. 0x1 **LOOPIT** — Loop application UART 0 and 1 together.
12 RSVD1	Reserved.
11–10 SAIF_CLKMUX_ SEL	Selects the source of the SAIF0 and SAIF1 input bit clock (BITCLK), and input left/right sample clock (LRCLK). 0x0 **DIRECT** — SAIF0 clock pins selected for SAIF0 input clocks, and SAIF1 clock pins selected for SAIF1 input clocks. 0x1 **CROSSINPUT** — SAIF1 clock inputs selected for SAIF0 input clocks, and SAIF0 clock inputs selected for SAIF1 input clocks. This is the cross input mode. 0x2 **CLKSRCSAIF0PIN** — SAIF0 clock pin selected for both SAIF0 and SAIF1 input clocks. 0x3 **CLKSRCSAIF1PIN** — SAIF1 clock pin selected for both SAIF0 and SAIF1 input clocks.
9–4 RSVD0	Reserved.
3 DEBUG_DISABLE	Set this bit to disable the ARM core's debug logic (for power savings). This bit must remain 0 following power-on reset for normal JTAG debugger operation of the ARM core. When set to 1, it gates off the clocks to the ARM core's debug logic. Once this bit is set, the part must undergo a power-on reset to re-enable debug operation. Manually clearing this bit through a write after it has been set produces unknown results.

HW_DIGCTL_CTRL field descriptions (continued)

Field	Description
2 USB0_CLKGATE	This bit must be cleared to 0 for normal operation of the USB controller. When set to 1, it gates off the clocks to the USB controller. USB_CLKGATE can be set during suspend to gate the USB clock during suspend. If this is gated, then the USB controller Reset Received bit (HW_USBCTRL_USBSTS_URI) should be not be polled for reset during suspend; use the HW_USBPHY_CTRL_RESUME_IRQ bit instead. This bit can be hardware auto cleared when wakeup from suspend happens 0x0 **RUN** — Allow USB to operate normally. 0x1 **NO_CLKS** — Do not clock USB gates in order to minimize power consumption.
1 JTAG_SHIELD	0 = The JTAG debugger is enabled. 1 = The JTAG debugger is disabled. 0x0 **NORMAL** — JTAG debugger enabled. 0x1 **SHIELDS_UP** — JTAG debugger disabled.
0 LATCH_ ENTROPY	Setting this bit latches the current value of the entropy register into HW_DIGCTL_ENTROPY_VALUE. This can be used get a stable value on players that do not deassert the PSWITCH while powered up.

19.4.2 DIGCTL Status Register (HW_DIGCTL_STATUS)

The DIGCTL Status Register provides a read-only view to various input conditions and internal states.

HW_DIGCTL_STATUS: 0x010

HW_DIGCTL_STATUS_SET: 0x014

HW_DIGCTL_STATUS_CLR: 0x018

HW_DIGCTL_STATUS_TOG: 0x01C

EXAMPLE

```
if(HW_DIGCTL_STATUS.PACKAGE_TYPE)
{
  // do 100-pin package things
}
```

Address: HW_DIGCTL_STATUS – 8001_C000h base + 10h offset = 8001_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USB0_HS_PRESENT	USB0_OTG_PRESENT	USB0_HOST_PRESENT	USB0_DEVICE_PRESENT	USB1_HS_PRESENT	USB1_OTG_PRESENT	USB1_HOST_PRESENT	USB1_DEVICE_PRESENT	RSVD0[23:16]							
W																
Reset	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0[15:5]											JTAG_IN_USE	PACKAGE_TYPE			WRITTEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_STATUS field descriptions

Field	Description
31 USB0_HS_ PRESENT	This read-only bit returns a 1 when USB high-speed mode is present.
30 USB0_OTG_ PRESENT	This read-only bit returns a 1 when USB on-the-go (OTG) functionality is present.
29 USB0_HOST_ PRESENT	This read-only bit returns a 1 when USB host functionality is present.
28 USB0_DEVICE_ PRESENT	This read-only bit returns a 1 when USB device functionality is present.
27 USB1_HS_ PRESENT	This read-only bit returns a 1 when USB high-speed mode is present.
26 USB1_OTG_ PRESENT	This read-only bit returns a 1 when USB on-the-go (OTG) functionality is present.
25 USB1_HOST_ PRESENT	This read-only bit returns a 1 when USB host functionality is present.
24 USB1_DEVICE_ PRESENT	This read-only bit returns a 1 when USB device functionality is present.

HW_DIGCTL_STATUS field descriptions (continued)

Field	Description
23–5 RSVD0	Reserved.
4 JTAG_IN_USE	This read-only bit is a 1 if JTAG debugger usage has been detected.
3–1 PACKAGE_TYPE	This read-only bit field returns the pin count and package type. 000=289BGA, 001-111=Reserved.
0 WRITTEN	Set to 1 by any successful write to the HW_DIGCTL_WRITEONCE register.

19.4.3 Free-Running HCLK Counter Register (HW_DIGCTL_HCLKCOUNT)

The Free-Running HCLK Counter Register is available for performance metrics. This counter increments once per HCLK rising edge.

HW_DIGCTL_HCLKCOUNT: 0x020

HW_DIGCTL_HCLKCOUNT_SET: 0x024

HW_DIGCTL_HCLKCOUNT_CLR: 0x028

HW_DIGCTL_HCLKCOUNT_TOG: 0x02C

EXAMPLE

```
    StartTime = HW_DIGCTL_HCLKCOUNT;
    // Do something you want timed here
    EndTime = HW_DIGCTL_HCLKCOUNT;
   Duration  = EndTime - StartTime; // make sure to handle rollover in a real application
```

Address: HW_DIGCTL_HCLKCOUNT – 8001_C000h base + 20h offset = 8001_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_HCLKCOUNT field descriptions

Field	Description
31–0 COUNT	This counter counts up from reset using HCLK. The count is valid for HCLK frequencies greater than 2 MHz.

19.4.4 On-Chip RAM Control Register (HW_DIGCTL_RAMCTRL)

The On-Chip RAM Control Register holds on-chip SRAM control bit fields. This register controls various parts of the on-chip RAM, including the speed select configuration.

HW_DIGCTL_RAMCTRL: 0x030

HW_DIGCTL_RAMCTRL_SET: 0x034

HW_DIGCTL_RAMCTRL_CLR: 0x038

HW_DIGCTL_RAMCTRL_TOG: 0x03C

EXAMPLE

```
        HW_DIGCTL_RAMCTRL_SET(BM_DIGCTL_RAMCTRL_REPAIR_TRANSMIT);   // Start the efuse state
 machine
```

Address: HW_DIGCTL_RAMCTRL – 8001_C000h base + 30h offset = 8001_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2 [15:15]	RSVD1	DEBUG_ ENABLE	DEBUG_CODE					RSVD0							
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_RAMCTRL field descriptions

Field	Description
31–15 RSVD2	Reserved.
14 RSVD1	Reserved, always set to zero.
13 DEBUG_ENABLE	Debug enable for on chip sram. Set DEBUG_CODE field for different debug mode.
12–8 DEBUG_CODE	Debug code for 8x32 OCRAM instances. Recommended value is 0x0. Used to change different data access time. These bits are ignored if debug enable is low. 0x0 **NORMAL** — Normal functional mode. 0x4 **DELAY1** — Normal functional mode. 0x5 **DELAY2** — Normal functional mode.

HW_DIGCTL_RAMCTRL field descriptions (continued)

Field	Description
	0x6 **DELAY3** — Normal functional mode. 0x7 **DELAY4** — Normal functional mode.
7–0 RSVD0	Reserved.

19.4.5 EMI Status Register (HW_DIGCTL_EMI_STATUS)

The EMI Status Register indicates the low power mode of the EMI and provides state data that is not available in the EMI controller.

HW_DIGCTL_EMI_STATUS: 0x040

HW_DIGCTL_EMI_STATUS_SET: 0x044

HW_DIGCTL_EMI_STATUS_CLR: 0x048

HW_DIGCTL_EMI_STATUS_TOG: 0x04C

EXAMPLE

NA

Address: HW_DIGCTL_EMI_STATUS – 8001_C000h base + 40h offset = 8001_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																											POWER_MODE				
W	▨																											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_EMI_STATUS field descriptions

Field	Description
31–5 RSVD0	Reserved.
4–0 POWER_MODE	A detail description of the low power modes can be found in the EMI documentation. This register provides a port with which to read the power mode status since this capability is not inherent in the block itself. 0x2 **PM4** — Power mode 4 is active. 0x4 **PM3** — Power mode 3 is active. 0x8 **PM2** — Power mode 2 is active. 0x10 **PM1** — Power mode 1 is active. 0x0 **NORMAL** — Normal operation, low power modes are not active.

19.4.6 On-Chip Memories Read Margin Register (HW_DIGCTL_READ_MARGIN)

The On-Chip Memories Read Margin Register provide speed select settings for numerous memories.

HW_DIGCTL_READ_MARGIN: 0x050

HW_DIGCTL_READ_MARGIN_SET: 0x054

HW_DIGCTL_READ_MARGIN_CLR: 0x058

HW_DIGCTL_READ_MARGIN_TOG: 0x05C

This register controls various parts of the on-chip RAM and ROM, including the sense amp configuration.

Address: HW_DIGCTL_READ_MARGIN – 8001_C000h base + 50h offset = 8001_C050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																												ROM			
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

HW_DIGCTL_READ_MARGIN field descriptions

Field	Description
31–4 RSVD0	Reserved.
3–0 ROM	This field is used for setting the read margin for the On-Chip ROM. It programs the sense amp differential setting and allows the trade off between speed and robustness. This field should not be changed unless instructed by Freescale.

19.4.7 Software Write-Once Register (HW_DIGCTL_WRITEONCE)

The Software Write-Once Register hold the value used in software certification management.

This register is used to hold a portion of a certificate that is not mutable after software initialization.

EXAMPLE

```
HW_DIGCTL_WRITEONCE.U = my_certificate;
```

Address: HW_DIGCTL_WRITEONCE – 8001_C000h base + 60h offset = 8001_C060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1

HW_DIGCTL_WRITEONCE field descriptions

Field	Description
31–0 BITS	This field can be written only one time. The contents are not used by hardware.

19.4.8 BIST Control Register (HW_DIGCTL_BIST_CTL)

The BIST Control Register provides the BIST control of all the blocks.

HW_DIGCTL_BIST_CTL: 0x070

HW_DIGCTL_BIST_CTL_SET: 0x074

HW_DIGCTL_BIST_CTL_CLR: 0x078

HW_DIGCTL_BIST_CTL_TOG: 0x07C

EXAMPLE

```
HW_DIGCTL_BIST_CTL();  //
```

Address: HW_DIGCTL_BIST_CTL – 8001_C000h base + 70h offset = 8001_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BIST_TESTMODE	BIST_RESETN	BIST_DEBUGZ	BIST_CHECKB	BIST_RESUME	RSVD0[26:16]										
W						▨										
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0 [15:15]	OCRAM_BIST_ RETENTION	OCRAM_ BIST_PASS	OCRAM_ BIST_FAIL	OCRAM_ BIST_DONE	OCRAM_BIST_START	PXP_BIST_START	LCDIF_BIST_START	DCP_BIST_START	ENET_BIST_START	USB1_BIST_START	USB0_BIST_START	DMA1_BIST_START	DMA0_BIST_START	CACHE_BIST_START	CAN_BIST_START
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_BIST_CTL field descriptions

Field	Description
31 BIST_ TESTMODE	bist test mode, open clock gatings
30 BIST_RESETN	bist reset.
29 BIST_DEBUGZ	enable bist debug
28 BIST_CHECKB	use checkboard algorithm with retention
27 BIST_RESUME	resume with bist for retention test
26–15 RSVD0	Reserved.
14 OCRAM_BIST_ RETENTION	This read-only bit is a 1 if the OCRAM BIST test has been enabled waiting for a resume.
13 OCRAM_BIST_ PASS	This read-only bit is a 1 if the OCRAM BIST pass.
12 OCRAM_BIST_ FAIL	This read-only bit is a 1 if the OCRAM BIST test returns a failure.
11 OCRAM_BIST_ DONE	This read-only bit is a 1 if the OCRAM BIST test has completed.
10 OCRAM_BIST_ START	Set this bit to start the OCRAM BIST.
9 PXP_BIST_ START	Set this bit to start the PXP BIST.
8 LCDIF_BIST_ START	Set this bit to start the LCDIF BIST.

HW_DIGCTL_BIST_CTL field descriptions (continued)

Field	Description
7 DCP_BIST_ START	Set this bit to start the DCP BIST.
6 ENET_BIST_ START	Set this bit to start the ENET BIST.
5 USB1_BIST_ START	Set this bit to start the USB1 BIST.
4 USB0_BIST_ START	Set this bit to start the USB0 BIST.
3 DMA1_BIST_ START	Set this bit to start the DMA1 BIST.
2 DMA0_BIST_ START	Set this bit to start the DMA0 BIST.
1 CACHE_BIST_ START	Set this bit to start the CACHE BIST.
0 CAN_BIST_ START	Set this bit to start the CAN BIST.

19.4.9 DIGCTL Status Register (HW_DIGCTL_BIST_STATUS)

The DIGCTL Status Register reports status for the digital control block.

HW_DIGCTL_BIST_STATUS: 0x080

HW_DIGCTL_BIST_STATUS_SET: 0x084

HW_DIGCTL_BIST_STATUS_CLR: 0x088

HW_DIGCTL_BIST_STATUS_TOG: 0x08C

EXAMPLE

```
HW_DIGCTL_BIST_STATUS();
```

Address: HW_DIGCTL_BIST_STATUS – 8001_C000h base + 80h offset = 8001_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0		PXP_BIST_RETENTION	LCDIF_BIST_RETENTION	DCP_BIST_RETENTION	ENET_BIST_RETENTION	USB1_BIST_RETENTION	USB0_BIST_RETENTION	DMA1_BIST_RETENTION	DMA0_BIST_RETENTION	CACHE_BIST_RETENTION	CAN_BIST_RETENTION	PXP_BIST_FAIL	LCDIF_BIST_FAIL	DCP_BIST_FAIL	ENET_BIST_FAIL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	USB1_BIST_FAIL	USB0_BIST_FAIL	DMA1_BIST_FAIL	DMA0_BIST_FAIL	CACHE_BIST_FAIL	CAN_BIST_FAIL	PXP_BIST_DONE	LCDIF_BIST_DONE	DCP_BIST_DONE	ENET_BIST_DONE	USB1_BIST_DONE	USB0_BIST_DONE	DMA1_BIST_DONE	DMA0_BIST_DONE	CACHE_BIST_DONE	CAN_BIST_DONE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_BIST_STATUS field descriptions

Field	Description
31–30 RSVD0	Reserved.
29 PXP_BIST_RETENTION	This read-only bit is a 1 if the PXP BIST test has been enabled waiting for a resume.
28 LCDIF_BIST_RETENTION	This read-only bit is a 1 if the LCDIF BIST test has been enabled waiting for a resume.
27 DCP_BIST_RETENTION	This read-only bit is a 1 if the DCP BIST test has been enabled waiting for a resume.
26 ENET_BIST_RETENTION	This read-only bit is a 1 if the ENET BIST test has been enabled waiting for a resume.
25 USB1_BIST_RETENTION	This read-only bit is a 1 if the USB1 BIST test has been enabled waiting for a resume.
24 USB0_BIST_RETENTION	This read-only bit is a 1 if the USB0 BIST test has been enabled waiting for a resume.

HW_DIGCTL_BIST_STATUS field descriptions (continued)

Field	Description
23 DMA1_BIST_ RETENTION	This read-only bit is a 1 if the DMA1 BIST test has been enabled waiting for a resume.
22 DMA0_BIST_ RETENTION	This read-only bit is a 1 if the DMA0 BIST test has been enabled waiting for a resume.
21 CACHE_BIST_ RETENTION	This read-only bit is a 1 if the CACHE BIST test has been enabled waiting for a resume.
20 CAN_BIST_ RETENTION	This read-only bit is a 1 if the CAN BIST test has been enabled waiting for a resume.
19 PXP_BIST_FAIL	This read-only bit is a 1 if the PXP BIST test returns a failure.
18 LCDIF_BIST_ FAIL	This read-only bit is a 1 if the LCDIF BIST test returns a failure.
17 DCP_BIST_FAIL	This read-only bit is a 1 if the DCP BIST test returns a failure.
16 ENET_BIST_ FAIL	This read-only bit is a 1 if the ENET BIST test returns a failure.
15 USB1_BIST_ FAIL	This read-only bit is a 1 if the USB1 BIST test returns a failure.
14 USB0_BIST_ FAIL	This read-only bit is a 1 if the USB0 BIST test returns a failure.
13 DMA1_BIST_ FAIL	This read-only bit is a 1 if the DMA1 BIST test returns a failure.
12 DMA0_BIST_ FAIL	This read-only bit is a 1 if the DMA0 BIST test returns a failure.
11 CACHE_BIST_ FAIL	This read-only bit is a 1 if the CACHE BIST test returns a failure.
10 CAN_BIST_FAIL	This read-only bit is a 1 if the CAN BIST test returns a failure.
9 PXP_BIST_ DONE	This read-only bit is a 1 if the PXP BIST test has completed.
8 LCDIF_BIST_ DONE	This read-only bit is a 1 if the LCDIF BIST test has completed.

HW_DIGCTL_BIST_STATUS field descriptions (continued)

Field	Description
7 DCP_BIST_ DONE	This read-only bit is a 1 if the DCP BIST test has completed.
6 ENET_BIST_ DONE	This read-only bit is a 1 if the ENET BIST test has completed.
5 USB1_BIST_ DONE	This read-only bit is a 1 if the USB1 BIST test has completed.
4 USB0_BIST_ DONE	This read-only bit is a 1 if the USB0 BIST test has completed.
3 DMA1_BIST_ DONE	This read-only bit is a 1 if the DMA1 BIST test has completed.
2 DMA0_BIST_ DONE	This read-only bit is a 1 if the DMA0 BIST test has completed.
1 CACHE_BIST_ DONE	This read-only bit is a 1 if the CACHE BIST test has completed.
0 CAN_BIST_ DONE	This read-only bit is a 1 if the CAN BIST test has completed.

19.4.10 Entropy Register (HW_DIGCTL_ENTROPY)

The Entropy register is a read-only test value register.

EXAMPLE

```
while(HW_DIGCTL_STATUS.PSWITCH != 0)
{
   //wait for pswitch to go away
}
HW_DIGCTL_WRITEONCE.BITS = rand(HW_DIGCTL_ENTROPY.VALUE);
```

Address: HW_DIGCTL_ENTROPY – 8001_C000h base + 90h offset = 8001_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_ENTROPY field descriptions

Field	Description
31–0 VALUE	This read-only bit field always reads back the results of an entropy calculation. It is used to randomize the seeds for random number generators.

19.4.11 Entropy Latched Register (HW_DIGCTL_ENTROPY_LATCHED)

The Entropy Latched Register is a read-only test value register.

Address: HW_DIGCTL_ENTROPY_LATCHED – 8001_C000h base + A0h offset = 8001_ C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_ENTROPY_LATCHED field descriptions

Field	Description
31–0 VALUE	When the LATCH_ENTROPY bit in the HW_DIGCTL_CTRL register is set to 1, the value of the HW_DIGCTL_ENTROPY register is latched into this register. This can be used to latch a stable random value on players where the PSWITCH is not deasserted after power-up.

19.4.12 Digital Control Microseconds Counter Register (HW_DIGCTL_MICROSECONDS)

The Digital Control Microseconds Counter Register is a read-only test value register.

HW_DIGCTL_MICROSECONDS: 0x0C0

HW_DIGCTL_MICROSECONDS_SET: 0x0C4

HW_DIGCTL_MICROSECONDS_CLR: 0x0C8

HW_DIGCTL_MICROSECONDS_TOG: 0x0CC

This fixed-rate timer always increments at 24.0 MHz divided by 24 or 1.0 MHz. It does not generate an interrupt.

EXAMPLE

```
        StartTime = HW_DIGCTL_MICROSECONDS_RD();
        EndTime = HW_DIGCTL_MICROSECONDS_RD();
        ElapsedTime = StartTime - EndTime;      // WARNING, handle rollover in real software
```

Address: HW_DIGCTL_MICROSECONDS – 8001_C000h base + C0h offset = 8001_C0C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_MICROSECONDS field descriptions

Field	Description
31–0 VALUE	This register maintains a 32-bit counter that increments at a 1-microsecond rate. The 1-MHz clock driving this counter is derived from the 24.0-MHz crystal oscillator. The count value is not preserved over power-downs. The 32-bit value wraps in less than two hours.

19.4.13 Digital Control Debug Read Test Register (HW_DIGCTL_DBGRD)

The Digital Control Debug Read Test Register is a read-only test value register.

This register is used for debugging purposes.

EXAMPLE

```
        debug_value = HW_DIGCTL_DBGRD_RD();
```

Address: HW_DIGCTL_DBGRD – 8001_C000h base + D0h offset = 8001_C0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COMPLEMENT																															
W	▨																															
Reset	0	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0

HW_DIGCTL_DBGRD field descriptions

Field	Description
31–0 COMPLEMENT	This read-only bit field always reads back the one's complement of the value in HW_DIGCTL_DBG.

19.4.14 Digital Control Debug Register (HW_DIGCTL_DBG)

The Digital Control Debug Register is a read-only test value register.

This register is used for debugging purposes.

EXAMPLE

```
        debug_value = HW_DIGCTL_DBG_RD();
```

Address: HW_DIGCTL_DBG – 8001_C000h base + E0h offset = 8001_C0E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W	▨																															
Reset	1	0	0	0	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1

HW_DIGCTL_DBG field descriptions

Field	Description
31–0 VALUE	This read-only bit field always reads back the fixed value 0x87654321.

19.4.15 USB LOOP BACK (HW_DIGCTL_USB_LOOPBACK)

This register used to control the USB test mode for loopback tests.

HW_DIGCTL_USB_LOOPBACK: 0x100

HW_DIGCTL_USB_LOOPBACK_SET: 0x104

HW_DIGCTL_USB_LOOPBACK_CLR: 0x108

HW_DIGCTL_USB_LOOPBACK_TOG: 0x10C

This register contains controls for the USB loop back

Address: HW_DIGCTL_USB_LOOPBACK – 8001_C000h base + 100h offset = 8001_C100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD0														USB1_TST_START	TSTI1_TX_LS
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TSTI1_TX_HS	TSTI1_TX_EN	TSTI1_TX_HIZ	UTMI1_DIG_TST1	UTMI1_DIG_TST0	USB0_TST_START	TSTI0_TX_LS	TSTI0_TX_HS	TSTI0_TX_EN	TSTI0_TX_HIZ	UTMI0_DIG_TST1	UTMI0_DIG_TST0	UTMO1_ DIG_TST1	UTMO1_ DIG_TST0	UTMO0_ DIG_TST1	UTMO0_ DIG_TST0
W													▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_USB_LOOPBACK field descriptions

Field	Description
31–18 RSVD0	Always write zeroes to this bit field.
17 USB1_TST_ START	This bit enables the USB loopback test.
16 TSTI1_TX_LS	Set to 1 to choose LS, set to 0 to choose HS or FS mode which defined by TSTI1_TX_HS.
15 TSTI1_TX_HS	chooses HS or FS mode.
14 TSTI1_TX_EN	eanbles TX
13 TSTI1_TX_HIZ	makes TX HIZ
12 UTMI1_DIG_ TST1	control for mode
11 UTMI1_DIG_ TST0	control for mode
10 USB0_TST_ START	This bit enables the USB loopback test.
9 TSTI0_TX_LS	Set to 1 to choose LS, set to 0 to choose HS or FS mode which defined by TSTI1_TX_HS.
8 TSTI0_TX_HS	chooses HS or FS mode.
7 TSTI0_TX_EN	enables TX
6 TSTI0_TX_HIZ	makes TX HIZ
5 UTMI0_DIG_ TST1	control for mode

HW_DIGCTL_USB_LOOPBACK field descriptions (continued)

Field	Description
4 UTMI0_DIG_ TST0	control for mode
3 UTMO1_DIG_ TST1	This read-only bit is a status bit for USB1 LB = 1 is pass
2 UTMO1_DIG_ TST0	This read-only bit is a status bit for USB1 LB = 0 is pass
1 UTMO0_DIG_ TST1	This read-only bit is a status bit for USB0 LB = 1 is pass
0 UTMO0_DIG_ TST0	This read-only bit is a status bit for USB0 LB = 0 is pass

19.4.16 SRAM Status Register 0 (HW_DIGCTL_OCRAM_STATUS0)

The SRAM Status Register 0 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS0: 0x110

HW_DIGCTL_OCRAM_STATUS0_SET: 0x114

HW_DIGCTL_OCRAM_STATUS0_CLR: 0x118

HW_DIGCTL_OCRAM_STATUS0_TOG: 0x11C

This register contains fail data for the first fail .

EXAMPLE

```
fail_data = HW_DIGCTL_OCRAM_STATUS0_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS0 – 8001_C000h base + 110h offset = 8001_C110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA00																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS0 field descriptions

Field	Description
31–0 FAILDATA00	This read-only bit field contains the fail data for the first fail .

19.4.17 SRAM Status Register 1 (HW_DIGCTL_OCRAM_STATUS1)

The SRAM Status Register 1 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS1: 0x120

HW_DIGCTL_OCRAM_STATUS1_SET: 0x124

HW_DIGCTL_OCRAM_STATUS1_CLR: 0x128

HW_DIGCTL_OCRAM_STATUS1_TOG: 0x12C

This register contains fail data for the second fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS1_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS1 – 8001_C000h base + 120h offset = 8001_C120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA01																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS1 field descriptions

Field	Description
31–0 FAILDATA01	This read-only bit field contains the fail data for the second fail .

19.4.18 SRAM Status Register 2 (HW_DIGCTL_OCRAM_STATUS2)

SRAM Status Register 2 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS2: 0x130

HW_DIGCTL_OCRAM_STATUS2_SET: 0x134

HW_DIGCTL_OCRAM_STATUS2_CLR: 0x138

HW_DIGCTL_OCRAM_STATUS2_TOG: 0x13C

This register contains fail data for the third fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS2_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS2 – 8001_C000h base + 130h offset = 8001_C130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA10																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS2 field descriptions

Field	Description
31–0 FAILDATA10	This read-only bit field contains the fail data for the third fail .

19.4.19 SRAM Status Register 3 (HW_DIGCTL_OCRAM_STATUS3)

RAM Status Register 3 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS3: 0x140

HW_DIGCTL_OCRAM_STATUS3_SET: 0x144

HW_DIGCTL_OCRAM_STATUS3_CLR: 0x148

HW_DIGCTL_OCRAM_STATUS3_TOG: 0x14C

This register contains fail data for the 4th fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS3_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS3 – 8001_C000h base + 140h offset = 8001_C140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA11																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS3 field descriptions

Field	Description
31–0 FAILDATA11	This read-only bit field contains the fail data for the 4th fail .

19.4.20 SRAM Status Register 4 (HW_DIGCTL_OCRAM_STATUS4)

SRAM Status Register 4 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS4: 0x150

HW_DIGCTL_OCRAM_STATUS4_SET: 0x154

HW_DIGCTL_OCRAM_STATUS4_CLR: 0x158

HW_DIGCTL_OCRAM_STATUS4_TOG: 0x15C

This register contains fail data for the 5th fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS4_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS4 – 8001_C000h base + 150h offset = 8001_C150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA20																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS4 field descriptions

Field	Description
31–0 FAILDATA20	This read-only bit field contains the fail data for the 5th fail .

19.4.21 SRAM Status Register 5 (HW_DIGCTL_OCRAM_STATUS5)

SRAM Status Register 5 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS5: 0x160

HW_DIGCTL_OCRAM_STATUS5_SET: 0x164

HW_DIGCTL_OCRAM_STATUS5_CLR: 0x168

HW_DIGCTL_OCRAM_STATUS5_TOG: 0x16C

This register contains fail data for the 6th fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS5_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS5 – 8001_C000h base + 160h offset = 8001_C160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA21																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS5 field descriptions

Field	Description
31–0 FAILDATA21	This read-only bit field contains the fail data for the 6th fail .

19.4.22 SRAM Status Register 6 (HW_DIGCTL_OCRAM_STATUS6)

SRAM Status Register 6 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS6: 0x170

HW_DIGCTL_OCRAM_STATUS6_SET: 0x174

HW_DIGCTL_OCRAM_STATUS6_CLR: 0x178

HW_DIGCTL_OCRAM_STATUS6_TOG: 0x17C

This register contains fail data for the 7th fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS6_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS6 – 8001_C000h base + 170h offset = 8001_C170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA30																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS6 field descriptions

Field	Description
31–0 FAILDATA30	This read-only bit field contains the fail data for the 7th fail .

19.4.23 SRAM Status Register 7 (HW_DIGCTL_OCRAM_STATUS7)

SRAM Status Register 7 is a read-only fail data register.

HW_DIGCTL_OCRAM_STATUS7: 0x180

HW_DIGCTL_OCRAM_STATUS7_SET: 0x184

HW_DIGCTL_OCRAM_STATUS7_CLR: 0x188

HW_DIGCTL_OCRAM_STATUS7_TOG: 0x18C

This register contains fail data for the 8th fail .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS7_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS7 – 8001_C000h base + 180h offset = 8001_C180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILDATA31																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS7 field descriptions

Field	Description
31–0 FAILDATA31	This read-only bit field contains the fail data for the 8th fail .

19.4.24 SRAM Status Register 8 (HW_DIGCTL_OCRAM_STATUS8)

SRAM Status Register 8 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS8: 0x190

HW_DIGCTL_OCRAM_STATUS8_SET: 0x194

HW_DIGCTL_OCRAM_STATUS8_CLR: 0x198

HW_DIGCTL_OCRAM_STATUS8_TOG: 0x19C

This register contains fail address for the first and second failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS8_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS8 – 8001_C000h base + 190h offset = 8001_C190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILADDR01																FAILADDR00															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS8 field descriptions

Field	Description
31–16 FAILADDR01	This read-only bit field contains the failing address for the second fail .
15–0 FAILADDR00	This read-only bit field contains the failing address for the first fail .

19.4.25 SRAM Status Register 9 (HW_DIGCTL_OCRAM_STATUS9)

SRAM Status Register 9 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS9: 0x1A0

HW_DIGCTL_OCRAM_STATUS9_SET: 0x1A4

HW_DIGCTL_OCRAM_STATUS9_CLR: 0x1A8

HW_DIGCTL_OCRAM_STATUS9_TOG: 0x1AC

This register contains fail address for the third and 4th failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS9_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS9 – 8001_C000h base + 1A0h offset = 8001_C1A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILADDR11																FAILADDR10															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS9 field descriptions

Field	Description
31–16 FAILADDR11	This read-only bit field contains the failing address for the 4th fail .
15–0 FAILADDR10	This read-only bit field contains the failing address for the third fail .

19.4.26 SRAM Status Register 10 (HW_DIGCTL_OCRAM_STATUS10)

SRAM Status Register 10 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS10: 0x1B0

HW_DIGCTL_OCRAM_STATUS10_SET: 0x1B4

HW_DIGCTL_OCRAM_STATUS10_CLR: 0x1B8

HW_DIGCTL_OCRAM_STATUS10_TOG: 0x1BC

This register contains fail address for the 5th and 6th failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS10_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS10 – 8001_C000h base + 1B0h offset = 8001_C1B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILADDR21																FAILADDR20															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS10 field descriptions

Field	Description
31–16 FAILADDR21	This read-only bit field contains the failing address for the 6th fail .
15–0 FAILADDR20	This read-only bit field contains the failing address for the 5th fail .

19.4.27 SRAM Status Register 11 (HW_DIGCTL_OCRAM_STATUS11)

SRAM Status Register 11 is a read-only fail address register.

HW_DIGCTL_OCRAM_STATUS11: 0x1C0

HW_DIGCTL_OCRAM_STATUS11_SET: 0x1C4

HW_DIGCTL_OCRAM_STATUS11_CLR: 0x1C8

HW_DIGCTL_OCRAM_STATUS11_TOG: 0x1CC

This register contains fail address for the 7th and 8th failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS11_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS11 – 8001_C000h base + 1C0h offset = 8001_C1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FAILADDR31																FAILADDR30															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS11 field descriptions

Field	Description
31–16 FAILADDR31	This read-only bit field contains the failing address for the 8th fail .

HW_DIGCTL_OCRAM_STATUS11 field descriptions (continued)

Field	Description
15–0 FAILADDR30	This read-only bit field contains the failing address for the 7th fail .

19.4.28 SRAM Status Register 12 (HW_DIGCTL_OCRAM_STATUS12)

SRAM Status Register 12 is a read-only fail state register.

HW_DIGCTL_OCRAM_STATUS12: 0x1D0

HW_DIGCTL_OCRAM_STATUS12_SET: 0x1D4

HW_DIGCTL_OCRAM_STATUS12_CLR: 0x1D8

HW_DIGCTL_OCRAM_STATUS12_TOG: 0x1DC

This register contains fail state for the first, second, third and 4th failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS12_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS12 – 8001_C000h base + 1D0h offset = 8001_C1D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3	FAILSTATE11							RSVD2	FAILSTATE10						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	FAILSTATE01							RSVD0	FAILSTATE00						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS12 field descriptions

Field	Description
31 RSVD3	This field is unused.
30–24 FAILSTATE11	This read-only bit field contains the failing state for the 4th fail .

HW_DIGCTL_OCRAM_STATUS12 field descriptions (continued)

Field	Description
23 RSVD2	This field is unused.
22–16 FAILSTATE10	This read-only bit field contains the failing state for the 3th fail .
15 RSVD1	This field is unused.
14–8 FAILSTATE01	This read-only bit field contains the failing state for the second fail .
7 RSVD0	This field is unused.
6–0 FAILSTATE00	This read-only bit field contains the failing state for the first fail .

19.4.29 SRAM Status Register 13 (HW_DIGCTL_OCRAM_STATUS13)

SRAM Status Register 13 is a read-only fail state register.

HW_DIGCTL_OCRAM_STATUS13: 0x1E0

HW_DIGCTL_OCRAM_STATUS13_SET: 0x1E4

HW_DIGCTL_OCRAM_STATUS13_CLR: 0x1E8

HW_DIGCTL_OCRAM_STATUS13_TOG: 0x1EC

This register contains fail state for the 5th, 6th, 7th, and 8th failures .

EXAMPLE

```
        fail_data = HW_DIGCTL_OCRAM_STATUS13_RD();
```

Address: HW_DIGCTL_OCRAM_STATUS13 – 8001_C000h base + 1E0h offset = 8001_C1E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3	FAILSTATE31							RSVD2	FAILSTATE30						
W	▨	▨							▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	FAILSTATE21							RSVD0	FAILSTATE20						
W	▨	▨							▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_OCRAM_STATUS13 field descriptions

Field	Description
31 RSVD3	This field is unused.
30–24 FAILSTATE31	This read-only bit field contains the failing state for the 8th fail .
23 RSVD2	This field is unused.
22–16 FAILSTATE30	This read-only bit field contains the failing state for the 7th fail .
15 RSVD1	This field is unused.
14–8 FAILSTATE21	This read-only bit field contains the failing state for the 6th fail .
7 RSVD0	This field is unused.
6–0 FAILSTATE20	This read-only bit field contains the failing state for the 5th fail .

19.4.30 Digital Control Scratch Register 0 (HW_DIGCTL_SCRATCH0)

Digital control scratch pad

Scratch Pad Register 0.

EXAMPLE

```
                scratch_pad = (*void)HW_DIGCTL_SCRATCH0.PTR;
```

Address: HW_DIGCTL_SCRATCH0 – 8001_C000h base + 280h offset = 8001_C280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PTR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_SCRATCH0 field descriptions

Field	Description
31–0 PTR	Digital control scratch pad

19.4.31 Digital Control Scratch Register 1 (HW_DIGCTL_SCRATCH1)

Scratch Pad Register 1.

EXAMPLE

```
                scratch_pad = (*void)HW_DIGCTL_SCRATCH1.PTR;
```

Address: HW_DIGCTL_SCRATCH1 – 8001_C000h base + 290h offset = 8001_C290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PTR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_SCRATCH1 field descriptions

Field	Description
31–0 PTR	Digital control scratch pad

19.4.32 Digital Control ARM Cache Register (HW_DIGCTL_ARMCACHE)

This register provides the ARM cache RAM controls.

ARM Cache Control Register.

EXAMPLE

```
        cache_timing = HW_DIGCTL_ARMCACHE.CACHE_SS;
```

Address: HW_DIGCTL_ARMCACHE – 8001_C000h base + 2A0h offset = 8001_
C2A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4														VALID_SS	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3		DRTY_SS		RSVD2		CACHE_SS		RSVD1		DTAG_SS		RSVD0		ITAG_SS	
W	▨				▨				▨				▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_ARMCACHE field descriptions

Field	Description
31–18 RSVD4	Reserved.
17–16 VALID_SS	Timing control for 64x24x1 RAMs (both instruction and data cache_valid arrays).
15–14 RSVD3	Reserved.
13–12 DRTY_SS	Timing control for 128x8x1 RAM (DDRTY).
11–10 RSVD2	Reserved.
9–8 CACHE_SS	Timing Control for 1024x32x4 RAMs (both instruction and data cache arrays).
7–6 RSVD1	Reserved.
5–4 DTAG_SS	Timing Control for 256x22x4 RAM (DTAG).
3–2 RSVD0	Reserved.
1–0 ITAG_SS	Timing Control for 128x22x4 RAM (ITAG).

19.4.33 Debug Trap Control and Status for AHB Layer 0 and 3 (HW_DIGCTL_DEBUG_TRAP)

The Debug Trap Register provides control and status information for the trap functionality.

HW_DIGCTL_DEBUG_TRAP: 0x2B0

HW_DIGCTL_DEBUG_TRAP_SET: 0x2B4

HW_DIGCTL_DEBUG_TRAP_CLR: 0x2B8

HW_DIGCTL_DEBUG_TRAP_TOG: 0x2BC

This register sets the lower address that defines the debug trap function. When this function is enabled, any active AHB cycle on either Layer 0 or Layer 0 which accesses this range will trigger an interrupt to the ARM core.

EXAMPLE

Address: HW_DIGCTL_DEBUG_TRAP – 8001_C000h base + 2B0h offset = 8001_C2B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2[15:12]				RSVD1		TRAP_L0_ MASTER_ID		RSVD0	TRAP_L3_MASTER_ ID			TRAP_L3_ IRQ	TRAP_L0_ IRQ	TRAP_IN_ RANGE	TRAP_ ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_DEBUG_TRAP field descriptions

Field	Description
31–12 RSVD2	Reserved.
11–10 RSVD1	Reserved.
9–8 TRAP_L0_ MASTER_ID	ID of master on AHB Layer 0 that triggered the TRAP_L0_IRQ. The value in this bitfield is updated when TRAP_L0_IRQ is set. 0x0 **PXP** — PXP 0x1 **LCDIF** — LCDIF 0x2 **BCH** — BCH 0x3 **DCP** — DCP
7 RSVD0	Reserved.
6–4 TRAP_L3_ MASTER_ID	ID of master on AHB Layer 3 that triggered the TRAP_L3_IRQ. The value in this bitfield is updated when TRAP_L3_IRQ is set. 0x0 **APBH_BRIDE_DMA** — APBH Bridge DMA 0x1 **APBX_BRIDE_DMA** — APBX Bridge DMA 0x2 **USB0** — USB0

HW_DIGCTL_DEBUG_TRAP field descriptions (continued)

Field	Description
	0x3 **USB1** — USB1 0x4 **ENET_M0** — ENET_M0 0x5 **ENET_M1** — ENET_M1
3 TRAP_L3_IRQ	This bit is set when an AHB access occurs to the range defined by the TRAP_ADDR registers below and the trap function is enabled with the TRAP_ENABLE bit.
2 TRAP_L0_IRQ	This bit is set when an AHB access occurs to the range defined by the TRAP_ADDR registers below and the trap function is enabled with the TRAP_ENABLE bit.
1 TRAP_IN_ RANGE	Determines whether the debug trap function causes a match when the master address is inside (low-address <= current-address <= high-address) the specified range. 0 = The trap occurs when the master address falls outside of the range. 1 = The check is inside the range.
0 TRAP_ENABLE	Enables the AHB arbiter debug trap functions. When a trap occurs and this bit is set, an interrupt is sent to the ARM core.

19.4.34 Debug Trap Range Low Address for AHB Layer 0 (HW_DIGCTL_DEBUG_TRAP_L0_ADDR_LOW)

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range. This register applies only to AHB Layer 0.

This register sets the lower address that defines the debug trap function. When this function is enabled, any active AHB cycle on Layer 0 which accesses this range will trigger an interrupt to the ARM core.

EXAMPLE

Address: HW_DIGCTL_DEBUG_TRAP_L0_ADDR_LOW – 8001_C000h base + 2C0h offset = 8001_C2C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_DEBUG_TRAP_L0_ADDR_LOW field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit lower address for the debug trap range.

19.4.35 Debug Trap Range High Address for AHB Layer 0 (HW_DIGCTL_DEBUG_TRAP_L0_ADDR_HIGH)

The Debug Trap Range High Address Register defines the upper bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range. This register applies only to AHB Layer 0.

This register sets the upper address that defines the debug trap function. When this function is enabled, any active AHB cycle on Layer 0 which accesses this range will trigger an interrupt to the ARM core.

EXAMPLE

Address: HW_DIGCTL_DEBUG_TRAP_L0_ADDR_HIGH – 8001_C000h base + 2D0h offset = 8001_C2D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_DEBUG_TRAP_L0_ADDR_HIGH field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit upper address for the debug trap range.

19.4.36 Debug Trap Range Low Address for AHB Layer 3 (HW_DIGCTL_DEBUG_TRAP_L3_ADDR_LOW)

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range. This register applies only to AHB Layer 3.

This register sets the lower address that defines the debug trap function. When this function is enabled, any active AHB cycle on Layer 3 which accesses this range will trigger an interrupt to the ARM core.

EXAMPLE

Address: HW_DIGCTL_DEBUG_TRAP_L3_ADDR_LOW – 8001_C000h base + 2E0h offset = 8001_C2E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_DEBUG_TRAP_L3_ADDR_LOW field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit lower address for the debug trap range.

19.4.37 Debug Trap Range High Address for AHB Layer 3 (HW_DIGCTL_DEBUG_TRAP_L3_ADDR_HIGH)

The Debug Trap Range High Address Register defines the upper bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AHB cycle occurs within this range. This register applies only to AHB Layer 3.

This register sets the upper address that defines the debug trap function. When this function is enabled, any active AHB cycle on Layer 3 which accesses this range will trigger an interrupt to the ARM core.

EXAMPLE

Address: HW_DIGCTL_DEBUG_TRAP_L3_ADDR_HIGH – 8001_C000h base + 2F0h offset = 8001_C2F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_DEBUG_TRAP_L3_ADDR_HIGH field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit upper address for the debug trap range.

19.4.38 Freescale Copyright Identifier Register (HW_DIGCTL_FSL)

Read-only Freescale Copyright Identifier Register.

This register provides read-only access to the zero-terminated twelve-byte Freescale copyright identification string. This register behaves somewhat differently from all other APB registers in that it provides different read-back values at its three successive SCT bus addresses. The following binary values are read back at 0x300, 0x304, and 0x308 respectively: 0x65657246 e,e,r,F at 0x300 0x6c616373 l,a,c,e at 0x304 0xAEA92d65 0x00, Registered Trademark (®), Copyright (©), hyphen (-), e at 0x308 The debugger does a string compare on these 12 successive little endian bytes. Any chip that reads back these values is either a Freescale chip or it is a competitors chip that is violating Freescale registered trademarks and or copyrights.

EXAMPLE

```
        printf("%s", (char *)HW_DIGCTL_SGTL_ADDR);
```

Address: HW_DIGCTL_FSL – 8001_C000h base + 300h offset = 8001_C300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COPYRIGHT																															
W																																
Reset	0	1	1	0	0	1	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	1	0	0	1	0	0	0	1	1	0

HW_DIGCTL_FSL field descriptions

Field	Description
31–0 COPYRIGHT	This read-only bit field contains the four bytes of the Freescale Copyright Identification String.

19.4.39 Digital Control Chip Revision Register (HW_DIGCTL_CHIPID)

Read-only chip revision register.

This register is for Chip Revision

EXAMPLE

```
                FormatAndPrintChipID(HW_DIGCTL_CHIPID_PRODUCT_CODE,HW_DIGCTL_CHIPID_REVISION
 );
```

Address: HW_DIGCTL_CHIPID – 8001_C000h base + 310h offset = 8001_C310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRODUCT_CODE																RSVD0								REVISION							
W																																
Reset	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_CHIPID field descriptions

Field	Description
31–16 PRODUCT_ CODE	This read-only bit field returns 0x2800, which identifies the generation from which the part is derived.
15–8 RSVD0	Reserved.
7–0 REVISION	This read-only bit field always reads back the mask revision level of the chip.

19.4.40 AHB Statistics Control Register (HW_DIGCTL_AHB_STATS_SELECT)

The AHB Statistics Control Register selects which AHB Masters on each Layer of the AHB subsystem are enabled to contribute to the statistics calculations.

This register is to enable performance monitoring for the corresponding AHB master in layers

EXAMPLE

Address: HW_DIGCTL_AHB_STATS_SELECT – 8001_C000h base + 330h offset = 8001_C330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								L3_MASTER_SELECT								L2_MASTER_SELECT								L1_MASTER_SELECT							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_AHB_STATS_SELECT field descriptions

Field	Description
31–24 RSVD0	Reserved.
23–16 L3_MASTER_ SELECT	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 3 arbiter for the corresponding AHB master. Bits [7:6] are currently reserved and should not be set 0x1 **APBHDMA** — Select APBH DMA Master. 0x2 **APBXDMA** — Select APBX DMA Master.

HW_DIGCTL_AHB_STATS_SELECT field descriptions (continued)

Field	Description
	0x4 **USB0** — Select USB0 Master. 0x8 **USB1** — Select USB1 Master. 0x10 **UDMA0** — Select UDMA0 Master. 0x20 **UDMA1** — Select UDMA1 Master.
15–8 L2_MASTER_ SELECT	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 2 arbiter for the corresponding AHB master. Bits [7:1] are currently reserved and should not be set 0x1 **ARMD** — Select ARM DATA Master.
7–0 L1_MASTER_ SELECT	Set various bits of this bit field to one to enable performance monitoring in the AHB Layer 1 arbiter for the corresponding AHB master. Bits [7:1] are currently reserved and should not be set 0x1 **ARMI** — Select ARM Instruction Master.

19.4.41 AHB Layer 1 Transfer Count Register (HW_DIGCTL_L1_AHB_ACTIVE_CYCLES)

The AHB Layer 1 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 0.

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_CTRL_MASTER_SELECT are used in the arbiter to mask which master\'s cycles are actually recorded here.

EXAMPLE

```
NumberCycles = HW_DIGCTL_L1_AHB_ACTIVE_CYCLES_COUNT_RD();
```

Address: HW_DIGCTL_L1_AHB_ACTIVE_CYCLES – 8001_C000h base + 370h offset = 8001_C370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L1_AHB_ACTIVE_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 1.

19.4.42 AHB Layer 1 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L1_AHB_DATA_STALLED)

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE

```
NumberStalledCycles = HW_DIGCTL_L1_AHB_DATA_STALLED_COUNT_RD();
```

Address: HW_DIGCTL_L1_AHB_DATA_STALLED – 8001_C000h base + 380h offset = 8001_C380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L1_AHB_DATA_STALLED field descriptions

Field	Description
31–0 COUNT	This field counts the number of AHB cycles in which a master was stalled.

19.4.43 AHB Layer 1 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L1_AHB_DATA_CYCLES)

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L1_AHB_DATA_CYCLES.COUNT less than 1000000)
{
    // wait for a specific number of xfers
}
```

```
        ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

Address: HW_DIGCTL_L1_AHB_DATA_CYCLES – 8001_C000h base + 390h offset = 8001_C390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L1_AHB_DATA_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

19.4.44 AHB Layer 2 Transfer Count Register (HW_DIGCTL_L2_AHB_ACTIVE_CYCLES)

The AHB Layer 2 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 2.

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_CTRL_MASTER_SELECT are used in the arbiter to mask which master\'s cycles are actually recorded here.

EXAMPLE

```
NumberCycles = HW_DIGCTL_L2_AHB_ACTIVE_CYCLES_COUNT_RD();
```

Address: HW_DIGCTL_L2_AHB_ACTIVE_CYCLES – 8001_C000h base + 3A0h offset = 8001_C3A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L2_AHB_ACTIVE_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 2.

19.4.45 AHB Layer 2 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L2_AHB_DATA_STALLED)

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE

```
NumberStalledCycles = HW_DIGCTL_L2_AHB_DATA_STALLED_COUNT_RD();
```

Address: HW_DIGCTL_L2_AHB_DATA_STALLED – 8001_C000h base + 3B0h offset = 8001_C3B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L2_AHB_DATA_STALLED field descriptions

Field	Description
31–0 COUNT	This field counts the number of AHB cycles in which a master was stalled.

19.4.46 AHB Layer 2 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L2_AHB_DATA_CYCLES)

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE

```
        StartTime = HW_DIGCTL_HCLKCOUNT_RD();
        while(HW_DIGCTL_L2_AHB_DATA_CYCLES.COUNT less than 1000000)
        {
            // wait for a specific number of xfers
        }
        ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

Address: HW_DIGCTL_L2_AHB_DATA_CYCLES – 8001_C000h base + 3C0h offset = 8001_C3C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L2_AHB_DATA_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

19.4.47 AHB Layer 3 Transfer Count Register (HW_DIGCTL_L3_AHB_ACTIVE_CYCLES)

The AHB Layer 3 Transfer Count Register counts the number of AHB bus cycles during which a transfer is active on AHB Layer 3.

This field counts the number of AHB cycles in which a master was requesting a transfer, and the slave had not responded. This includes cycles in which it was requesting transfers but was not granted them, as well as cycles in which it was granted and driving the bus but the targeted slave was not ready. The master selects in HW_DIGCTL_CTRL_MASTER_SELECT are used in the arbiter to mask which master\'s cycles are actually recorded here.

EXAMPLE

```
NumberCycles = HW_DIGCTL_L3_AHB_ACTIVE_CYCLES_COUNT_RD();
```

Address: HW_DIGCTL_L3_AHB_ACTIVE_CYCLES – 8001_C000h base + 3D0h offset = 8001_C3D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L3_AHB_ACTIVE_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which a master was active on the AHB Layer 3.

19.4.48 AHB Layer 3 Performance Metric for Stalled Bus Cycles Register (HW_DIGCTL_L3_AHB_DATA_STALLED)

Used for AHB bus utilization measurements, the AHB Performance Metric for Stalled Bus Cycles Register counts the number of stalled AHB cycles.

This counter increments on a data-phase of the AHB in which the HREADY signal is low, indicating a stalled data transfer.

EXAMPLE

```
NumberStalledCycles = HW_DIGCTL_L3_AHB_DATA_STALLED_COUNT_RD();
```

Address: HW_DIGCTL_L3_AHB_DATA_STALLED – 8001_C000h base + 3E0h offset = 8001_C3E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L3_AHB_DATA_STALLED field descriptions

Field	Description
31–0 COUNT	This field counts the number of AHB cycles in which a master was stalled.

19.4.49 AHB Layer 3 Performance Metric for Valid Bus Cycles Register (HW_DIGCTL_L3_AHB_DATA_CYCLES)

Used for AHB bus utilization measurements, the AHB Performance Metric for Valid Bus Cycles Register counts the number of actual AHB cycles in which a data transfer is completed.

This field counts the number of AHB cycles in which a master completed a data transfer (a data-phase in which HREADY is high).

EXAMPLE

```
StartTime = HW_DIGCTL_HCLKCOUNT_RD();
while(HW_DIGCTL_L3_AHB_DATA_CYCLES.COUNT less than 1000000)
{
    // wait for a specific number of xfers
}
ElapsedTime = HW_DIGCTL_HCLKCOUNT_RD() - StartTime;
```

Address: HW_DIGCTL_L3_AHB_DATA_CYCLES – 8001_C000h base + 3F0h offset = 8001_C3F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_L3_AHB_DATA_CYCLES field descriptions

Field	Description
31–0 COUNT	This field contains the count of AHB bus cycles during which data was actually transferred from a master to a slave or from a slave to a master.

19.4.50 Default First Level Page Table Movable PTE Locator 0 (HW_DIGCTL_MPTE0_LOC)

This register is used by the DFLPT to set the location for MPTE0.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE0) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE0_LOC – 8001_C000h base + 500h offset = 8001_C500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_DIGCTL_MPTE0_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the 4-values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.51 Default First-Level Page Table Movable PTE Locator 1 (HW_DIGCTL_MPTE1_LOC)

This register is used by the DFLPT to set the location for MPTE1.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE1) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE1_LOC – 8001_C000h base + 510h offset = 8001_C510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_DIGCTL_MPTE1_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the 4-values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.52 Default First-Level Page Table Movable PTE Locator 2 (HW_DIGCTL_MPTE2_LOC)

This register is used by the DFLPT to set the location for MPTE2.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE2) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE2_LOC – 8001_C000h base + 520h offset = 8001_C520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

HW_DIGCTL_MPTE2_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.

HW_DIGCTL_MPTE2_LOC field descriptions (continued)

Field	Description
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1MB addression within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the 4-values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.53 Default First-Level Page Table Movable PTE Locator 3 (HW_DIGCTL_MPTE3_LOC)

This register is used by the DFLPT to set the location for MPTE3.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE3) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE3_LOC – 8001_C000h base + 530h offset = 8001_C530h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

HW_DIGCTL_MPTE3_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the 4-values within that span; that is, all base addresses within a span are contiguous.

HW_DIGCTL_MPTE3_LOC field descriptions (continued)

Field	Description
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.54 Default First-Level Page Table Movable PTE Locator 4 (HW_DIGCTL_MPTE4_LOC)

This register is used by the DFLPT to set the location for MPTE4.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE4) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE4_LOC – 8001_C000h base + 540h offset = 8001_C540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_DIGCTL_MPTE4_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the 4-values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.55 Default First-Level Page Table Movable PTE Locator 5 (HW_DIGCTL_MPTE5_LOC)

This register is used by the DFLPT to set the location for MPTE5.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE5) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE5_LOC – 8001_C000h base + 550h offset = 8001_C550h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

HW_DIGCTL_MPTE5_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.56 Default First-Level Page Table Movable PTE Locator 6 (HW_DIGCTL_MPTE6_LOC)

This register is used by the DFLPT to set the location for MPTE6.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE6) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE6_LOC – 8001_C000h base + 560h offset = 8001_C560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

HW_DIGCTL_MPTE6_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.57 Default First-Level Page Table Movable PTE Locator 7 (HW_DIGCTL_MPTE7_LOC)

This register is used by the DFLPT to set the location for MPTE7.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE7) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE7_LOC – 8001_C000h base + 570h offset = 8001_C570h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

HW_DIGCTL_MPTE7_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.58 Default First-Level Page Table Movable PTE Locator 8 (HW_DIGCTL_MPTE8_LOC)

This register is used by the DFLPT to set the location for MPTE8.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE8) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE8_LOC – 8001_C000h base + 580h offset = 8001_C580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W	DIS	▨	▨	▨	▨	SPAN			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	LOC											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

HW_DIGCTL_MPTE8_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.

HW_DIGCTL_MPTE8_LOC field descriptions (continued)

Field	Description
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.59 Default First-Level Page Table Movable PTE Locator 9 (HW_DIGCTL_MPTE9_LOC)

This register is used by the DFLPT to set the location for MPTE9.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE9) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE9_LOC – 8001_C000h base + 590h offset = 8001_C590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

HW_DIGCTL_MPTE9_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.

HW_DIGCTL_MPTE9_LOC field descriptions (continued)

Field	Description
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.60 Default First-Level Page Table Movable PTE Locator 10 (HW_DIGCTL_MPTE10_LOC)

This register is used by the DFLPT to set the location for MPTE10.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE10) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE10_LOC – 8001_C000h base + 5A0h offset = 8001_C5A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0

HW_DIGCTL_MPTE10_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.61 Default First-Level Page Table Movable PTE Locator 11 (HW_DIGCTL_MPTE11_LOC)

This register is used by the DFLPT to set the location for MPTE11.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE11) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE11_LOC – 8001_C000h base + 5B0h offset = 8001_C5B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1

HW_DIGCTL_MPTE11_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.62 Default First-Level Page Table Movable PTE Locator 12 (HW_DIGCTL_MPTE12_LOC)

This register is used by the DFLPT to set the location for MPTE12.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE12) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE12_LOC – 8001_C000h base + 5C0h offset = 8001_C5C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

HW_DIGCTL_MPTE12_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.63 Default First-Level Page Table Movable PTE Locator 13 (HW_DIGCTL_MPTE13_LOC)

This register is used by the DFLPT to set the location for MPTE13.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE13) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE13_LOC – 8001_C000h base + 5D0h offset = 8001_C5D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1

HW_DIGCTL_MPTE13_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.64 Default First-Level Page Table Movable PTE Locator 14 (HW_DIGCTL_MPTE14_LOC)

This register is used by the DFLPT to set the location for MPTE14.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE14) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE14_LOC – 8001_C000h base + 5E0h offset = 8001_C5E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W	DIS	▨				SPAN			▨												LOC											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

HW_DIGCTL_MPTE14_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.

HW_DIGCTL_MPTE14_LOC field descriptions (continued)

Field	Description
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

19.4.65 Default First-Level Page Table Movable PTE Locator 15 (HW_DIGCTL_MPTE15_LOC)

This register is used by the DFLPT to set the location for MPTE15.

When using the hardware-based Default First-Level Page Table (DFLPT), program this value to set the location for Movable PTE (MPTE15) to any of the 4096 sections.

Address: HW_DIGCTL_MPTE15_LOC – 8001_C000h base + 5F0h offset = 8001_C5F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DIS	RSVD1				SPAN			RSVD0												LOC											
W		▨	▨	▨	▨				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

HW_DIGCTL_MPTE15_LOC field descriptions

Field	Description
31 DIS	Setting this bit to 1 disables the MPTE. A disabled MPTE cannot be bound to the page table and cannot be modified.
30–27 RSVD1	Reserved.
26–24 SPAN	This bit-field allows this PTE to span a larger than 1MB section of memory. Specifically 1MB * (2^SPAN). Care must be taken to make sure that spanned regions do not overlap in the page table, since the DFLPT does not provide any correction mechanism in an overlapped scenario. Because only one value can be set for the MPTE base-address section entry, the DFLPT assumes linear physical 1 MB addressing within a SPAN. For example, assuming a value N for LOC and SPAN=2, the DFLPT assumes the value of LOC, LOC+1, LOC+2 and LOC+3 for the four values within that span; that is, all base addresses within a span are contiguous.

HW_DIGCTL_MPTE15_LOC field descriptions (continued)

Field	Description
23–12 RSVD0	Reserved.
11–0 LOC	Value of LOC corresponds to 1MB section number (0x000-0xFFF) within the DFLPT.Note that when the SPAN field is used, any MPTE can cover up to 128MB. Do not program to 0x800 (fixed PIO entry). No two HW_DIGCTL_MPTEn_LOC registers can have the same value, and care must be taken to not create overlap when using the SPAN feature. Doing so will result in non-deterministic behavior of the DFLPT.

Chapter 20
On-Chip OTP (OCOTP) Controller

20.1 OCOTP Overview

The on-chip OTP controller (OCOTP) provides the following functions:

- Full memory-mapped (restricted) read access of 1.25 Kbit of on-chip OTP ROM.
- Data-register programming interface for the 1.25 Kbit of OTP.
- Generation of the chip hardware capability bus.
- Sources of two security keys that are supplied automatically to the DCP block through private internal busses.
- Chip-level pin access to nonrestricted portions of OTP.

The OCOTP is connected to the APBH system peripheral bus and is accessible through the ARM core. Read accesses can be performed at the maximum HCLK frequency. Programming/writes can be performed at 24 MHz. There are two security encription keys that are supplied to the DCP block. The first is the Crypto key which is made up of the 128 bits from registers HW_OCOTP_CRYPTO0 through HW_OCOTP_CRYPTO3. The least significant bits are in register 0. The Unique key is made up of the 64 bits from registers HW_OCOTP_OPS2 and HW_OCOTP_UN2. The least significant bits are contained in the OPS2 register. The system diagram for the OCOTP is shown below.

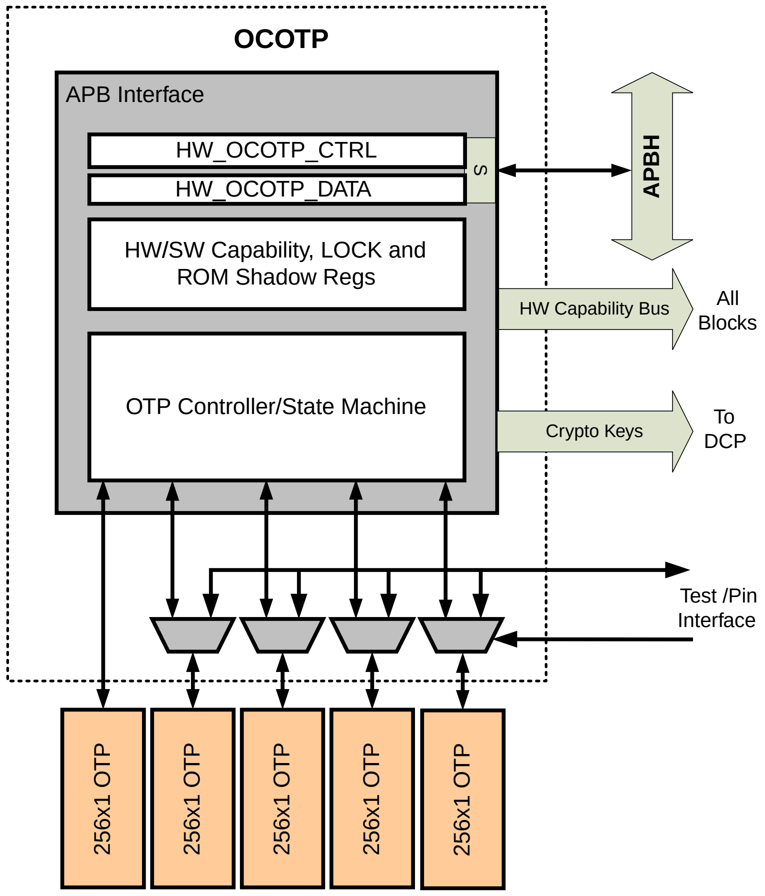


Figure 20-1. On-Chip OTP (OCOTP) Controller Block Diagram

20.2 Operation

The APB interface of the OCOTP provides two functions:

- Programmer-model access to registers (see Programmable Registers for register details). These operations require a bank opening sequence through HW_OCOTP_CTRL_RD_BANK_OPEN.
- Restricted 32-bit word write/program access to the 1.25 Kbit OTP.

The OTP is divided into 32-bit words (40 in total). All of the 40 words are memory-mapped to APBH addresses (for reads only). Writes require the use of HW_OCOTP_CTRL_ADDR. The high-level OTP allocation for i.MX28 is shown below.

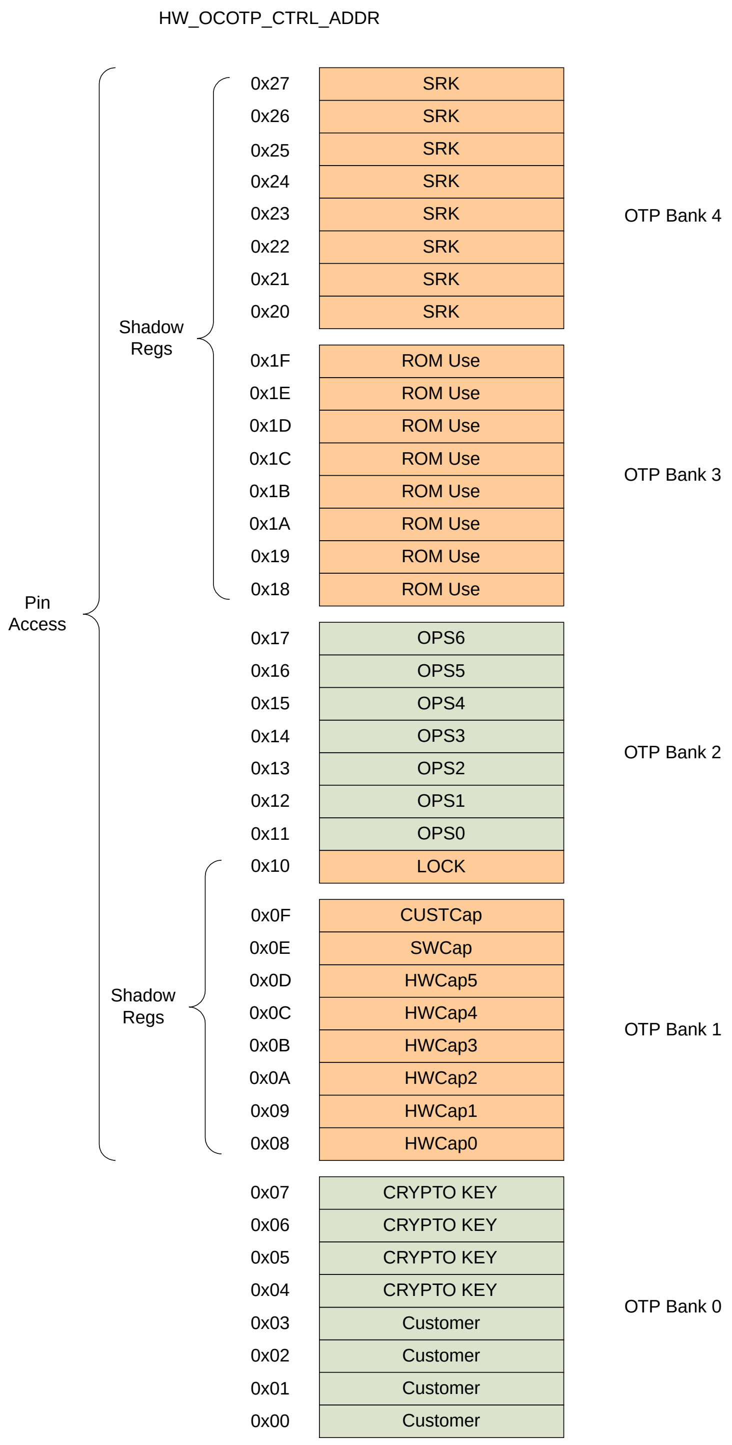


Figure 20-2. OCOTP Allocation-Internal View

OTP reads and writes can be performed on 32-bit words only. For writes, the 32-bit word reflects the write mask, such that bit fields with 0 will not be programmed and bit fields with 1 will be programmed.

For OTP random access, the programming interface consists of:

- HW_OCOTP_DATA—Data register (32- bit) for OTP programming (writes).
- HW_OCOTP_CTRL_ADDR—Address register (6-bit) for OTP programming (writes).
- HW_OCOTP_CTRL_BUSY—Programming/write request/status handshake bit.
- HW_OCOTP_CTRL_ERROR—Read/write access error status.
- HW_OCOTP_CTRL_RD_BANK_OPEN—Status of OTP read availability (reads).

20.2.1 Software Read Sequence

Reading OTP contents is relatively simple, because all OTP words are memory-mapped on the APB space (see Programmable Registers for details). These registers are read-only, except for the HW/SW capability, CUSTCAP, ROM and SRK shadow registers, which are writable until the appropriate LOCK bit in OTP is set.

Due to the fuse-read architecture, the OTP banks must be open before they can be read. This is accomplished as follows (the following does not apply to shadow registers, which can be read at any time).

1. Program the HCLK to a frequency up to the maximum allowable HCLK frequency. Note that this cannot exceed 200 MHz.
2. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear.
3. Set HW_OCOTP_CTRL_RD_BANK_OPEN. This will kick the controller to put the fuses into read mode. The controller will set HW_OCOTP_CTRL_BUSY until the OTP contents are readable. Note that if there was a pending write (holding HW_OCOTP_CTRL_BUSY) and HW_OCOTP_CTRL_RD_BANK_OPEN was set, the controller would complete the write and immediately move into read operation (keeping HW_OCOTP_CTRL_BUSY set while the banks are being opened).
4. Poll for HW_OCOTP_CTRL_BUSY clear. When HW_OCOTP_CTRL_BUSY is clear and HW_OCOTP_CTRL_RD_BANK_OPEN is set, read the data from the appropriate memory-mapped address. Note that this is not necessary for registers that are shadowed. Reading before HW_OCOTP_CTRL_BUSY is cleared by the controller, will return 0xBADA_BADA and will result in the setting of HW_OCOTP_CTRL_ERROR. Because opening banks takes approximately 33 HCLK cycles, immediate polling for BUSY is not recommended.
5. Once accesses are complete, clear HW_OCOTP_CTRL_RD_BANK_OPEN. Leaving the banks open will cause current drain.

If data is accessed from a protected region (such as the crypto key, once a read LOCK bit has been set), the controller returns 0xBADA_BADA. In addition HW_OCOTP_CTRL_ERROR is set. It must be cleared by the software before any new write access can be issued. Subsequent reads to unrestricted mapped OTP locations will still work successfully assuming that HW_OCOTP_CTRL_RD_BANK_OPEN is set and HW_OCOTP_CTRL_BUSY is clear.

It should be noted that after opening the banks, read latencies to OTP are instant (meaning they behave like regular reads from hardware registers), since parallel loading is used.

It should also be noted that setting HW_OCOTP_CTRL_RELOAD_SHADOWS to reload shadow registers does not set HW_OCOTP_CTRL_RD_BANK_OPEN. HW_OCOTP_CTRL_RD_BANK_OPEN can only be set and cleared by software. Forced reloading of shadows is covered in Shadow Registers and Hardware Capability Bus.

20.2.2 Software Write Sequence

In order to avoid erroneous code performing erroneous writes to OTP, a special unlocking sequence is required for writes.

1. Program HCLK to 24 MHz. OTP writes do not work at frequencies above 24 MHz.
2. Set the VDDIO voltage to 2.8 V (using HW_POWER_VDDIOCTRL_TRG). The VDDIO voltage is used to program OTP. Incorrect voltage and frequency settings will result in the OTP being programmed with incorrect values.
3. Check that HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_ERROR are clear. Overlapped accesses are not supported by the controller. Any pending write must be completed before a write access can be requested. In addition, the banks cannot be open for reading, so HW_OCOTP_CTRL_RD_BANK_OPEN must also be clear. If the write is done following a previous write, the postamble wait period of 2 µs must be followed after clearing the HW_OCOTP_CTRL_BUSY (see Write Postamble).
4. Write the requested address to HW_OCOTP_CTRL_ADDR and program the unlock code into HW_OCOTP_CTRL_WR_UNLOCK. This must be programmed for each write access. The lock code is documented in the register description. Both the unlock code and address can be written in the same operation.
5. Write the data to HW_OCOTP_DATA. This automatically sets HW_OCOTP_CTRL_BUSY and clears HW_OCOTP_CTRL_WR_UNLOCK. In this case, the data is a programming mask. Bit fields with ones will result in that OTP bit being set. Only the controller can clear HW_OCOTP_CTRL_BUSY. The controller will use the mask to program a 32-bit word in the OTP per the address in ADDR. At the same time that the write is accepted, the controller makes an internal copy of

HW_OCOTP_CTRL_ADDR that cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to HW_OCOTP_CTRL_ADDR will not affect an active write operation. It should also be noted that, during programming, HW_OCOTP_DATA will shift right (with zero fill). This shifting is required to program the OTP serially. During the write operation, HW_OCOTP_DATA cannot be modified.

6. Once complete, the controller clears BUSY. Beyond this, the 2-µs postamble requirement must be met before submitting any further OTP operations (see Write Postamble). A write request to a protected region will result in no OTP access and no setting of HW_OCOTP_CTRL_BUSY. In addition, HW_OCOTP_CTRL_ERROR will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are in the order of 10s to 100s of microseconds per word. Write latencies will vary based on the location of the word within the OTP bank. Once a write is initiated, HW_OCOTP_DATA is shifted one bit per every 32 HCLK cycles.

Given:

8 words per OTP bank
32 bits per word
tHCLK is the HCLK clock period
n word locations (where $0 \le n \le 7$)

Then, the approximate write latency for a given word is:

*tHCLK * 32 * 32 * n*

In addition to this latency, software must allow for the 2-µs postamble (using HW_DIGCTL_MICROSECONDS), as described in Write Postamble

20.2.3 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 µs after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes. A recommended software sequence to meet the postamble requirements is as follows:

1. Issue the write and poll for BUSY (as per Software Write Sequence).
2. Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 µs.
3. Perform the next OTP operation.

20.2.4 Shadow Registers and Hardware Capability Bus

The on-chip hardware capability bus is generated using a direct connection to the shadow registers HW_OCOTP_HWCAP0–5 and HW_OCOTP_HWSWCAP. The bits are copied from the OTP on reset. They can be modified until either HW_OCOTP_LOCK_HWSW_SHADOW or HW_OCOTP_LOCK_HWSW_SHADOW_ALT is set. In addition, HW_OCOTP_SWCAP and HW_OCOTP_LOCK are also shadowed into physical registers immediately after reset.

The user can force a reload of the shadow registers (including HW_OCOTP_LOCK) without having to reset the device, which is useful for debugging code. To force a reload:

- Set HW_OCOTP_CTRL_RELOAD_SHADOWS.
- Wait for HW_OCOTP_CTRL_BUSY and HW_OCOTP_CTRL_RELOAD_SHADOWS to be cleared by the controller.
- Attempting to write to the shadow registers while the shadows are being reloaded will result in the setting of HW_OCOTP_CTRL_ERROR. In addition, the register will not take the attempted write (yielding to the reload instead).
- Attempting to write to a shadow register that is locked will result in the setting of HW_OCOTP_CTRL_ERROR.

HW_OCOTP_CTRL_RELOAD_SHADOWS can be set at any time. There is no need to wait for HW_OCOTP_CTRL_BUSY or HW_OCOTP_CTRL_ERROR to be clear.

- In the case of HW_OCOTP_CTRL_BUSY being set due to an active write, the controller will perform the bank opening and shadow reloading immediately after the completion of the write.
- In the case where HW_OCOTP_CTRL_RD_BANK_OPEN is set, the shadow reload will be performed immediately after the banks are closed by the software (by clearing HW_OCOTP_CTRL_RD_BANK_OPEN). It should be noted that BUSY will take approximately 33 HCLK cycles to clear, so polling for HW_OCOTP_CTRL_BUSY immediately after clearing HW_OCOTP_CTRL_RD_BANK_OPEN is not recommended.
- In all cases, the controller will clear HW_OCOTP_CTRL_RELOAD_SHADOWS after the successful completion of the operation.

20.3 Behavior During Reset

The OCOTP is always active. The shadow registers described in Programmable Registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is approximately 32 HCLK cycles after the deassertion of reset. These shadow registers can be reloaded as described in Shadow Registers and Hardware Capability Bus.

20.4 Programmable Registers

OCOTP Hardware Register Format Summary

HW_OCOTP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_C000	OTP Controller Control Register (HW_OCOTP_CTRL)	32	R/W	0000_0000h	20.4.1/1430
8002_C010	OTP Controller Write Data Register (HW_OCOTP_DATA)	32	R/W	0000_0000h	20.4.2/1432
8002_C020	Value of OTP Bank0 Word0 (Customer) (HW_OCOTP_CUST0)	32	R	0000_0000h	20.4.3/1432
8002_C030	Value of OTP Bank0 Word1 (Customer) (HW_OCOTP_CUST1)	32	R	0000_0000h	20.4.4/1433
8002_C040	Value of OTP Bank0 Word2 (Customer) (HW_OCOTP_CUST2)	32	R	0000_0000h	20.4.5/1433
8002_C050	Value of OTP Bank0 Word3 (Customer) (HW_OCOTP_CUST3)	32	R	0000_0000h	20.4.6/1434
8002_C060	Value of OTP Bank0 Word4 (Crypto Key) (HW_OCOTP_CRYPTO0)	32	R	0000_0000h	20.4.7/1434
8002_C070	Value of OTP Bank0 Word5 (Crypto Key) (HW_OCOTP_CRYPTO1)	32	R	0000_0000h	20.4.8/1435
8002_C080	Value of OTP Bank0 Word6 (Crypto Key) (HW_OCOTP_CRYPTO2)	32	R	0000_0000h	20.4.9/1435
8002_C090	Value of OTP Bank0 Word7 (Crypto Key) (HW_OCOTP_CRYPTO3)	32	R	0000_0000h	20.4.10/1436
8002_C0A0	HW Capability Shadow Register 0 (HW_OCOTP_HWCAP0)	32	R/W	0000_0000h	20.4.11/1436
8002_C0B0	HW Capability Shadow Register 1 (HW_OCOTP_HWCAP1)	32	R/W	0000_0000h	20.4.12/1437
8002_C0C0	HW Capability Shadow Register 2 (HW_OCOTP_HWCAP2)	32	R/W	0000_0000h	20.4.13/1437
8002_C0D0	HW Capability Shadow Register 3 (HW_OCOTP_HWCAP3)	32	R/W	0000_0000h	20.4.14/1438
8002_C0E0	HW Capability Shadow Register 4 (HW_OCOTP_HWCAP4)	32	R/W	0000_0000h	20.4.15/1438
8002_C0F0	HW Capability Shadow Register 5 (HW_OCOTP_HWCAP5)	32	R/W	0000_0000h	20.4.16/1439
8002_C100	SW Capability Shadow Register (HW_OCOTP_SWCAP)	32	R/W	0000_0000h	20.4.17/1439

HW_OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_C110	Customer Capability Shadow Register (HW_OCOTP_CUSTCAP)	32	R/W	0000_0000h	20.4.18/1440
8002_C120	LOCK Shadow Register OTP Bank 2 Word 0 (HW_OCOTP_LOCK)	32	R	0000_0000h	20.4.19/1441
8002_C130	Value of OTP Bank2 Word1 (Freescale OPS0) (HW_OCOTP_OPS0)	32	R	0000_0000h	20.4.20/1443
8002_C140	Value of OTP Bank2 Word2 (Freescale OPS1) (HW_OCOTP_OPS1)	32	R	0000_0000h	20.4.21/1444
8002_C150	Value of OTP Bank2 Word3 (Freescale OPS2) (HW_OCOTP_OPS2)	32	R	0000_0000h	20.4.22/1444
8002_C160	Value of OTP Bank2 Word4 (Freescale OPS3) (HW_OCOTP_OPS3)	32	R	0000_0000h	20.4.23/1445
8002_C170	Value of OTP Bank2 Word5 (Unassigned0) (HW_OCOTP_UN0)	32	R	0000_0000h	20.4.24/1445
8002_C180	Value of OTP Bank2 Word6 (Unassigned1) (HW_OCOTP_UN1)	32	R	0000_0000h	20.4.25/1446
8002_C190	Value of OTP Bank2 Word7 (Unassigned2) (HW_OCOTP_UN2)	32	R	0000_0000h	20.4.26/1446
8002_C1A0	Shadow Register for OTP Bank3 Word0 (ROM Use 0) (HW_OCOTP_ROM0)	32	R/W	0000_0000h	20.4.27/1447
8002_C1B0	Shadow Register for OTP Bank3 Word1 (ROM Use 1) (HW_OCOTP_ROM1)	32	R/W	0000_0000h	20.4.28/1449
8002_C1C0	Shadow Register for OTP Bank3 Word2 (ROM Use 2) (HW_OCOTP_ROM2)	32	R/W	0000_0000h	20.4.29/1451
8002_C1D0	Shadow Register for OTP Bank3 Word3 (ROM Use 3) (HW_OCOTP_ROM3)	32	R/W	0000_0000h	20.4.30/1452
8002_C1E0	Shadow Register for OTP Bank3 Word4 (ROM Use 4) (HW_OCOTP_ROM4)	32	R/W	0000_0000h	20.4.31/1453
8002_C1F0	Shadow Register for OTP Bank3 Word5 (ROM Use 5) (HW_OCOTP_ROM5)	32	R/W	0000_0000h	20.4.32/1454
8002_C200	Shadow Register for OTP Bank3 Word6 (ROM Use 6) (HW_OCOTP_ROM6)	32	R/W	0000_0000h	20.4.33/1454
8002_C210	Shadow Register for OTP Bank3 Word7 (ROM Use 7) (HW_OCOTP_ROM7)	32	R/W	0000_0000h	20.4.34/1455
8002_C220	Shadow Register for OTP Bank4 Word0 (Data Use 0) (HW_OCOTP_SRK0)	32	R/W	0000_0000h	20.4.35/1457
8002_C230	Shadow Register for OTP Bank4 Word1 (Data Use 1) (HW_OCOTP_SRK1)	32	R/W	0000_0000h	20.4.36/1457
8002_C240	Shadow Register for OTP Bank4 Word2 (Data Use 2) (HW_OCOTP_SRK2)	32	R/W	0000_0000h	20.4.37/1458
8002_C250	Shadow Register for OTP Bank4 Word3 (Data Use 3) (HW_OCOTP_SRK3)	32	R/W	0000_0000h	20.4.38/1458

HW_OCOTP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_C260	Shadow Register for OTP Bank4 Word4 (Data Use 4) (HW_OCOTP_SRK4)	32	R/W	0000_0000h	20.4.39/1459
8002_C270	Shadow Register for OTP Bank4 Word5 (Data Use 5) (HW_OCOTP_SRK5)	32	R/W	0000_0000h	20.4.40/1459
8002_C280	Shadow Register for OTP Bank4 Word6 (Data Use 6) (HW_OCOTP_SRK6)	32	R/W	0000_0000h	20.4.41/1460
8002_C290	Shadow Register for OTP Bank4 Word7 (Data Use 7) (HW_OCOTP_SRK7)	32	R/W	0000_0000h	20.4.42/1460
8002_C2A0	OTP Controller Version Register (HW_OCOTP_VERSION)	32	R	0105_0000h	20.4.43/1461

20.4.1 OTP Controller Control Register (HW_OCOTP_CTRL)

The OCOTP Control and Status Register specifies the copy state, as well as the control required for random access of the OTP memory

HW_OCOTP_CTRL: 0x000

HW_OCOTP_CTRL_SET: 0x004

HW_OCOTP_CTRL_CLR: 0x008

HW_OCOTP_CTRL_TOG: 0x00C

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR and BUSY/ERROR may be used in conjunction with the HW_OCOTP_DATA register to perform write operations. Read operations are performed through the direct memory mapped registers. In the cases where OTP values are shadowed into local memory storage, the memory mapped location can be read directly. In the cases where the OTP values are not shadowed into local memory, the read-preparation sequence involving RD_BANK_OPEN and BUSY/ERROR fields must be used before performing the read.

Address: HW_OCOTP_CTRL – 8002_C000h base + 0h offset = 8002_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	WR_UNLOCK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2		RELOAD_SHADOWS	RD_BANK_OPEN	RSRVD1		ERROR	BUSY	RSRVD0		ADDR					
W	▨				▨			▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CTRL field descriptions

Field	Description
31–16 WR_UNLOCK	Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). 0x3E77 **KEY** — Key needed to unlock HW_OCOTP_DATA register.
15–14 RSRVD2	These bits always read back zero.
13 RELOAD_ SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically open banks (but will not set RD_BANK_OPEN) and set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller. There is no need to set RD_BANK_OPEN to force the reload. If RD_BANK_OPEN is already set, its still possible to set RELOAD_SHADOWS. In this case, the shadow registers will only be updated upon the clearing of RD_BANK_OPEN.
12 RD_BANK_ OPEN	Set to open the all the OTP banks for reading. When set, the controller sets BUSY to allow time for the banks to become available (approximately 32 HCLK cycles later at which time the controller will clear BUSY). Once BUSY is clear, the various OTP words are accessible through their memory mapped address. Note that OTP words which are shadowed, can be read at anytime and will not be affected by RD_BANK_OPEN. This bit must be cleared after reading is complete. Keeping the OTP banks open causes additional current draw. BUSY must be clear before this setting will take affect. If there is a write transaction pending (holding BUSY), then the bank opening sequence will begin automatically upon the previous transaction clears BUSY. Note that if a read is performed from non-shadowed locations without RD_BANK_OPEN, ERROR will be set
11–10 RSRVD1	These bits always read back zero.
9 ERROR	Set by the controller when either an access to a locked region is requested or a read is requested from non-shadowed efuse locations without the banks being open. Must be cleared before any further write access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or bank open operations (including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete (for writes), or the banks are open (for reads). After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller.
7–6 RSRVD0	These bits always read back zero.
5–0 ADDR	OTP write access address register. Specifies one of 40 word address locations (0x00 - 0x27). If a valid write is accepted by the controller (see HW_OCOTP_DATA for details on what constitutes a valid write), the controller makes an internal copy of this value (to avoid the OTP programming being corrupted). This internal copy will not update until the write access is complete.

20.4.2 OTP Controller Write Data Register (HW_OCOTP_DATA)

The OCOTP Data Register is used for OTP Programming

This register is used in conjunction with HW_OCOTP_CTRL to perform one-time writes to the OTP. See the Software Write Sequence section for operating details.

Address: HW_OCOTP_DATA – 8002_C000h base + 10h offset = 8002_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_DATA field descriptions

Field	Description
31–0 DATA	Used to initiate a write to OTP. To blow a fuse bit, the "1" shall be written through this data register

20.4.3 Value of OTP Bank0 Word0 (Customer) (HW_OCOTP_CUST0)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

Address: HW_OCOTP_CUST0 – 8002_C000h base + 20h offset = 8002_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CUST0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 0 (ADDR = 0x00)

20.4.4 Value of OTP Bank0 Word1 (Customer) (HW_OCOTP_CUST1)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

Address: HW_OCOTP_CUST1 – 8002_C000h base + 30h offset = 8002_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CUST1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 1 (ADDR = 0x01)

20.4.5 Value of OTP Bank0 Word2 (Customer) (HW_OCOTP_CUST2)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

Address: HW_OCOTP_CUST2 – 8002_C000h base + 40h offset = 8002_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CUST2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 2 (ADDR = 0x02)

20.4.6 Value of OTP Bank0 Word3 (Customer) (HW_OCOTP_CUST3)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

Address: HW_OCOTP_CUST3 – 8002_C000h base + 50h offset = 8002_C050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CUST3 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03)

20.4.7 Value of OTP Bank0 Word4 (Crypto Key) (HW_OCOTP_CRYPTO0)

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

Address: HW_OCOTP_CRYPTO0 – 8002_C000h base + 60h offset = 8002_C060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CRYPTO0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). If HW_OCOTP_LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

20.4.8 Value of OTP Bank0 Word5 (Crypto Key) (HW_OCOTP_CRYPTO1)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

Address: HW_OCOTP_CRYPTO1 – 8002_C000h base + 70h offset = 8002_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CRYPTO1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). If HW_OCOTP_LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

20.4.9 Value of OTP Bank0 Word6 (Crypto Key) (HW_OCOTP_CRYPTO2)

OTP banks must be open via HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

Address: HW_OCOTP_CRYPTO2 – 8002_C000h base + 80h offset = 8002_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CRYPTO2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). If HW_OCOTP_LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

20.4.10 Value of OTP Bank0 Word7 (Crypto Key) (HW_OCOTP_CRYPTO3)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

Address: HW_OCOTP_CRYPTO3 – 8002_C000h base + 90h offset = 8002_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CRYPTO3 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). If HW_OCOTP_LOCK[CRYPTOKEY] is set, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR]

20.4.11 HW Capability Shadow Register 0 (HW_OCOTP_HWCAP0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

Address: HW_OCOTP_HWCAP0 – 8002_C000h base + A0h offset = 8002_C0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP0 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 31:0 (copy of OTP bank 1, word 0 (ADDR = 0x08)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.12 HW Capability Shadow Register 1 (HW_OCOTP_HWCAP1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

Address: HW_OCOTP_HWCAP1 – 8002_C000h base + B0h offset = 8002_C0B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP1 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 63:32 (copy of OTP bank 1, word 1 (ADDR = 0x09)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.13 HW Capability Shadow Register 2 (HW_OCOTP_HWCAP2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

Address: HW_OCOTP_HWCAP2 – 8002_C000h base + C0h offset = 8002_C0C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP2 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 95:64 (copy of OTP bank 1, word 2 (ADDR = 0x0A)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.14 HW Capability Shadow Register 3 (HW_OCOTP_HWCAP3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

Address: HW_OCOTP_HWCAP3 – 8002_C000h base + D0h offset = 8002_C0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP3 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 127:96 (copy of OTP bank 1, word 3 (ADDR = 0x0B)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.15 HW Capability Shadow Register 4 (HW_OCOTP_HWCAP4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

Address: HW_OCOTP_HWCAP4 – 8002_C000h base + E0h offset = 8002_C0E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP4 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 159:128 (copy of OTP bank 1, word 4 (ADDR = 0x0C)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.16 HW Capability Shadow Register 5 (HW_OCOTP_HWCAP5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

Address: HW_OCOTP_HWCAP5 – 8002_C000h base + F0h offset = 8002_C0F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_HWCAP5 field descriptions

Field	Description
31–0 BITS	Shadow register for HW capability bits 191:160 (copy of OTP bank 1, word 5 (ADDR = 0x0D)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.17 SW Capability Shadow Register (HW_OCOTP_SWCAP)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

Address: HW_OCOTP_SWCAP – 8002_C000h base + 100h offset = 8002_C100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SWCAP field descriptions

Field	Description
31–0 BITS	Shadow register for SW capability bits 31:0 (copy of OTP bank 1, word 6 (ADDR = 0x0E)). These bits become read-only after the HW_OCOTP_LOCK[HWSW_SHADOW] or HW_OCOTP_LOCK[HWSW_SHADOW_ALT] bit is set.

20.4.18 Customer Capability Shadow Register (HW_OCOTP_CUSTCAP)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

Address: HW_OCOTP_CUSTCAP – 8002_C000h base + 110h offset = 8002_C110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:3]													RTC_XTAL_32768_PRESENT	RTC_XTAL_32000_PRESENT	RSRVD0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_CUSTCAP field descriptions

Field	Description
31–3 RSRVD1	Reserved - do not blow these bits.
2 RTC_XTAL_ 32768_PRESENT	Blow to indicate the presence of an optional 32.768KHz off-chip oscillator.
1 RTC_XTAL_ 32000_PRESENT	Blow to indicate the presence of an optional 32.000KHz off-chip oscillator.
0 RSRVD0	Reserved - do not blow these bits.

20.4.19 LOCK Shadow Register OTP Bank 2 Word 0 (HW_OCOTP_LOCK)

Shadow register for OCOTP Lock Status Value (ADDR = 0x10). Copy of the state of the OTP lock regions. Copied from the OTP upon reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS]

Shadowed memory mapped access to OTP bank 2, word 0 (ADDR = 0x10).

Address: HW_OCOTP_LOCK – 8002_C000h base + 120h offset = 8002_C120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ROM7	ROM6	ROM5	ROM4	ROM3	ROM2	ROM1	ROM0	HWSW_ SHADOW_ALT	CRYPTODCP_ALT	CRYPTOKEY_ALT	PIN	OPS	UN2	UN1	UN0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SRK	UNALLOCATED			SRK_SHADOW	ROM_ SHADOW	CUSTCAP	HWSW	CUSTCAP_ SHADOW	HWSW_ SHADOW	CRYPTODCP	CRYPTOKEY	CUST3	CUST2	CUST1	CUST0
W	▨	▨			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_LOCK field descriptions

Field	Description
31 ROM7	Status of ROM use region write lock bits (ADDR = 0x1F). When set, word 0x1F in the ROM region is locked.
30 ROM6	Status of ROM use region write lock bits (ADDR = 0x1E). When set, word 0x1E in the ROM region is locked.
29 ROM5	Status of ROM use region write lock bits (ADDR = 0x1D). When set, word 0x1D in the ROM region is locked.
28 ROM4	Status of ROM use region write lock bits (ADDR = 0x1C). When set, word 0x1C in the ROM region is locked.

HW_OCOTP_LOCK field descriptions (continued)

Field	Description
27 ROM3	Status of ROM use region write lock bits (ADDR = 0x1B). When set, word 0x1B in the ROM region is locked.
26 ROM2	Status of ROM use region write lock bits (ADDR = 0x1A). When set, word 0x1A in the ROM region is locked.
25 ROM1	Status of ROM use region write lock bits (ADDR = 0x19). When set, word 0x19 in the ROM region is locked.
24 ROM0	Status of ROM use region write lock bits (ADDR = 0x18). When set, word 0x18 in the ROM region is locked.
23 HWSW_ SHADOW_ALT	Status of alternate bit for HWSW_SHADOW lock
22 CRYPTODCP_ ALT	Status of alternate bit for CRYPTODCP lock
21 CRYPTOKEY_ ALT	Status of alternate bit for CRYPTOKEY lock
20 PIN	Status of Pin access lock bit. When set, pin access is disabled.
19 OPS	Status of OPS region (ADDR = 0x11-0x14) write lock bit. When set, region is locked.
18 UN2	Status of un-assigned (ADDR = 0x17) write-lock bit. When set, un-asigned word at OTP address 0x17 is locked.
17 UN1	Status of un-assigned (ADDR = 0x16) write-lock bit. When set, un-asigned word at OTP address 0x16 is locked.
16 UN0	Status of un-assigned (ADDR = 0x15) write-lock bit. When set, un-asigned word at OTP address 0x15 is locked.
15 SRK	Status of SRK bank (ADDR = 0x20-0x27) write-lock bit. When set, the 8 words (bank4) at OTP addresses 0x20-0x27 are locked.
14–12 UNALLOCATED	Value of un-used portion of LOCK word
11 SRK_SHADOW	Status of SRK region shadow register lock. When set, over-ride of DATA-region shadow bits is blocked.
10 ROM_SHADOW	Status of ROM region shadow register lock. When set, over-ride of ROM-region shadow bits is blocked.
9 CUSTCAP	Status of Customer Capability region (ADDR = 0x0F) write lock bit. When set, region is locked.
8 HWSW	Status of HW/SW region (ADDR = 0x08-0x0E) write lock bit. When set, region is locked.
7 CUSTCAP_ SHADOW	Status of Customer Capability shadow register lock. When set, over-ride of customer capabality shadow bits is blocked.

HW_OCOTP_LOCK field descriptions (continued)

Field	Description
6 HWSW_ SHADOW	Status of HW/SW Capability shadow register lock. When set, over-ride of HW/SW capabality shadow bits is blocked.
5 CRYPTODCP	Status of read lock bit for DCP APB crypto access. When set, the DCP will disallow reads of its crypto keys through its APB interface.
4 CRYPTOKEY	Status of crypto key region (ADDR = 0x04-0x07) read/write lock bit. When set, region is locked.
3 CUST3	Status of customer region word (ADDR = 0x03) write lock bit. When set, the region is locked.
2 CUST2	Status of customer region word (ADDR = 0x02) write lock bit. When set, the region is locked.
1 CUST1	Status of customer region word (ADDR = 0x01) write lock bit. When set, the region is locked.
0 CUST0	Status of customer region word (ADDR = 0x00) write lock bit. When set, the region is locked.

20.4.20 Value of OTP Bank2 Word1 (Freescale OPS0) (HW_OCOTP_OPS0)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Shadowed memory mapped access to OTP Bank 2, word 1 (ADDR = 0x11).

Address: HW_OCOTP_OPS0 – 8002_C000h base + 130h offset = 8002_C130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_OPS0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 1 (ADDR = 0x11)

20.4.21 Value of OTP Bank2 Word2 (Freescale OPS1) (HW_OCOTP_OPS1)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Shadowed memory mapped access to OTP Bank 2, word 2 (ADDR = 0x12).

Address: HW_OCOTP_OPS1 – 8002_C000h base + 140h offset = 8002_C140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_OPS1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 2 (ADDR = 0x12)

20.4.22 Value of OTP Bank2 Word3 (Freescale OPS2) (HW_OCOTP_OPS2)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Shadowed memory mapped access to OTP Bank 2, word 3 (ADDR = 0x13).

Address: HW_OCOTP_OPS2 – 8002_C000h base + 150h offset = 8002_C150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_OPS2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 3 (ADDR = 0x13)

20.4.23 Value of OTP Bank2 Word4 (Freescale OPS3) (HW_OCOTP_OPS3)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Shadowed memory mapped access to OTP Bank 2, word 4 (ADDR = 0x14).

Address: HW_OCOTP_OPS3 – 8002_C000h base + 160h offset = 8002_C160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_OPS3 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 4 (ADDR = 0x14), it is used to form the least significant bits of the unique key.

20.4.24 Value of OTP Bank2 Word5 (Unassigned0) (HW_OCOTP_UN0)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 2, word 5 (ADDR = 0x15).

Address: HW_OCOTP_UN0 – 8002_C000h base + 170h offset = 8002_C170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_UN0 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 5 (ADDR = 0x15)

20.4.25 Value of OTP Bank2 Word6 (Unassigned1) (HW_OCOTP_UN1)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 2, word 6 (ADDR = 0x16).

Address: HW_OCOTP_UN1 – 8002_C000h base + 180h offset = 8002_C180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_UN1 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 6 (ADDR = 0x16)

20.4.26 Value of OTP Bank2 Word7 (Unassigned2) (HW_OCOTP_UN2)

OTP banks must be open through HW_OCOTP_CTRL[RD_BANK_OPEN] before reading this register. Reading this register without having HW_OCOTP_CTRL[RD_BANK_OPEN] set and HW_OCOTP_CTRL[BUSY] clear, will result in HW_OCOTP_CTRL[ERROR] being set and 0xBADA_BADA being returned.

Non-shadowed memory mapped access to OTP Bank 2, word 7 (ADDR = 0x17).

Address: HW_OCOTP_UN2 – 8002_C000h base + 190h offset = 8002_C190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BITS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_UN2 field descriptions

Field	Description
31–0 BITS	Reflects value of OTP Bank 2, word 7 (ADDR = 0x17) . It is used to form the unique key

20.4.27 Shadow Register for OTP Bank3 Word0 (ROM Use 0) (HW_OCOTP_ROM0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

Address: HW_OCOTP_ROM0 – 8002_C000h base + 1A0h offset = 8002_C1A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	BOOT_MODE								SD_MMC_ MODE		SD_POWER_ GATE_GPIO		SD_POWER_UP_DELAY [19:16]			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SD_POWER_ UP_DELAY [15:14]		SD_BUS_ WIDTH		SSP_SCK_INDEX				EMMC_USE_DDR	DISABLE_SPI_NOR_ FAST_READ	ENABLE_USB_ BOOT_SERIAL_NUM	ENABLE_ UNENCRYPTED_BOOT	RSRVD0	DISABLE_ RECOVERY_MODE	USE_ALT_ DEBUG_ UART_PINS	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM0 field descriptions

Field	Description
31–24 BOOT_MODE	Encoded boot mode.

HW_OCOTP_ROM0 field descriptions (continued)

Field	Description
23–22 SD_MMC_MODE	SD/MMC BOOT MODE 00 MBR BOOT 01 Reserved 10 eMMC 4.3 Fast Boot 11 eSD2.1 Fast Boot
21–20 SD_POWER_ GATE_GPIO	SD card power gate GPIO pin select: 00 - PWM3, 01 - PWM4, 10 - LCD_DOTCLK, 11 - NO_GATE.
19–14 SD_POWER_UP_ DELAY	SD card power up delay required after enabling GPIO power gate: 000000 - 0 ms, 000001 - 10 ms, 000010 - 20 ms, 111111 - 630 ms.
13–12 SD_BUS_WIDTH	SD card bus width: 00 - 4-bit, 01 - 1-bit, 10 - 8-bit, 11 - reserved.
11–8 SSP_SCK_INDEX	Index to the SSP clock speed 0000 - invalid 0001 - 240kHz 0010 - 1Mhz 0011 - 2Mhz 0100 - 4Mhz 0101 - 6Mhz 0110 - 8Mhz 0111 - 10Mhz 1000 - 12Mhz 1001 - 16Mhz 1010 - 20Mhz 1011 - 30Mhz 1100 - 40Mhz 1101 - 48Mhz 1110 - 51.4Mhz (Max Frequency) 1111 - invalid
7 EMMC_USE_DDR	Blow to operate eMMC4.4 card in DDR mode.
6 DISABLE_SPI_ NOR_FAST_ READ	Blow to disable SPI NOR fast reads which are used by default.
5 ENABLE_USB_ BOOT_SERIAL_ NUM	Blow to enable USB boot serial number.

HW_OCOTP_ROM0 field descriptions (continued)

Field	Description
4 ENABLE_ UNENCRYPTED_ BOOT	Blow to enable unencrypted boot modes.
3 RSRVD0	Reserved - do not blow this bit.
2 DISABLE_ RECOVERY_ MODE	Blow to disable recovery boot mode.
1–0 USE_ALT_ DEBUG_UART_ PINS	00-PWM; 01-AUART0; 10-I2C0; 11-Reserved;

20.4.28 Shadow Register for OTP Bank3 Word1 (ROM Use 1) (HW_OCOTP_ROM1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

Address: HW_OCOTP_ROM1 – 8002_C000h base + 1B0h offset = 8002_C1B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2		SSP3_EXT_PULLUP	SSP2_EXT_PULLUP	ENABLE_NAND7_CE_RDY_PULLUP	ENABLE_NAND6_CE_RDY_PULLUP	ENABLE_NAND5_CE_RDY_PULLUP	ENABLE_NAND4_CE_RDY_PULLUP	ENABLE_NAND3_CE_RDY_PULLUP	ENABLE_NAND2_CE_RDY_PULLUP	ENABLE_NAND1_CE_RDY_PULLUP	ENABLE_NAND0_CE_RDY_PULLUP	UNTOUCH_INTERNAL_SSP_PULLUP	SSP1_EXT_PULLUP	SSP0_EXT_PULLUP	SD_INCREASE_INIT_SEQ_TIME
W	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SD_INIT_SEQ_2_ENABLE	SD_CMD0_DISABLE	SD_INIT_SEQ_1_DISABLE	RSRVD1	BOOT_SEARCH_COUNT				BOOT_SEARCH_STRIDE				RSRVD0	NUMBER_OF_NANDS		
W				▨									▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM1 field descriptions

Field	Description
31–30 RSRVD2	Reserved - do not blow this bit.
29 SSP3_EXT_ PULLUP	Blow to indicate external pull-ups implemented for SSP3.
28 SSP2_EXT_ PULLUP	Blow to indicate external pull-ups implemented for SSP2.
27 ENABLE_ NAND7_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE7 and GPMI_RDY7 pins.
26 ENABLE_ NAND6_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE6 and GPMI_RDY6 pins.
25 ENABLE_ NAND5_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE5 and GPMI_RDY5 pins.
24 ENABLE_ NAND4_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE4 and GPMI_RDY4 pins.
23 ENABLE_ NAND3_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE3 and GPMI_RDY3 pins.
22 ENABLE_ NAND2_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE2 and GPMI_RDY2 pins.
21 ENABLE_ NAND1_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE1 and GPMI_RDY1 pins.
20 ENABLE_ NAND0_CE_ RDY_PULLUP	If the bit is blown then ROM NAND driver will enable internal pull-up for GPMI_CE0 and GPMI_RDY0 pins.
19 UNTOUCH_ INTERNAL_SSP_ PULLUP	If this bit is blown then internal pull-ups for SSP are neither enabled nor disabled. This bit is used only if external pull-ups are implemented and ROM1:18 and/or ROM1:17 are blown.
18 SSP1_EXT_ PULLUP	Blow to indicate external pull-ups implemented for SSP1.

HW_OCOTP_ROM1 field descriptions (continued)

Field	Description
17 SSP0_EXT_ PULLUP	Blow to indicate external pull-ups implemented for SSP0.
16 SD_INCREASE_ INIT_SEQ_TIME	Blow to increase the SD card initialization sequence time from 1ms (default) to 4ms.
15 SD_INIT_SEQ_ 2_ENABLE	Blow to enable the second initialization sequence for SD boot.
14 SD_CMD0_ DISABLE	Cmd0 (reset cmd) is called by default to reset the SD card during startup. Blow this bit to not reset the card during SD boot.
13 SD_INIT_SEQ_ 1_DISABLE	Blow to disable the first initialization sequence for SD.
12 RSRVD1	Reserved - do not blow these bits.
11–8 BOOT_ SEARCH_ COUNT	Number of 64 page blocks that should be read by the boot loader.
7–4 BOOT_ SEARCH_ STRIDE	the value of these bits is multiplied by 64 to get boot search stride. The default is 1 boot search stride, i.e. 64 pages.
3 RSRVD0	Reserved - do not blow these bits.
2–0 NUMBER_OF_ NANDS	Encoded value indicates number of external NAND devices (0 to 7). Zero indicates ROM will probe for the number of NAND devices connected in the system.

20.4.29 Shadow Register for OTP Bank3 Word2 (ROM Use 2) (HW_OCOTP_ROM2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

Address: HW_OCOTP_ROM2 – 8002_C000h base + 1C0h offset = 8002_C1C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	USB_VID																USB_PID															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM2 field descriptions

Field	Description
31–16 USB_VID	USB Vendor ID.
15–0 USB_PID	USB Product ID

20.4.30 Shadow Register for OTP Bank3 Word3 (ROM Use 3) (HW_OCOTP_ROM3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

Address: HW_OCOTP_ROM3 – 8002_C000h base + 1D0h offset = 8002_C1D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:12]				FAST_ BOOT_ACK	ALT_FAST_ BOOT	RSRVD0									
W	▨						▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM3 field descriptions

Field	Description
31–12 RSRVD1	Reserved - do not blow these bits.
11 FAST_BOOT_ ACK	Enable the fast boot acknowledge.

HW_OCOTP_ROM3 field descriptions (continued)

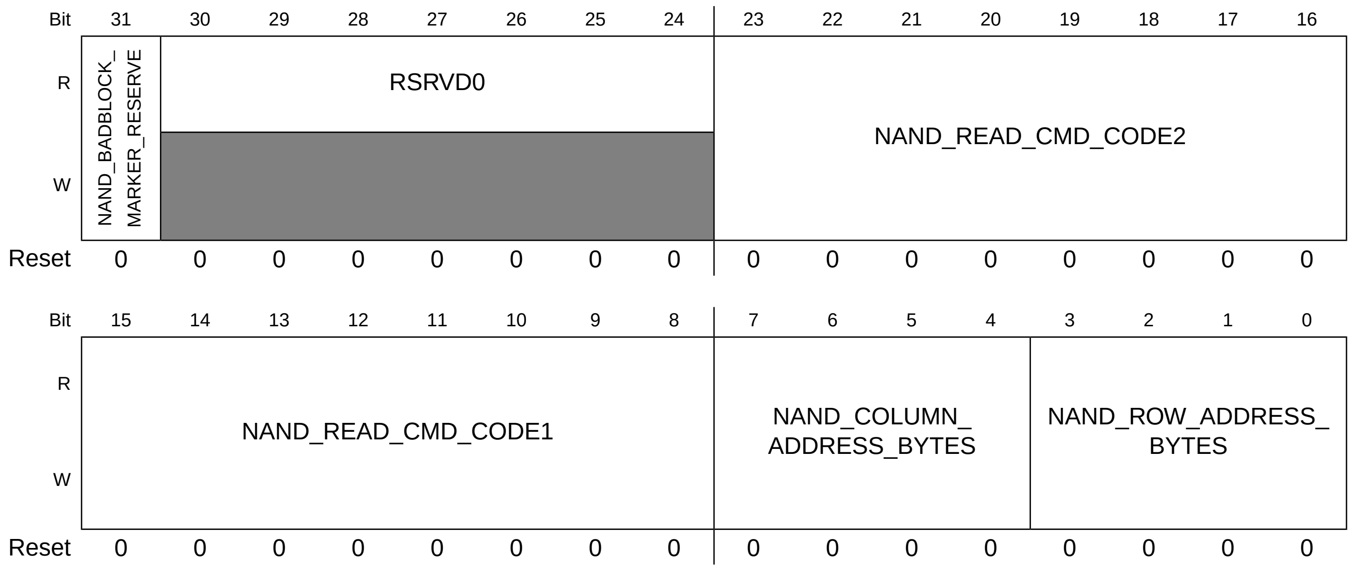
Field	Description
10 ALT_FAST_ BOOT	Enable the alternative fast boot mode.
9–0 RSRVD0	Reserved - do not blow these bits.

20.4.31 Shadow Register for OTP Bank3 Word4 (ROM Use 4) (HW_OCOTP_ROM4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

Address: HW_OCOTP_ROM4 – 8002_C000h base + 1E0h offset = 8002_C1E0h



HW_OCOTP_ROM4 field descriptions

Field	Description
31 NAND_ BADBLOCK_ MARKER_ RESERVE	If this bit is blown, the factory bad block marker preservation will be disabled.
30–24 RSRVD0	Reserved - do not blow these bits.
23–16 NAND_READ_ CMD_CODE2	NAND flash read command confirm code.

HW_OCOTP_ROM4 field descriptions (continued)

Field	Description
15–8 NAND_READ_ CMD_CODE1	NAND flash read command setup code.
7–4 NAND_ COLUMN_ ADDRESS_ BYTES	NAND flash column address cycles.
3–0 NAND_ROW_ ADDRESS_ BYTES	NAND flash row address cycles.

20.4.32 Shadow Register for OTP Bank3 Word5 (ROM Use 5) (HW_OCOTP_ROM5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

Address: HW_OCOTP_ROM5 – 8002_C000h base + 1F0h offset = 8002_C1F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM5 field descriptions

Field	Description
31–0 BITS	Shadow register for ROM-use word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x1D)). These bits become read-only after the HW_OCOTP_LOCK[ROM_SHADOW] bit is set.

20.4.33 Shadow Register for OTP Bank3 Word6 (ROM Use 6) (HW_OCOTP_ROM6)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

Address: HW_OCOTP_ROM6 – 8002_C000h base + 200h offset = 8002_C200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM6 field descriptions

Field	Description
31–0 BITS	Shadow register for ROM-use word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x1E)). These bits become read-only after the HW_OCOTP_LOCK[ROM_SHADOW] bit is set.

20.4.34 Shadow Register for OTP Bank3 Word7 (ROM Use 7) (HW_OCOTP_ROM7)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

Address: HW_OCOTP_ROM7 – 8002_C000h base + 210h offset = 8002_C210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2								FORCE_RECOVERY_DISABLE	ARM_PLL_DISABLE	HAB_CONFIG		RECOVERY_BOOT_MODE [19:16]			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RECOVERY_BOOT_MODE [15:12]				HAB_DISABLE	RSRVD1	RESET_USB_PHY_AT_STARTUP	ENABLE_SSP_12MA_DRIVE	RSRVD0				I2C_USE_400KHZ	ENABLE_ARM_ICACHE	MMU_DISABLE	ENABLE_PIN_BOOT_CHECK
W						▨			▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_ROM7 field descriptions

Field	Description
31–24 RSRVD2	Reserved - do not blow these bits.
23 FORCE_ RECOVERY_ DISABLE	0 - Force Recovery Enable 1 - Force Recovery Disable
22 ARM_PLL_ DISABLE	0 - Not disabled : ARM core will run at 240MHz for high speed boot 1 - Disabled : ARM core will run at xtal 24MHz for boot
21–20 HAB_CONFIG	00 - HAB_FAB : Blank out of FAB 01 - HAB_OPEN : For non-secure shipping products, or secure products under development., the ROM will boot the image even if the HAB authentication fails. Other - HAB_CLOSED : For the end product, the ROM will not boot unless the image passes the HAB authentication.
19–12 RECOVERY_ BOOT_MODE	If USB is not desirable for recovery boot mode then these bits can be blown to have the ROM boot from a non-USB device in recovery mode.
11 HAB_DISABLE	0 - HAB is enabled in default 1 - HAB is disabled
10 RSRVD1	Reserved - do not blow these bits.
9 RESET_USB_ PHY_AT_ STARTUP	Blow to reset the USBPHY at startup This bit will be listed as Reserved in the Data Sheet.
8 ENABLE_SSP_ 12MA_DRIVE	Blow to force SSP pins to drive 12mA, default is 4mA.
7–4 RSRVD0	Reserved - do not blow these bits.
3 I2C_USE_ 400KHZ	Blow to force the I2C to be programmed by the boot loader to run at 400KHz, 100KHz is the default.
2 ENABLE_ARM_ ICACHE	Blow to enable the ARM 926 ICache during boot.
1 MMU_DISABLE	0 - MMU and D-Cache are enabled during boot. 1 - MMU and D-Cache are disabled during boot.
0 ENABLE_PIN_ BOOT_CHECK	Blow to enable boot loader to first test the LCD_RS pin to determine if pin boot mode is enabled. If this bit is blown, and LCD_RS is pulled high, then boot mode is determined by the state of LCD_D[5:0] pins. If this bit is not blown, skip testing the LCD_RS pin and go directly to determining boot mode by reading the state of LCD_D[5:0].

20.4.35 Shadow Register for OTP Bank4 Word0 (Data Use 0) (HW_OCOTP_SRK0)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

Address: HW_OCOTP_SRK0 – 8002_C000h base + 220h offset = 8002_C220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK0 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word0. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 0 (ADDR = 0x20)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.36 Shadow Register for OTP Bank4 Word1 (Data Use 1) (HW_OCOTP_SRK1)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

Address: HW_OCOTP_SRK1 – 8002_C000h base + 230h offset = 8002_C230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK1 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word1. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 1 (ADDR = 0x21)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.37 Shadow Register for OTP Bank4 Word2 (Data Use 2) (HW_OCOTP_SRK2)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

Address: HW_OCOTP_SRK2 – 8002_C000h base + 240h offset = 8002_C240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK2 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word2. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 2 (ADDR = 0x22)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.38 Shadow Register for OTP Bank4 Word3 (Data Use 3) (HW_OCOTP_SRK3)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

Address: HW_OCOTP_SRK3 – 8002_C000h base + 250h offset = 8002_C250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK3 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word3. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 3 (ADDR = 0x23)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.39 Shadow Register for OTP Bank4 Word4 (Data Use 4) (HW_OCOTP_SRK4)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

Address: HW_OCOTP_SRK4 – 8002_C000h base + 260h offset = 8002_C260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK4 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word4. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 4 (ADDR = 0x24)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.40 Shadow Register for OTP Bank4 Word5 (Data Use 5) (HW_OCOTP_SRK5)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 5 (ADDR = 0x25).

Address: HW_OCOTP_SRK5 – 8002_C000h base + 270h offset = 8002_C270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK5 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word5. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 5 (ADDR = 0x25)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.41 Shadow Register for OTP Bank4 Word6 (Data Use 6) (HW_OCOTP_SRK6)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

Address: HW_OCOTP_SRK6 – 8002_C000h base + 280h offset = 8002_C280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK6 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word6. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 6 (ADDR = 0x26)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.42 Shadow Register for OTP Bank4 Word7 (Data Use 7) (HW_OCOTP_SRK7)

Copied from the OTP automatically after reset. Can be re-loaded by setting HW_OCOTP_CTRL[RELOAD_SHADOWS].

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

Address: HW_OCOTP_SRK7 – 8002_C000h base + 290h offset = 8002_C290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_SRK7 field descriptions

Field	Description
31–0 BITS	Shadow register for the hash of the Super Root Key word7. If HAB is not required, these fuses can be used for other purposes (Copy of OTP Bank 4, word 7 (ADDR = 0x27)). These bits become read-only after the HW_OCOTP_LOCK[SRK_SHADOW] bit is set.

20.4.43 OTP Controller Version Register (HW_OCOTP_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

Address: HW_OCOTP_VERSION – 8002_C000h base + 2A0h offset = 8002_C2A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_OCOTP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 21
Performance Monitor (PERFMON)

21.1 Overview

The i.MX28 implements a real time systerm performance monitor, it provides the following functions:

- Real-time performance statistics collection on AXI buses.
- Single or multiple master transactions monitoring.
- Various interrupts support: address trap, latency threshold, and so on.
- AXI protocol and out-of-order sequence support.

The performance monitor is illustrated in Figure 21-1.

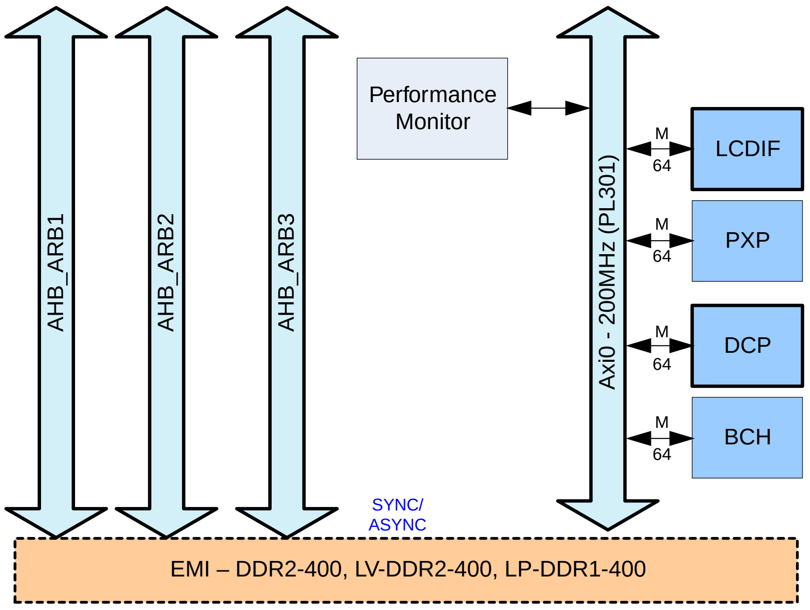


Figure 21-1. Block diagram of PerfMon

21.2 Operation

The role of the performance monitor in an AXI system is to collect real time bus transaction statistics, including transaction average and maximum latency, total data bandwidth, and total active cycles. The statistics collection can be configured for single master or multiple masters on the same AXI bus, and can be enabled for either read or write transactions.

There is one AXI bus in the i.MX28 system which has DCP, PXP, LCDIF and BCH.

AXI protocol supports out-of-order transaction completion. In i.MX28, out-of-order transaction is programmable in EMI. Tag ID is used to trace each transaction and match the data phase with the correct address phase. Performance Monitor supports out-of-order transactions. AXI protocol separates the address/control phase and data phase, and has the ability to issue multiple outstanding addresses. It gives an ID tag to every transaction across the interface, which consists 4-bit master ID and 4-bit subID. Every master has its own unique master ID, and each master could have up to 16 sub IDs, which means up to 16 outstanding address phases.

PerfMon collects the accumulated latency on cycle counts and number of transfers, the average latency can be calculated by dividing the total latency over number of transfers. Since the AXI protocol separates the address and data phases, the address and data phases could be overlapped, and individual data phase could overlap with multiple address phase. Statistics on total cycles, total data transfer in bytes will be collected as well. All statistics are collected for either read or write transactions, which could be switched by the enable bit in the control register.

To calculate the maximum latency and support out-of-order transaction, PerfMon needs to cache the tag ID for all the outstanding requests. The potential number of outstanding requests depends on how deep the command queue in the memory controller(EMI) which includes two level of command queues. A scoreboard will be implemented to match the tag ID of the ongoing data phase with the ID from the outstanding requests. PerfMon snapshots the master ID(8 bits), burst length(4 bits), burst type(2 bits), burst size(3 bits) for the content register of maximum latency, but no transfer address(32 bits).

Registers in PerfMon are accessible through APBH bridge, and PerfMon registers sits on APBH clock domain. Shade registers are added for the coherence among statistics/status registers, and the snapshot bit in PerfMon control register will trigger the shade register being filled at the same cycle for these registers: total transfers, total latency, maximum latency, content register for max_latency, total data bytes, total active cycles, and total cycles.

Various interrupts supported in this module are as follows:

1. Address trap IRQ. IRQ will be asserted if any transaction address hit within the pre-configured address range.

2. Address range inversion IRQ. IRQ will be asserted if any transaction address hit outside the pre-configured address range.

3. Max latency threshold IRQ: IRQ will be asserted if maximum latency is above the value defined in the threshold register.

4. Bus error IRQ: IRQ will be asserted if any AXI transaction triggers an error response.

21.3 Programmable Registers

PERFMON Hardware Register Format Summary

HW_PERFMON memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_6000	PerfMon Control Register (HW_PERFMON_CTRL)	32	R/W	C000_0000h	21.3.1/1466
8000_6010	PerfMon Master Enable Register (HW_PERFMON_MASTER_EN)	32	R/W	0000_0000h	21.3.2/1468
8000_6020	PerfMon Trap Range Low Address Register (HW_PERFMON_TRAP_ADDR_LOW)	32	R/W	0000_0000h	21.3.3/1469
8000_6030	PerfMon Trap Range High Address Register (HW_PERFMON_TRAP_ADDR_HIGH)	32	R/W	0000_0000h	21.3.4/1469
8000_6040	PerfMon Latency Threshold Register (HW_PERFMON_LAT_THRESHOLD)	32	R/W	0000_0000h	21.3.5/1470
8000_6050	PerfMon AXI Active Cycle Count Register (HW_PERFMON_ACTIVE_CYCLE)	32	R	0000_0000h	21.3.6/1470
8000_6060	PerfMon Transfer Count Register (HW_PERFMON_TRANSFER_COUNT)	32	R	0000_0000h	21.3.7/1471
8000_6070	PerfMon Total Latency Count Register (HW_PERFMON_TOTAL_LATENCY)	32	R	0000_0000h	21.3.8/1472
8000_6080	PerfMon Total Data Count Register (HW_PERFMON_DATA_COUNT)	32	R	0000_0000h	21.3.9/1472
8000_6090	PerfMon Maximum Latency Register (HW_PERFMON_MAX_LATENCY)	32	R	0000_0000h	21.3.10/1473
8000_60A0	PerfMon Debug Register (HW_PERFMON_DEBUG)	32	R/W	0000_0001h	21.3.11/1473
8000_60B0	PerfMon Version Register (HW_PERFMON_VERSION)	32	R	0100_0000h	21.3.12/1474

21.3.1 PerfMon Control Register (HW_PERFMON_CTRL)

The PerfMon Control Register specifies the reset state and the interrupt controls for the Performance Monitor.

HW_PERFMON_CTRL: 0x000

HW_PERFMON_CTRL_SET: 0x004

HW_PERFMON_CTRL_CLR: 0x008

HW_PERFMON_CTRL_TOG: 0x00C

This register controls various funcitons of the systerm performance monitor.

EXAMPLE

```
// turn off soft reset and clock gating
HW_PERFMON_CTRL_CLR(BM_PERFMON_CTRL_SFTRST| BM_PERFMON_CTRL_CLKGATE);
```

Address: HW_PERFMON_CTRL – 8000_6000h base + 0h offset = 8000_6000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSVD2						IRQ_MID							
W			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1			BUS_ERR_IRQ	LATENCY_IRQ	TRAP_IRQ	BUS_ERR_IRQ_EN	LATENCY_IRQ_EN	TRAP_IRQ_EN	LATENCY_ENABLE	TRAP_IN_RANGE	TRAP_ENABLE	READ_EN	CLR	SNAP	RUN
W	▨	▨	▨													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_CTRL field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. 0x0 **RUN** — Allow PerfMon to operate normally. 0x1 **RESET** — Hold PerfMon in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow PerfMon to operate normally. 0x1 **NO_CLKS** — Do not clock PerfMon gates in order to minimize power consumption.
29–24 RSVD2	Always set this bit field to zero.

HW_PERFMON_CTRL field descriptions (continued)

Field	Description
23–16 IRQ_MID	This field contains the master ID and sub ID associated with the interrupt. If multiple IRQs, it is the ID associated the first IRQ transaction, and will not update until the IRQ being cleared.
15–13 RSVD1	Always set this bit field to zero.
12 BUS_ERR_IRQ	This bit is set if there is error on any AXI transaction and the BUS_ERR_IRQ_EN bit is set.
11 LATENCY_IRQ	This status bit is set if maximum latency is above the value defined in the latency threshold register and the LATENCY_ENABLE bit is set. Writing a one to its SCT clear address to clear it.
10 TRAP_IRQ	This status bit is set to indicate that an address trap occurs. This bit is cleared by software by writing a one to its SCT clear address. This bit is set if axi address is within the pre-configured address range and the trap function is enabled with TRAP_ENABLE bit.
9 BUS_ERR_IRQ_ EN	Enables the PerfMon AXI BUS ERROR IRQ. When an error occurs and this bit is set, an interrupt is sent to the ARM core.
8 LATENCY_IRQ_ EN	Enables the PerfMon Latency threshold IRQ. When an error occurs and this bit is set, an interrupt is sent to the ARM core.
7 TRAP_IRQ_EN	Enables the PerfMon Address Trap IRQ. When an trap occurs and this bit is set, an interrupt is sent to the ARM core.
6 LATENCY_ ENABLE	Enables the PerfMon AXI latency threshold functions. When a transfer latency larger than the threshold and this bit is set, the LATENCY_IRQ status bit will be set.
5 TRAP_IN_ RANGE	Determines whether the debug trap function causes a match when the master address is inside (low-address <= current-address <= high-address) the specified range. 0 = The trap occurs when the master address falls outside of the range. 1 = The check is inside the range.
4 TRAP_ENABLE	Enables the PerfMon AXI address trap functions. When a trap occurs and this bit is set, the TRAP_IRQ status bit will be set.
3 READ_EN	Set this bit to One to monitor all Read transactions, set to zero to monitor all Write transactions. 0 = performance monitoring on all WRITE activities. 1 = performance monitoring on all READ activities.
2 CLR	Set this bit to clear all the PerfMon's statistics registers. This bit will be reset to 0 once the clear process is done.
1 SNAP	Set this bit to snap shot PerfMon's statistics registers into the shadow registers for reads. PerfMon will continue collecting and updating statistics registers. This bit will be reset to 0 once the snapshot processing is done.
0 RUN	Set this bit to one to enable the PerfMon operation. 0x0 **HALT** — No PerfMon command in progress. 0x1 **RUN** — Process Performance monitoring.

21.3.2 PerfMon Master Enable Register (HW_PERFMON_MASTER_EN)

The Master Enable register defines single or multiple MasterIDs to be monitored by Perfmon. The master ID of PXP is 0, LCDIF is 1, BCH is 2 and DCP is 3.

Address: HW_PERFMON_MASTER_EN – 8000_6000h base + 10h offset = 8000_6010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																MID15	MID14	MID13	MID12	MID11	MID10	MID9	MID8	MID7	MID6	MID5	MID4	MID3	MID2	MID1	MID0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_MASTER_EN field descriptions

Field	Description
31–16 RSVD0	Always set this bit field to zero.
15 MID15	Set to 1 to enable performancce monitoring and statistics collection on MasterID 15.
14 MID14	Set to 1 to enable performancce monitoring and statistics collection on MasterID 14.
13 MID13	Set to 1 to enable performancce monitoring and statistics collection on MasterID 13.
12 MID12	Set to 1 to enable performancce monitoring and statistics collection on MasterID 12.
11 MID11	Set to 1 to enable performancce monitoring and statistics collection on MasterID 11.
10 MID10	Set to 1 to enable performancce monitoring and statistics collection on MasterID 10.
9 MID9	Set to 1 to enable performancce monitoring and statistics collection on MasterID 9.
8 MID8	Set to 1 to enable performancce monitoring and statistics collection on MasterID 8.
7 MID7	Set to 1 to enable performancce monitoring and statistics collection on MasterID 7.
6 MID6	Set to 1 to enable performancce monitoring and statistics collection on MasterID 6.
5 MID5	Set to 1 to enable performancce monitoring and statistics collection on MasterID 5.
4 MID4	Set to 1 to enable performancce monitoring and statistics collection on MasterID 4.
3 MID3	Set to 1 to enable performancce monitoring and statistics collection on MasterID 3.

HW_PERFMON_MASTER_EN field descriptions (continued)

Field	Description
2 MID2	Set to 1 to enable performancce monitoring and statistics collection on MasterID 2.
1 MID1	Set to 1 to enable performancce monitoring and statistics collection on MasterID 1.
0 MID0	Set to 1 to enable performancce monitoring and statistics collection on MasterID 0.

21.3.3 PerfMon Trap Range Low Address Register (HW_PERFMON_TRAP_ADDR_LOW)

The Debug Trap Range Low Address Register defines the lower bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AXI cycle occurs within this range.

Address: HW_PERFMON_TRAP_ADDR_LOW – 8000_6000h base + 20h offset = 8000_6020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_TRAP_ADDR_LOW field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit lower address for the debug trap range.

21.3.4 PerfMon Trap Range High Address Register (HW_PERFMON_TRAP_ADDR_HIGH)

The Trap Range High Address Register defines the upper bound for an address range that can be enabled to trigger an interrupt to the ARM core when an AXI cycle occurs within this range.

Address: HW_PERFMON_TRAP_ADDR_HIGH – 8000_6000h base + 30h offset = 8000_6030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_TRAP_ADDR_HIGH field descriptions

Field	Description
31–0 ADDR	This field contains the 32-bit upper address for the debug trap range.

21.3.5 PerfMon Latency Threshold Register (HW_PERFMON_LAT_THRESHOLD)

The latency threshold Register defines the threshold for AXI transaction latency that can be enabled to trigger an interrupt to the ARM core when an AXI transaction latency is above this value.

Address: HW_PERFMON_LAT_THRESHOLD – 8000_6000h base + 40h offset = 8000_6040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																				VALUE											
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_LAT_THRESHOLD field descriptions

Field	Description
31–12 RSVD0	Always set this bit field to zero.
11–0 VALUE	This field contains the 12-bit transaction latency threshold.

21.3.6 PerfMon AXI Active Cycle Count Register (HW_PERFMON_ACTIVE_CYCLE)

The active cycle count register counts the number of AXI cycles during which a transaction is active on AXI bus.

This register counts the number of AXI cycles in which a master was requesting a transfer, and the slave had not responded. This is including cycles in which it was requesting transfers but was not accepted by slave yet. The master enable register HW_PERFMON_MASTER_EN are used to mask which master(s) cycles are actually recorded here.

Address: HW_PERFMON_ACTIVE_CYCLE – 8000_6000h base + 50h offset = 8000_6050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_ACTIVE_CYCLE field descriptions

Field	Description
31–0 COUNT	This field contains the total active AXI cycle counts. If overflow, the value will be 0xFFFFFFFF.

21.3.7 PerfMon Transfer Count Register (HW_PERFMON_TRANSFER_COUNT)

The transfer count register defines the total number of transfers has been completed during the whole monitoring period.

Divide the register HW_PERFMON_LATENCY_CYCLE value over this value of this register will give the average latency for each transfer.

Address: HW_PERFMON_TRANSFER_COUNT – 8000_6000h base + 60h offset = 8000_6060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_TRANSFER_COUNT field descriptions

Field	Description
31–0 VALUE	This field contains the total amount transfers. If overflow, the value will be 0xFFFFFFFF.

21.3.8 PerfMon Total Latency Count Register (HW_PERFMON_TOTAL_LATENCY)

The transfer count register defines the total number of latency in cycles during the monitoring period.

Divide the register value over value of register HW_PERFMON_TRANSFER_COUNT will give the average latency for each transfer.

Address: HW_PERFMON_TOTAL_LATENCY – 8000_6000h base + 70h offset = 8000_6070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_TOTAL_LATENCY field descriptions

Field	Description
31–0 COUNT	This field contains the total latency in cycle counts. If overflow, the value will be 0xFFFFFFFF.

21.3.9 PerfMon Total Data Count Register (HW_PERFMON_DATA_COUNT)

The data count register defines the total number of data transfered in bytes during the monitoring period.

Address: HW_PERFMON_DATA_COUNT – 8000_6000h base + 80h offset = 8000_6080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_DATA_COUNT field descriptions

Field	Description
31–0 COUNT	This field contains the total bytes of data transfered. If overflow, the value will be 0xFFFFFFFF.

21.3.10 PerfMon Maximum Latency Register (HW_PERFMON_MAX_LATENCY)

The maximum latency register contains the maximum latency in cycles and control signals for this transaction during the monitoring period.

Address: HW_PERFMON_MAX_LATENCY – 8000_6000h base + 90h offset = 8000_6090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ABURST		ALEN				ASIZE			TAGID[22:16]						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TAGID [15:15]	RSVD0			COUNT											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_MAX_LATENCY field descriptions

Field	Description
31–30 ABURST	This field contains axi_aburst signal associated with the worst latency transaction.
29–26 ALEN	This field contains axi_alen signal associated with the worst latency transaction.
25–23 ASIZE	This field contains axi_asize signal associated with the worst latency transaction.
22–15 TAGID	This field contains the master id and sub id associated with the worst latency transaction.
14–12 RSVD0	Always set this bit field to zero.
11–0 COUNT	This field contains the worst transfer latency. If overflow, the value will be 0xFFF.

21.3.11 PerfMon Debug Register (HW_PERFMON_DEBUG)

The debug register is for internal use only.

Divide the register HW_PERFMON_LATENCY_CYCLE value over this value of this register will give the average latency for each transfer.

Address: HW_PERFMON_DEBUG – 8000_6000h base + A0h offset = 8000_60A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:2]														TOTAL_CYCLE_CLR_EN	ERR_MID
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_PERFMON_DEBUG field descriptions

Field	Description
31–2 RSVD	Always set this bit field to zero.
1 TOTAL_CYCLE_ CLR_EN	This field is for debug purpose. Set to 1 to clear the internal total cycle register no matter there is pending request or not when the clear bit in control register is set.
0 ERR_MID	This field is for debug purpose. Set to 0 will not record the MID for bus error irq.

21.3.12 PerfMon Version Register (HW_PERFMON_VERSION)

This register always returns a known read value for debug purposes. It indicates the version of the block.

EXAMPLE

```
if (HW_PERFMON_VERSION.B.MAJOR != 1) Error();
```

Address: HW_PERFMON_VERSION – 8000_6000h base + B0h offset = 8000_60B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PERFMON_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 22
Real-Time Clock Alarm Watchdog Persistent Bits

22.1 RTC Overview

The real-time clock (RTC) and alarm share a one-second pulse time domain. The watchdog reset and millisecond counter run on a one-millisecond time domain. The RTC, alarm, and persistent bits use persistent storage and reside in a special power domain (crystal domain) that remains powered up even when the rest of the chip is in its powered-down state. Figure 22-1 illustrates this block.

Note

The term *power-down*, as used here, refers to a state in which the DC-DC converter and various parts of the crystal power domain are still powered up, but the rest of the chip is powered down. If the battery is removed, then the persistent bits, the alarm value, and the second counter value will be lost. The *crystal power domain* powers both the 32-KHz and 24-MHz crystals.

Upon battery insertion, the crystals (32-KHz and 24-MHz) are in a quiescent state. The activation of these crystals is under software control through the *RTC persistent bits*, as described later in this chapter.

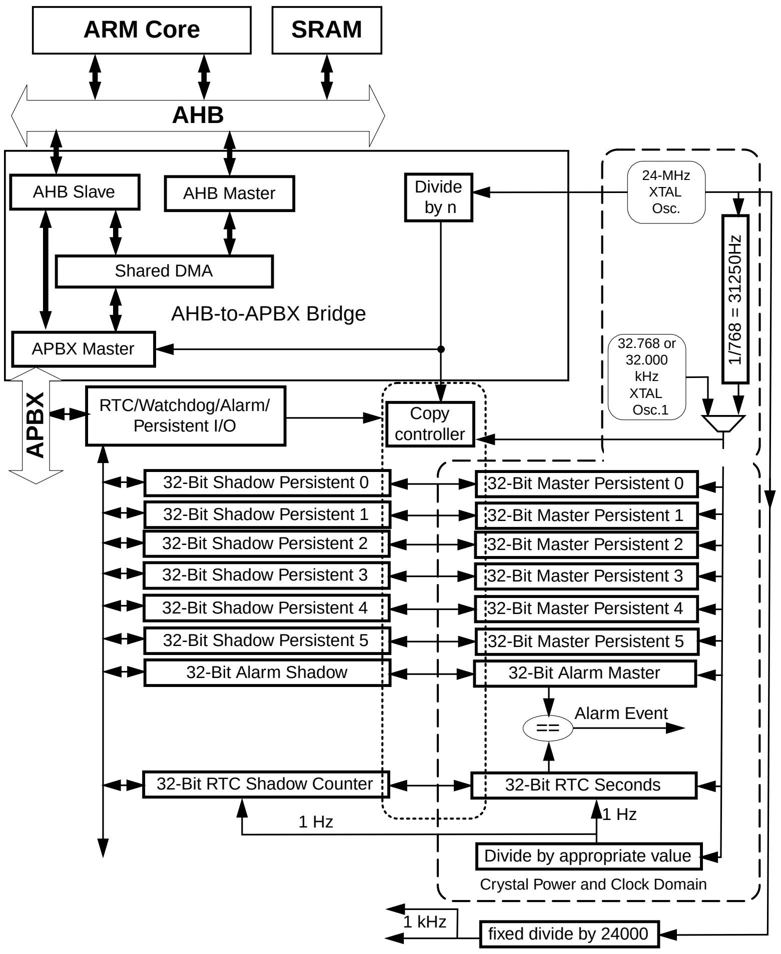
- The XTAL32KHZ_PWRUP bit in the Persistent Register 0 controls the activity of the 32-KHz crystal at all times (chip power on or off).
- The XTAL24MHZ_PWRUP bit in the Persistent Register 0 controls the behavior of the 24-MHz crystal during the power-off state. (The 24-MHz crystal is always on when the chip is powered up.)

The one-second time base is derived either from the 24.0-MHz crystal oscillator or a 32-KHz crystal oscillator (which can be either exactly 32.0 KHz or 32.768 KHz), as controlled by bits in Persistent Register 0. The time base thus generated is used to increment the value of the persistent seconds count register. Like the values of the other persistent registers, the value of the persistent seconds count register is not lost across a power-down state and will continue to count seconds during that time.

Contrary to the one-second time base, no record or count is made of the one-millisecond time base in the crystal power domain. The one-millisecond time base is always derived from the 24.0-MHz crystal oscillator, but is not available when the chip is powered down.

The real-time clock seconds counter, alarm functions, and persistent bit storage are kept in the (always on) crystal power domain. Shadow versions of these values are maintained in the CPU's power and APBX clock domain when the chip is in the power-up state. When the chip transitions from power-off to power-on, the master values are copied to shadow registers by the copy controller. Whenever software writes to a shadow register, then the copy controller copies the new value into the master register in the crystal oscillator power domain.

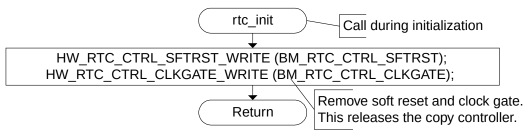
Some of the persistent bits are used to control features that can continue to operate after power-down, such as the second counter and the alarm function and bits in the HW_RTC_PERSISTENT0 register. The bits in registers HW_RTC_PERSISTENT1 through HW_RTC_PERSISTENT5 are available to store application state information over power-downs and are completely software-defined.



ARM Core
SRAM
AHB
AHB Slave
AHB Master
Divide by n
24-MHz XTAL Osc.
Shared DMA
AHB-to-APBX Bridge
APBX Master
1/768 = 31250Hz
32.768 or 32.000 kHz XTAL Osc.1
APBX
RTC/Watchdog/Alarm/ Persistent I/O
Copy controller
32-Bit Shadow Persistent 0
32-Bit Shadow Persistent 1
32-Bit Shadow Persistent 2
32-Bit Shadow Persistent 3
32-Bit Shadow Persistent 4
32-Bit Shadow Persistent 5
32-Bit Alarm Shadow
32-Bit Master Persistent 0
32-Bit Master Persistent 1
32-Bit Master Persistent 2
32-Bit Master Persistent 3
32-Bit Master Persistent 4
32-Bit Master Persistent 5
32-Bit Alarm Master
Alarm Event
==
32-Bit RTC Shadow Counter
32-Bit RTC Seconds
1 Hz
1 Hz
Divide by appropriate value
Crystal Power and Clock Domain
1 kHz
fixed divide by 24000

Figure 22-1. RTC, Watchdog, Alarm, and Persistent Bits Block Diagram

Immediately after reset, it will take approximately three milliseconds for the copy controller to complete the copy process from the analog domain to the digital domain. Software cannot rely on the contents of the seconds counter, alarm, or persistent bits until this copy is complete. Therefore, software must wait until all bits of interest in the HW_RTC_STAT_STALE_REGS field have been reset to 0 by the copy controller before reading the initial state of these values (see Figure 22-2). The order in which registers are updated is Persistent 0, 1, 2, 3, 4, 5, Alarm, Seconds. (This list is in bitfield order, from LSB to MSB, as they would appear in the STALE_REGS and NEW_REGS bitfields of the HW_RTC_STAT register. For example, the Seconds register corresponds to STALE_REGS or NEW_REGS containing 0x80.)



rtc_init
Call during initialization
HW_RTC_CTRL_SFTRST_WRITE (BM_RTC_CTRL_SFTRST);
HW_RTC_CTRL_CLKGATE_WRITE (BM_RTC_CTRL_CLKGATE);
Remove soft reset and clock gate. This releases the copy controller.
Return

Figure 22-2. RTC Initialization Sequence

22.2 Programming and Enabling the RTC Clock

The RTC functions implemented in the crystal power domain are referred to as RTC analog functions. The clock frequency and clock source for the RTC analog functions are programmable. There are three possible clock options.

- If the HW_RTC_PERSISTENT0_CLOCKSOURCE bit is set to 0, these functions operate on a clock domain derived from the 24.0-MHz crystal oscillator divided by 768 to yield 31.250 KHz.
- If the HW_RTC_PERSISTENT0_CLOCKSOURCE bit is 1 and the HW_RTC_PERSISTENT0_XTAL32_FREQ bit is 0, then the optional external driving crystal clock will be used and its frequency should be 32.768 KHz.
- However, if the HW_RTC_PERSISTENT0_CLOCKSOURCE is 1 and the HW_RTC_PERSISTENT0_XTAL32_FREQ bit is also 1, then the external crystal generated clock should be 32.000 KHz for correct operation.

Thus, the HW_RTC_PERSISTENT0_XTAL32_FREQ bit gives the systems designer some flexibility as to which external crystal to use on the board.

Switching between these two clock domains is handled by a glitch-free clock mux and can be done on the fly. The 1-Hz time base is derived by dividing either 32.768 KHz by 32768 or by dividing 31.250 KHz by 31,250, or by dividing 32.000 KHz by 32000.

By reading and examining the HW_RTC_STAT_XTAL32000_PRESENT and HW_RTC_STAT_XTAL32768_PRESENT bits, software can discover if there is an optional crystal clock present and the frequency at which it runs (32.768 KHz or 32.000 KHz). Only one of these fuse bits will be asserted if there is such a crystal attached. If there is no crystal present, both bits will be deasserted.

22.3 RTC Persistent Register Copy Control

The copying of a persistent shadow register (digital) to persistent master storage (analog) occurs automatically. This automatic write-back that occurs for each register as the copy controller services writes to the shadow registers can lead to some very long timing loops if efficient write procedures are not used. Writing all eight shadow registers can take several milliseconds to complete. Do not attempt to write to more than one shadow register immediately before power down. Whenever possible, software should ensure that the HW_RTC_STAT_NEW_REGS field is 0 before powering down the chip or setting the

HW_RTC_CTRL_SFTRST bit. Otherwise, some of the write data could be lost because the shadow registers could be powered down/reset before the new values can be copied to the persistent master storage.

Registers are copied between the digital and analog sides one by one and in 32-bit words. There are no hardwired uses for any of the bits of Persistent registers 1 through 5. Therefore, the bits in these registers can be defined and set by the software. Persistent Register 0 is reserved for hardware programming and configuration.

Before a new value is written to a shadow register by the CPU, software must first confirm that the corresponding bit of HW_RTC_STAT_NEW_REGS is 0. This ensures that a value previously written to the register has been completely handled by the copy state machine. Failure to obey this constraint could cause a newer updated value to be lost.

NOTE: The HW_RTC_CTRL_SUPPRESS_COPY2ANALOG diagnostic bit is never set while any copy or update operation is underway. Doing so will result in undefined operation of the copy controller.

Figure 22-3 shows the copy and test procedure for a single register. However, software can write to any register whose HW_RTC_STAT_NEW_REGS bit is 0 even if the copy controller is currently busy copying a different register. For example, if the copy controller is busy copying Persistent Register 1 from a previous write, software can simultaneously write to Persistent Register 0 during this copy. After the copy controller has finished copying register 1, it will then, in turn, begin the copy process for the new value of Register 0. Therefore, registers can thus be written in any order. Again, the main important rule that must be followed is that the HW_RTC_STAT_NEW_REGS bit for a particular register must be 0 before it can be written and is independent of the state of the HW_RTC_STAT_NEW_REGS bits for the other registers.

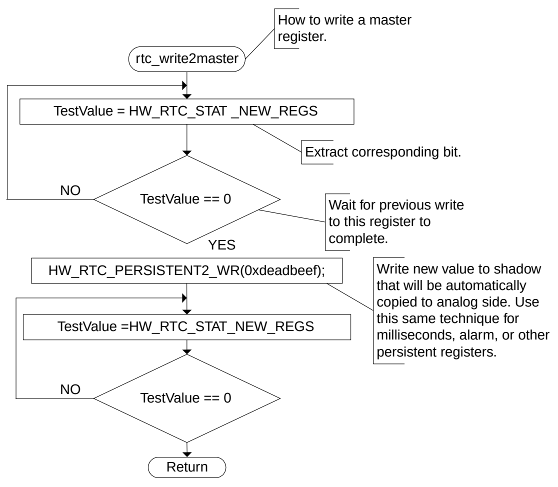

How to write a master register.
rtc_write2master
TestValue = HW_RTC_STAT _NEW_REGS
Extract corresponding bit.
NO
TestValue == 0
Wait for previous write to this register to complete.
YES
HW_RTC_PERSISTENT2_WR(0xdeadbeef);
Write new value to shadow that will be automatically copied to analog side. Use this same technique for milliseconds, alarm, or other persistent registers.
TestValue =HW_RTC_STAT_NEW_REGS
NO
TestValue == 0
Return

Figure 22-3. RTC Writing to a Master Register from CPU

The digital shadow registers will be updated (copied from analog to digital) with values from their analog persistent counterparts under two conditions: first, whenever the chip is powered on, and second, whenever the HW_RTC_CTRL_FORCE_UPDATE bit is set by the software. Then, an update occurs, and all persistent registers are updated. Persistent registers cannot be updated singly.

22.4 Real-Time Clock Function

The real-time clock is a CPU-accessible, continuously-running 32-bit counter that increments every second and that can be derived from either the 24-MHz clock or the 32-KHz clock, as determined by writable bit values in the RTC Persistent Register 0.

A 32-bit second counter has enough resolution to count up to 136 years with one-second increments. The RTC can continue to count time as long as a voltage is applied to the BATT pin, irrespective of whether the rest of the chip is powered up. The normal digital reset has no effect on the master RTC registers located in the crystal power and clock domain. A special first-power-on reset establishes the default value of the master RTC registers when a voltage is first applied to the BATT pin (battery insertion).

For consistency across applications, it is recommended that the seconds timer should be referenced to January 1, 1980 at a 32-bit value of 0 (same epoch reference as PC) in applications that use it as a time-of-day clock. If the real-time clock function is not present on a specific chip, as indicated in the control and status register (HW_RTC_STAT_RTC_PRESENT), then no real-time epoch is maintained over power-down cycles.

22.4.1 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

22.5 Millisecond Resolution Timing Function

A millisecond counter facility is provided based on a 1-KHz signal derived from the 24-MHz clock. The count value is neither maintained nor incremented during power-down cycles. At each power-up, this register is set to its reset state. On each tick of the 1-KHz source, the milliseconds counter increments. With a 32-bit counter, a kernel can run up to

4,294,967,294 milliseconds or 49.7 days before it must deal with a counter wrap. The programmer can change the resolution of the millisecond counter to be 1, 2, 4, 8, or 16 milliseconds. This is programmed through bits in Persistent Register 0.

CAUTION: When the 32.768-KHz or 32.000-KHz crystal oscillator is selected as the source for the seconds counter, an anomaly is created between the time intervals of the millisecond counter and the seconds counter. That is, the manufacturing tolerance of the two crystals are such that 1000 millisecond counter increments are not exactly one second as measured by the real-time clock seconds counter.

22.6 Alarm Clock Function

The alarm clock function allows an application to specify a future instant at which the chip should be awakened, that is, if powered down, it can be powered up. The alarm clock setting is a CPU-accessible, 32-bit value that is continuously matched against the 32-bit real-time clock seconds counter. When the two values are equal, an alarm event is triggered. Persistent bits indicate whether an alarm event should power up the chip from its powered-down state. In addition to or instead of powering up the chip, the alarm event can also cause a CPU interrupt. Although these two functions can be enabled at the same time, one should remember that the CPU will only be interrupted if the chip is powered up at the time of the alarm event.

NOTE: If the alarm is set to power up the chip in the event of an alarm and such an event occurs, then the only record of the cause of the wake-up is located in the analog side. At power-up, the analog side registers are copied to the digital shadow registers and the ALARM_WAKE bit in the Persistent register 0 is visible in the digital shadow register. If an alarm wake event occurs while the chip is powered up, the ALARM_WAKE bit will not be set in the persistent register because the chip was not woken up.

The alarm must be present on an actual chip to perform this function (see the HW_RTC_STAT_ALARM_PRESENT bit description).

22.7 Watchdog Reset Function

The watchdog reset is a CPU-configurable device. It is programmed by software to generate a chip-wide reset after HW_RTC_WATCHDOG milliseconds. The watchdog generates this reset if software does not rewrite this register before this time elapses.

The watchdog timer decrements the register value once for every tick of the 1-KHz clock supplied from the RTC analog section (see Figure 22-1). The reset generated by the watchdog timer has no effect on the values retained in the master registers of the real-time clock seconds counter, alarm, or persistent registers (analog persistent storage).

The watchdog timer is initially disabled and set to count 4,294,967,295 milliseconds before generating a watchdog reset.

The watchdog timer does not run when the chip is in its powered-down state. Therefore, there is no master/shadow register pairing for the watchdog timer, and it must be reprogrammed after cycling power or resetting the block.

The watchdog timer must be "present" on an actual chip to perform this function (see the HW_RTC_STAT_WATCHDOG_PRESENT bit description).

22.8 Programmable Registers

RTC Hardware Register Format Summary

HW_RTC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8005_6000	Real-Time Clock Control Register (HW_RTC_CTRL)	32	R/W	C000_0020h	22.8.1/1485
8005_6010	Real-Time Clock Status Register (HW_RTC_STAT)	32	R	E0FF_0000h	22.8.2/1486
8005_6020	Real-Time Clock Milliseconds Counter (HW_RTC_MILLISECONDS)	32	R/W	0000_0000h	22.8.3/1488
8005_6030	Real-Time Clock Seconds Counter (HW_RTC_SECONDS)	32	R/W	0000_0000h	22.8.4/1489
8005_6040	Real-Time Clock Alarm Register (HW_RTC_ALARM)	32	R/W	0000_0000h	22.8.5/1490
8005_6050	Watchdog Timer Register (HW_RTC_WATCHDOG)	32	R/W	FFFF_FFFFh	22.8.6/1491
8005_6060	Persistent State Register 0 (HW_RTC_PERSISTENT0)	32	R/W	0000_0100h	22.8.7/1492
8005_6070	Persistent State Register 1 (HW_RTC_PERSISTENT1)	32	R/W	0000_0000h	22.8.8/1494
8005_6080	Persistent State Register 2 (HW_RTC_PERSISTENT2)	32	R/W	0000_0000h	22.8.9/1495
8005_6090	Persistent State Register 3 (HW_RTC_PERSISTENT3)	32	R/W	0000_0000h	22.8.10/1496
8005_60A0	Persistent State Register 4 (HW_RTC_PERSISTENT4)	32	R/W	0000_0000h	22.8.11/1497
8005_60B0	Persistent State Register 5 (HW_RTC_PERSISTENT5)	32	R/W	0000_0000h	22.8.12/1497
8005_60C0	Real-Time Clock Debug Register (HW_RTC_DEBUG)	32	R/W	0000_0000h	22.8.13/1498
8005_60D0	Real-Time Clock Version Register (HW_RTC_VERSION)	32	R	0203_0000h	22.8.14/1499

22.8.1 Real-Time Clock Control Register (HW_RTC_CTRL)

HW_RTC_CTRL is the control register for the RTC.

HW_RTC_CTRL: 0x000

HW_RTC_CTRL_SET: 0x004

HW_RTC_CTRL_CLR: 0x008

HW_RTC_CTRL_TOG: 0x00C

The contents of this register control the operation of the RTC portions implemented as an APBX peripheral running in the APBX clock domain. These functions only operate when the chip is in its full power up state.

EXAMPLE

```
            HW_RTC_CTRL_CLR(BM_RTC_CTRL_SFTRST);  // remove the soft reset condition
            HW_RTC_CTRL_CLR(BM_RTC_CTRL_CLKGATE); // enable clocks within the RTC
           HW_RTC_CTRL_CLR(BM_RTC_CTRL_ALARM_IRQ); // reset the alarm interrupt by clearing
 its status bit
```

Address: HW_RTC_CTRL – 8005_6000h base + 0h offset = 8005_6000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSVD0[29:16]													
W			▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0[15:7]									SUPPRESS_ COPY2ANALOG	FORCE_UPDATE	WATCHDOGEN	ONEMSEC_IRQ	ALARM_IRQ	ONEMSEC_IRQ_ EN	ALARM_IRQ_EN
W	▨	▨	▨	▨	▨	▨	▨	▨	▨							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

HW_RTC_CTRL field descriptions

Field	Description
31 SFTRST	1= Hold real time clock digital side in soft reset state. This bit has no effect on the RTC analog.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. This bit has no effect on RTC analog.

HW_RTC_CTRL field descriptions (continued)

Field	Description
29–7 RSVD0	Reserved, write only zeroes.
6 SUPPRESS_ COPY2ANALOG	This bit is used for diagnostic purposes. 1= suppress the automatic copy that normally occurs to the analog side, whenever a shadow register is written. 0= Normal operation. Use SCT writes to set, clear, or toggle.
5 FORCE_UPDATE	This bit is how the software requests the update of the shadow registers from values in RTC analog. When software sets this bit, all eight of the shadow registers are updated from the corresponding values in the persistent registers in RTC analog. The state of this update operation is reflected on the STALEREGS bits on the STAT register, which are set to all ones upon an update request and are cleared by hardware as the update proceeds. Hardware clears this bit immediately after detecting it has been set. Software does not need to clear. Software must NOT look to the state of this bit to determine the status of the update operation, it must look to the STALEREG bits in the STAT register to determine when any given register has been updated and/or when the update operation is complete. Notice that the default value of this bit is 1, so that that a reset (either chip-wide or soft) always results in an update.
4 WATCHDOGEN	1= Enable Watchdog Timer to force chip wide resets. Use SCT writes to set, clear, or toggle.
3 ONEMSEC_IRQ	1= one millisecond interrupt request status. Use SCT writes to clear this interrupt status bit.
2 ALARM_IRQ	1= Alarm Interrupt Status. Use SCT writes to clear this interrupt status bit.
1 ONEMSEC_IRQ_ EN	1= Enable one millisecond interrupt. Use SCT writes to set, clear, or toggle.
0 ALARM_IRQ_EN	1= Enable Alarm Interrupt. Use SCT writes to set, clear, or toggle.

22.8.2 Real-Time Clock Status Register (HW_RTC_STAT)

HW_RTC_STAT is the status register for the RTC.

HW_RTC_STAT: 0x010

HW_RTC_STAT_SET: 0x014

HW_RTC_STAT_CLR: 0x018

HW_RTC_STAT_TOG: 0x01C

This register reflects the current state of the RTC in terms of its enabled capabilities and the state of the persistent registers.

EXAMPLE

```
while(HW_RTC_STAT.STALE_REGS !=0)
{
```

```
                printf(" something is stale in one of the digital side registers\n");
                // the copy controller will copy analog registers to digtial registers as
required,
                // turning off staleregs bits as it goes about its business.
             }
             if(HW_RTC_STAT.WATCHDOG_PRESENT != 0) // then you can use the watchdog timer on
 this chip
```

Address: HW_RTC_STAT – 8005_6000h base + 10h offset = 8005_6010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RTC_PRESENT	ALARM_PRESENT	WATCHDOG_PRESENT	XTAL32000_PRESENT	XTAL32768_PRESENT	RSVD1			STALE_REGS							
W																
Reset	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NEW_REGS								RSVD0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_STAT field descriptions

Field	Description
31 RTC_PRESENT	This read-only bit reads back a one if the RTC is present in the device.
30 ALARM_ PRESENT	This read-only bit reads back a one if the Alarm function is present in the device.
29 WATCHDOG_ PRESENT	This read-only bit reads back a one if the Watchdog Timer function is present in the device.
28 XTAL32000_ PRESENT	This read-only bit reads back a one if the 32.000-kHz crystal oscillator function is present in the device.
27 XTAL32768_ PRESENT	This read-only bit reads back a one if the 32.768-kHz crystal oscillator function is present in the device.

HW_RTC_STAT field descriptions (continued)

Field	Description
26–24 RSVD1	Reserved, write only zeroes.
23–16 STALE_REGS	These read-only bits are set to one whenever the corresponding shadow register contents are older than the analog side contents. These bits are set by reset and cleared by the copy controller. They are also set by writing a one to the FORCE_UPDATE bit.
15–8 NEW_REGS	These read-only bits are set to one whenever the corresponding shadow register contents are newer than the analog side contents. These bits are set by writing to the corresponding register and cleared by the copy controller.
7–0 RSVD0	Reserved, write only zeroes.

22.8.3 Real-Time Clock Milliseconds Counter (HW_RTC_MILLISECONDS)

The millisecond count register provides a reliable elapsed time reference to the kernel with millisecond resolution.

HW_RTC_MILLISECONDS: 0x020

HW_RTC_MILLISECONDS_SET: 0x024

HW_RTC_MILLISECONDS_CLR: 0x028

HW_RTC_MILLISECONDS_TOG: 0x02C

HW_RTC_MILLISECONDS provides access to the 32-bit millisecond counter. This counter is not a shadow register, i.e., the contents of this register are not preserved over power-down states. This counter increments once per millisecond based on a pulse from RTC analog which is derived from the 24.0-MHz crystal clock. This 1-kHz source hence does not vary as the APBX clock frequency is changed. The millisecond counter wraps at 4,294,967,294 milliseconds or 49.7 days.

EXAMPLE

```
             HW_RTC_MILLISECONDS_WR(0);        // write an initial starting value to the
milliseconds counter
             Count = HW_RTC_MILLISECONDS_RD(); // read the current value of the milliseconds
counter.
```

Address: HW_RTC_MILLISECONDS – 8005_6000h base + 20h offset = 8005_6020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_MILLISECONDS field descriptions

Field	Description
31–0 COUNT	32 bit milliseconds counter.

22.8.4 Real-Time Clock Seconds Counter (HW_RTC_SECONDS)

The real-time clock seconds counter is used to maintain real time for applications, even across certain chip power-down states.

HW_RTC_SECONDS: 0x030

HW_RTC_SECONDS_SET: 0x034

HW_RTC_SECONDS_CLR: 0x038

HW_RTC_SECONDS_TOG: 0x03C

HW_RTC_SECONDS provides access to the 32-bit real-time seconds counter. Both the shadow register on the digital side and the analog side register update every second. When the chip enters the power-down state, the shadow register is powered down and loses its state value. When the chip powers up, the analog side register contents are automatically copied to the shadow register. The reset value of 0x0 for the digital side register is only visible until the copy controller overwrites with the value from the analog side. The analog side register resets to zero only upon power-on reset (POR), i.e., when the battery is first inserted or whenever a battery-less part is plugged into USB power or into a wall transformer. Note that if a low frequency (32.000khz or 32.768khz) crystal is available in the system, then the seconds count and the milliseconds count will be derived from different clocks. Namely, the msec count is derived from the 24MHz crystal and the seconds count from the low-frequency crystal. This limits the precision of the relation between these two clocks.

EXAMPLE

```
             HW_RTC_SECONDS_WR(0);           // write an initial value to the digital side.
This value will
                                             // be automatically copied to the analog side
             rt_clock = HW_RTC_SECONDS_RD(); // read the 32 seconds counter value
```

Address: HW_RTC_SECONDS – 8005_6000h base + 30h offset = 8005_6030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_SECONDS field descriptions

Field	Description
31–0 COUNT	Increments once per second.

22.8.5 Real-Time Clock Alarm Register (HW_RTC_ALARM)

The 32-bit alarm value is matched against the 32-bit seconds counter to detect an alarm condition.

HW_RTC_ALARM: 0x040

HW_RTC_ALARM_SET: 0x044

HW_RTC_ALARM_CLR: 0x048

HW_RTC_ALARM_TOG: 0x04C

The 32-bit alarm value can be used to awaken the chip from a power-down state or simply to cause an interrupt at a specific time. When the chip enters the power-down state, the shadow register is powered down and loses its state value. When the chip powers up, the analog side register contents are automatically copied to the shadow register. The reset value of 0x0 for the digital side register is only visible until the copy controller overwrites with the value from the analog side. The analog side register resets to zero only upon power-on reset (POR), i.e., when the battery is first inserted or whenever a battery-less part is plugged into a power USB or into a wall transformer.

EXAMPLE

```
HW_RTC_ALARM_WR(60);  // generate rtc alarm after 60 seconds
```

Address: HW_RTC_ALARM – 8005_6000h base + 40h offset = 8005_6040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_ALARM field descriptions

Field	Description
31–0 VALUE	Seconds match-value used to trigger assertion of the RTC alarm.

22.8.6 Watchdog Timer Register (HW_RTC_WATCHDOG)

The 32-bit watch dog timer can be used to reset the chip if enabled and not adequately serviced.

HW_RTC_WATCHDOG: 0x050

HW_RTC_WATCHDOG_SET: 0x054

HW_RTC_WATCHDOG_CLR: 0x058

HW_RTC_WATCHDOG_TOG: 0x05C

The 32-bit watchdog timer will reset the chip upon decrementing to zero, if this function is enabled and present on the chip. The 1-kHz source is based on a msec pulse from RTC analog which is derived from the 24-MHz crystal oscillator and hence does not vary when the APBX clock is changed.

EXAMPLE

```
            HW_RTC_WATCHDOG_WR(10000);  // reload the watchdog and keep it from resetting the
chip
```

Address: HW_RTC_WATCHDOG – 8005_6000h base + 50h offset = 8005_6050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNT																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_RTC_WATCHDOG field descriptions

Field	Description
31–0 COUNT	If the watchdog timer decrements to zero and the watchdog timer reset is enabled, then the chip will be reset. The watchdog timer decrements once per millisecond, when enabled.

22.8.7 Persistent State Register 0 (HW_RTC_PERSISTENT0)

The 32-bit persistent registers are used to retain certain control states during chip wide power-down states.

HW_RTC_PERSISTENT0: 0x060

HW_RTC_PERSISTENT0_SET: 0x064

HW_RTC_PERSISTENT0_CLR: 0x068

HW_RTC_PERSISTENT0_TOG: 0x06C

The general persistent bits are available for software use. The register initalizes to a known reset pattern. The copy controller overwrites the digital reset values very soon after power on, but not in zero time.

EXAMPLE

```
             HW_RTC_PERSISTENT0_SET(BM_RTC_PERSISTENT0_ALARM_WAKE_EN); // wake up the chip if
 the alarm event occurs
             HW_RTC_PERSISTENT0_SET(BM_RTC_PERSISTENT0_CLOCKSOURCE);   // select the 32KHz
oscillator as the source for the RTC analog clock
```

Address: HW_RTC_PERSISTENT0 – 8005_6000h base + 60h offset = 8005_6060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ADJ_POSLIMITBUCK				RSVD1						EXTERNAL_RESET	THERMAL_RESET	RSVD0	ENABLE_ LRADC_PWRUP	AUTO_RESTART	DISABLE_PSWITCH
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOWERBIAS		DISABLE_ XTALOK	MSEC_RES					ALARM_WAKE	XTAL32_FREQ	XTAL32KHZ_ PWRUP	XTAL24MHZ_ PWRUP	LCK_ SECS	ALARM_EN	ALARM_WAKE_ EN	CLOCKSOURCE
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT0 field descriptions

Field	Description
31–28 ADJ_ POSLIMITBUCK	This field can be used to allow dcdc startup with lower Liion battery voltages. With the default value of 0x0, the dcdc converter will not startup below 2.83V, and increasing this value will lower the Liion startup voltage. It is not recommended to set a value higher than 0x6. The value of this field effects the minimum startup voltage as follows: 0x0=2.83V, 0x1=2.78V, 0x2=2.73V, 0x3=2.68V, 0x4=2.62V, 0x5=2.57V, 0x6=2.52V, 0x7-0xF=2.48V.
27–22 RSVD1	Reserved for special uses.
21 EXTERNAL_ RESET	This bit is set by the analog hardware. On powerup, it indicate to software that that the chip had previously been powered down due to a reset event on the reset pin by an external source. This bit must be cleared by software. Note this bit is only valid when 32K XTAL is selected as the clock source of RTC, otherwise the external reset event might not be correctly captured when the reset happens.
20 THERMAL_ RESET	This bit is set by the analog hardware. On powerup, it indicate to software that that the chip had previously been powered down due to overheating. This bit must be cleared by software. This bit is analogous to the HW_POWER_STS_THERMAL_WARNING bit only in persistent storage. Note this bit is only valid when 32K XTAL is selected as the clock source of RTC, otherwise the thermal reset event might not be correctly captured when the reset happens.
19 RSVD0	Reserved for special uses.
18 ENABLE_ LRADC_PWRUP	Set to one to enable a low voltage on LRADC0 to powerup the chip. This is to support powerup through open drain pulldowns connected to LRADC0. Ths requires external circuitry and this bit should not be set unless the correct circuitry is present. Freescale will provide information on the external circuitry, if this functionality is required.
17 AUTO_RESTART	Set to one to enable the chip to automatically power up approximately 180 ms after powering down.
16 DISABLE_ PSWITCH	Disables the pswitch pin startup functionality unless the voltage on the pswitch pin goes above the VDDXTAL pin voltage by a threshold voltage. Typically, this voltage is created by pulling pswitch up with a current limiting resistor to a higher voltage, such as the Liion battery voltage.
15–14 LOWERBIAS	Reduce bias current of 24mhz crystal. b00: nominal, b01: -25%, b10: -25%, b11: -50%,
13 DISABLE_ XTALOK	Set to one to disable the circuit that resets the chip if 24-MHz frequency falls below 2 MHz. The circuit defaults to enabled and will power down the device if the 24-MHz stop oscillating for any reason.

HW_RTC_PERSISTENT0 field descriptions (continued)

Field	Description
12–8 MSEC_RES	This bit field encodes the value of the millisecond count resolution in a one-hot format. Resolutions supported are: 1, 2, 4, 8, and 16 msec. The bitfield directly gives the resolution without need for decode.
7 ALARM_WAKE	Set when the chip is powered up by an alarm event from rtc_ana. Can then be cleared by software as desired.
6 XTAL32_FREQ	If CLOCKSOURCE (bit 0 of this register) is one, then this bit gives the exact frequency of the 32kHz crystal. If this bit is zero, the frequency is 32.768kHZ, if it is one, the frequency is 32.000kHz. If CLOCKSOURCE is zero, the value of this bit is immaterial.
5 XTAL32KHZ_ PWRUP	Set to one to power up the 32kHz crystal oscillator. Set to zero to disable the oscillator. This bit controls the oscillator at all times (chip powered on or not).
4 XTAL24MHZ_ PWRUP	Set to one to keep the 24.0-MHz crystal oscillator powered up while the chip is powered down. Set to zero to disable during chip power down. Note: The oscillator is always on while the chip is powered on.
3 LCK_SECS	Set to one to lock down the seconds count. Once this bit is written with a 1, the user will not be able to either write to the seconds register or to change this bit back to a zero -- except by removing the battery.
2 ALARM_EN	Set this bit to one to enable the detection of an alarm event. This bit must be turned on before an alarm event can awaken a powered-down device, or before it can generate an alarm interrupt to a powered-up CPU. When the alarm is not present in the device (as indicated by the fuse bits) the copy of shadow0 to persistent0 will not allow bit[2] to be written and persistent bit[2] will always read back 0, regardless of the values in the shadow register.
1 ALARM_WAKE_ EN	This bit is set to one to upon the arrival of an alarm event that powers up the chip. ALARM_EN must be set to one to enable the detection of an alarm event. This bit is reset by writing a zero directly to the shadow register, which causes the copy controller to move it across to the analog domain.
0 CLOCKSOURCE	Set to one to select the 32-kHz crystal oscillator as the source for the 32-kHz clock domain used by the RTC analog domain circuits. Set to zero to select the 24-MHz crystal oscillator as the source for generating the 32-kHz clock domain used by the RTC analog domain circuits.

22.8.8 Persistent State Register 1 (HW_RTC_PERSISTENT1)

The 32-bit persistent registers are used to retain certain control states during chip wide power-down states.

HW_RTC_PERSISTENT1: 0x070

HW_RTC_PERSISTENT1_SET: 0x074

HW_RTC_PERSISTENT1_CLR: 0x078

HW_RTC_PERSISTENT1_TOG: 0x07C

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on but not in zero time.

EXAMPLE

```
            HW_RTC_PERSISTENT1_WR(0x12345678); // this write will ultimately push data to the
 analog side via the copy controller
```

Address: HW_RTC_PERSISTENT1 – 8005_6000h base + 70h offset = 8005_6070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GENERAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT1 field descriptions

Field	Description
31–0 GENERAL	Firmware use, defined as follows: 0x1000 **ENUMERATE_500MA_TWICE** — Enumerate at 500mA twice before dropping back to 100mA. 0x0800 **USB_BOOT_PLAYER_MODE** — Boot to player when connected to USB. 0x0400 **SKIP_CHECKDISK** — Run Checkdisk flag. 0x0200 **USB_LOW_POWER_MODE** — USB Hi/Lo Current select. 0x0100 **OTG_HNP_BIT** — HNP has been required if set to one. 0x0080 **OTG_ATL_ROLE_BIT** — USB role.

22.8.9 Persistent State Register 2 (HW_RTC_PERSISTENT2)

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

HW_RTC_PERSISTENT2: 0x080

HW_RTC_PERSISTENT2_SET: 0x084

HW_RTC_PERSISTENT2_CLR: 0x088

HW_RTC_PERSISTENT2_TOG: 0x08C

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on but not in zero time.

EXAMPLE

```
            HW_RTC_PERSISTENT2_WR(0x12345678); // this write will ultimately push data to the
```

```
 analog side via the copy controller
```

Address: HW_RTC_PERSISTENT2 – 8005_6000h base + 80h offset = 8005_6080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	GENERAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT2 field descriptions

Field	Description
31–0 GENERAL	FIRMWARE/SOFTWARE

22.8.10 Persistent State Register 3 (HW_RTC_PERSISTENT3)

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

HW_RTC_PERSISTENT3: 0x090

HW_RTC_PERSISTENT3_SET: 0x094

HW_RTC_PERSISTENT3_CLR: 0x098

HW_RTC_PERSISTENT3_TOG: 0x09C

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on but not in zero time.

EXAMPLE

```
             HW_RTC_PERSISTENT3_WR(0x12345678); // this write will ultimately push data to the
 analog side via the copy controller
```

Address: HW_RTC_PERSISTENT3 – 8005_6000h base + 90h offset = 8005_6090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	GENERAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT3 field descriptions

Field	Description
31–0 GENERAL	FIRMWARE/SOFTWARE

22.8.11 Persistent State Register 4 (HW_RTC_PERSISTENT4)

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

HW_RTC_PERSISTENT4: 0x0A0

HW_RTC_PERSISTENT4_SET: 0x0A4

HW_RTC_PERSISTENT4_CLR: 0x0A8

HW_RTC_PERSISTENT4_TOG: 0x0AC

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on but not in zero time.

EXAMPLE

```
            HW_RTC_PERSISTENT4_WR(0x12345678); // this write will ultimately push data to the
 analog side via the copy controller
```

Address: HW_RTC_PERSISTENT4 – 8005_6000h base + A0h offset = 8005_60A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GENERAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT4 field descriptions

Field	Description
31–0 GENERAL	FIRMWARE/SOFTWARE

22.8.12 Persistent State Register 5 (HW_RTC_PERSISTENT5)

The 32-bit persistent registers are used to retain certain control states during chip-wide power-down states.

HW_RTC_PERSISTENT5: 0x0B0

HW_RTC_PERSISTENT5_SET: 0x0B4

HW_RTC_PERSISTENT5_CLR: 0x0B8

HW_RTC_PERSISTENT5_TOG: 0x0BC

The register initalizes to a known reset pattern. The copy controller overwrites this digital reset value very soon after power on but not in zero time.

EXAMPLE

```
            HW_RTC_PERSISTENT5_WR(0x12345678); // this write will ultimately push data to the
 analog side via the copy controller
```

Address: HW_RTC_PERSISTENT5 – 8005_6000h base + B0h offset = 8005_60B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GENERAL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_PERSISTENT5 field descriptions

Field	Description
31–0 GENERAL	FIRMWARE/SOFTWARE

22.8.13 Real-Time Clock Debug Register (HW_RTC_DEBUG)

This 32-bit register provides debug read access to various internal states for diagnostic purposes.

HW_RTC_DEBUG: 0x0C0

HW_RTC_DEBUG_SET: 0x0C4

HW_RTC_DEBUG_CLR: 0x0C8

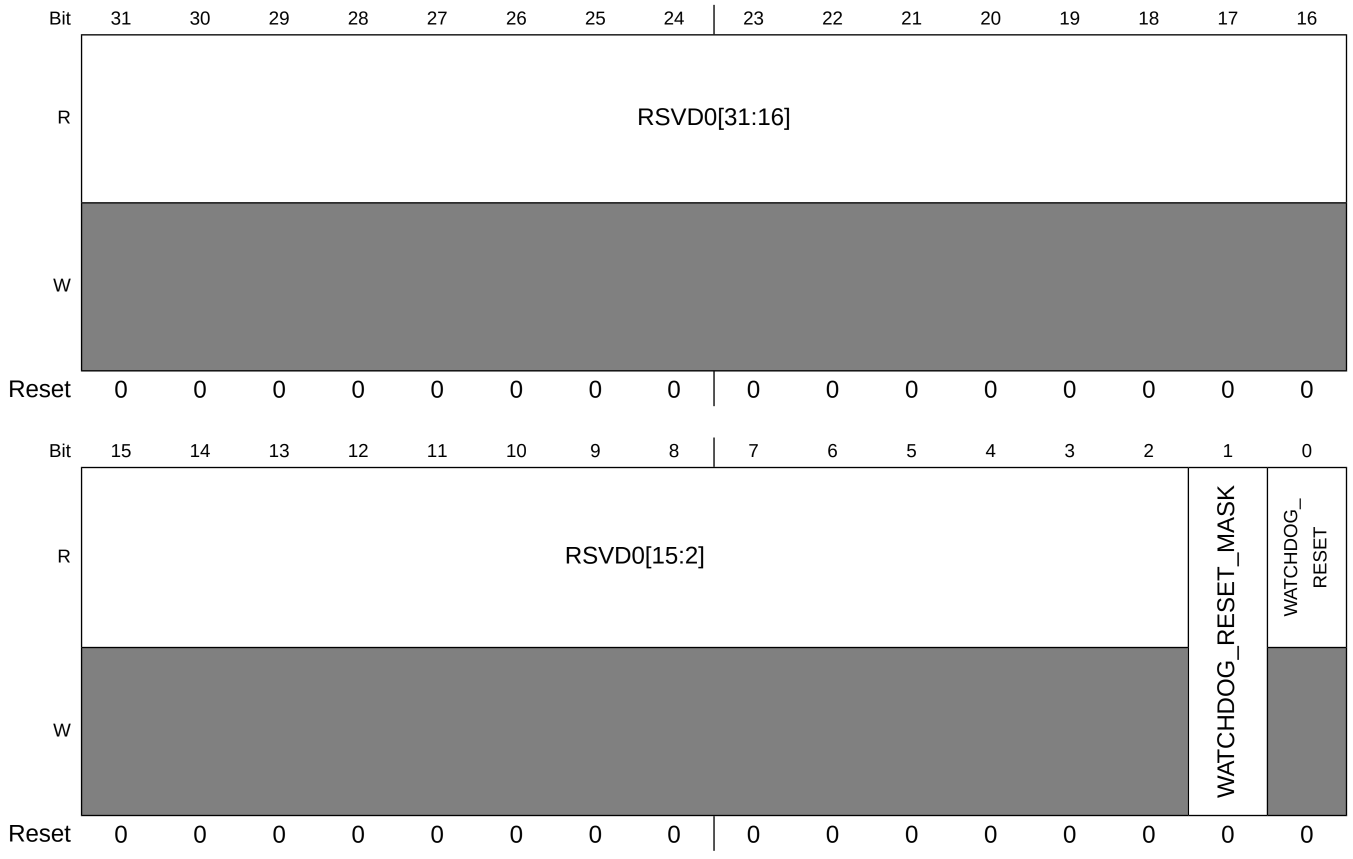
HW_RTC_DEBUG_TOG: 0x0CC

Read-only view into the internals of the digital side of the RTC for diagnostic purposes.

EXAMPLE

```
        DebugValue = HW_RTC_DEBUG_RD();  // read debug register value
```

Address: HW_RTC_DEBUG – 8005_6000h base + C0h offset = 8005_60C0h



HW_RTC_DEBUG field descriptions

Field	Description
31–2 RSVD0	Debug read-only view of various state machine bits.
1 WATCHDOG_ RESET_MASK	When set, mask the reset generation by the watchdog timer for testing purposes.
0 WATCHDOG_ RESET	Reflects the state of the watchdog reset. Used for testing purposes so the watchdog can be tested without resetting part. When set, Watchdog reset is asserted.

22.8.14 Real-Time Clock Version Register (HW_RTC_VERSION)

Version register.

EXAMPLE

```
            VersionValue = HW_RTC_VERSION_RD();  // read debug register value
```

Address: HW_RTC_VERSION – 8005_6000h base + D0h offset = 8005_60D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Re-set	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_RTC_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 23
Timers and Rotary Decoder (TIMROT)

23.1 Timers and Rotary Decoder (TIMROT) Overview

The device implements four timers and a rotary decoder, as shown in Figure 23-1. The timers and decoder can take their inputs from any of the pins defined for PWM, rotary encoders, or certain divisions from the 32-KHz clock input. Therefore, the PWM pins can be inputs or outputs, depending on the application.

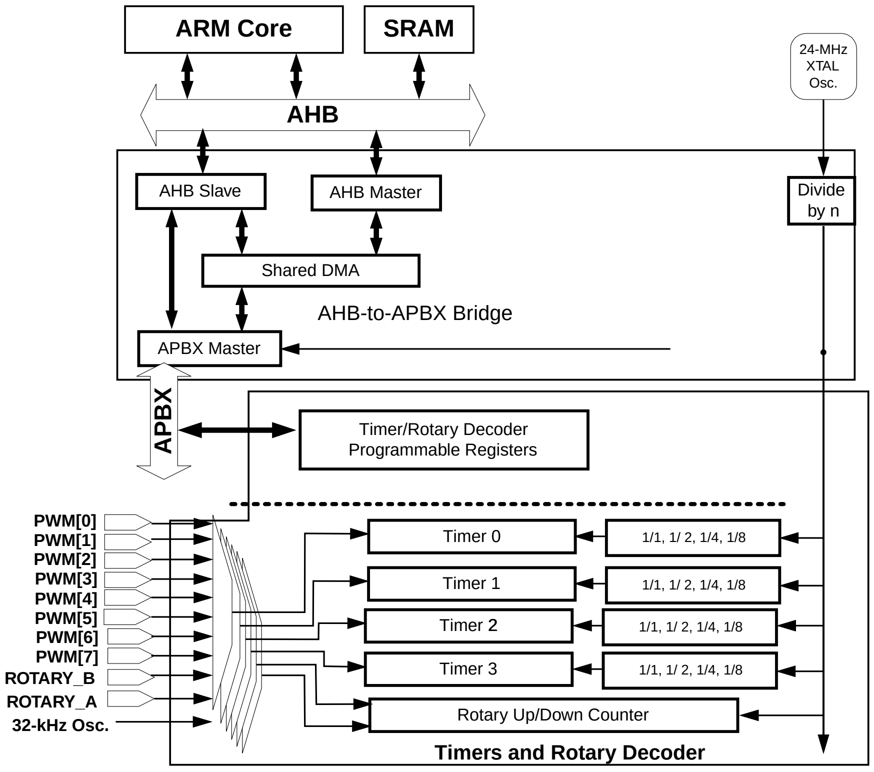


Figure 23-1. Timers and Rotary Decoder Block Diagram

The timer/rotary decoder block is a programmed I/O interface connected to the APBX bus. Recall that the APBX typically runs at a divided clock rate from the 24-MHz crystal clock. Each timer and rotary channel can sample at a rate that is further subdivided from the APBX clock. Each timer can select a different pre-scaler value.

23.2 Timer

There are two running mode for each timer. One is fixed count mode and the other is match count mode. By default Timer works in fixed count mode, and match count mode can be enabled by writing 1'b1 to bit **MATCH_MODE** of register **TIMERx Control and Status Register** (x = 0-3).

23.2.1 Fixed count mode

In this mode, the timer consists of a 32-bit fixed count value and a 32-bit free-running count value. The free running count decrements until it reaches 0 where it sets an interrupt status bit associated with the counter.

- If the RELOAD bit is set to 1, then the fixed count is automatically copied to the free-running counter and the count continues.
- If the RELOAD bit is not set, the timer stalls when it reaches 0.

Figure 23-2 shows a detailed view of each timer in fixed count mode.

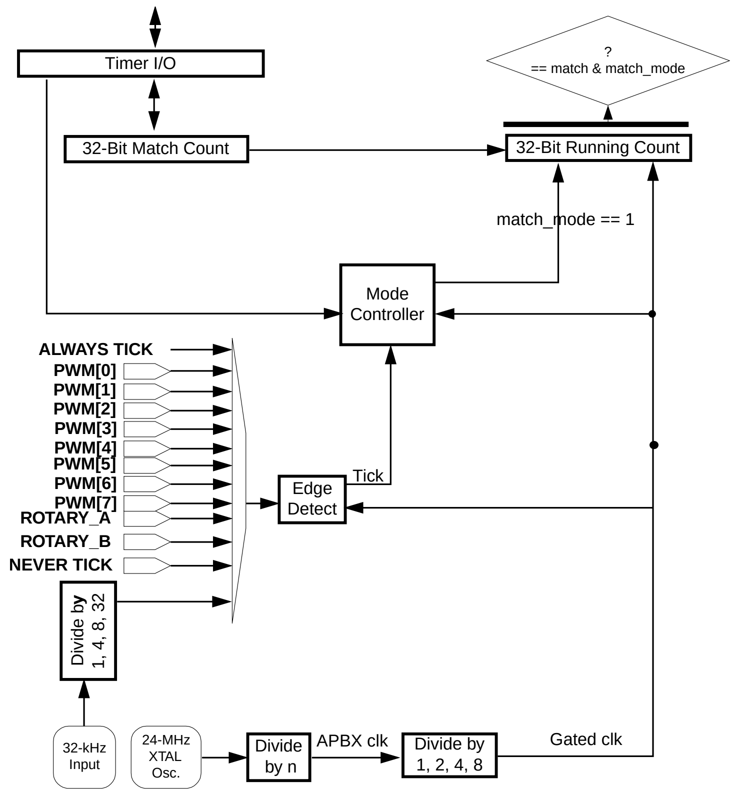


Figure 23-2. Timer 0, Timer 1, Timer 2 or Timer 3 Detail

Each timer has an UPDATE bit that controls whether the free-running counter is loaded at the same time the fixed-count register is written from the CPU. The output of each timer's source select has a polarity control that allows the timer to operate on either edge.

Table 23-1 lists the timer state machine transitions.

Table 23-1. Timer State Machine Transitions

UPDATE	RELOAD	RUNNING
0	0	PIO writes to the fixed-count bit field have no effect on the running count.
0	1	The value written to the fixed count is used to reload the running count the next time it reaches 0. When the fixed count has been written with a value of 0 and the running count reaches 0, it continuously copies the fixed count value to the running count. Therefore, writing a non-zero value to the fixed count register kicks off a continuous count and update operation.
1	0	The value written to the fixed count bit field is copied, immediately to the running count, restarting any existing running count operation. When the new running count reaches 0, it freezes.
1	1	The value written to the fixed count bit field is copied, immediately to the running count, restarting any existing running count operation. When the new running count reaches 0, it is reloaded from the value in the fixed count bit field, therefore running continuously using the newly supplied fixed count.

When generating a periodic timer interrupt using the RELOAD bit, the user must compute proper fixed-count value (count_value) based on clock speeds and clock divider settings. Note that, in this case, the actual value written to the FIXED_COUNT register field should be count_value – 1. For one-shot interrupts (RELOAD bit not set), the value written should be count_value.

Any timer interrupt can be programmed as an FIQ or as a regular IRQ. See Interrupt Collector (ICOLL) Overview for programming details.

For proper detection of the input source signal, it should be much slower than the pre-scaled APBX clock (not greater than one-third the frequency of the pre-scaled APBX clock).

Selecting the ALWAYS tick causes the timer to decrement continuously at the rate established by the pre-scaled APBX clock. The NEVER TICK selection causes the timer to stall. Setting the fixed-count to 0xFFFFFFFF and setting the RELOAD bit causes the timer to operate continuously with a 4,294,967,296 count.

The state of the 32-bit free-running count can be read by the CPU for each timer.

23.2.2 Match count mode

In this mode, the timer consists of a 32-bit match count value and a 32-bit free-running count value. This mode is enabled once **MATCH MODE** of Register **TIMERx Control and Status Register** (x = 0-3) is written to 1. At the same time, the free-running count returns to its default start value 0xFFFF_FFFF, then starts decrementing from 0xFFFF_FFFF to 0. Every time it decreases to 0, it restores to 0xFFFF_FFFF and keeps decreasing repeatedly. During this ceaseless decrement, interrupt is triggered every time free running count equals match count.

Figure 23-3 shows a detailed view of each timer in match mode.

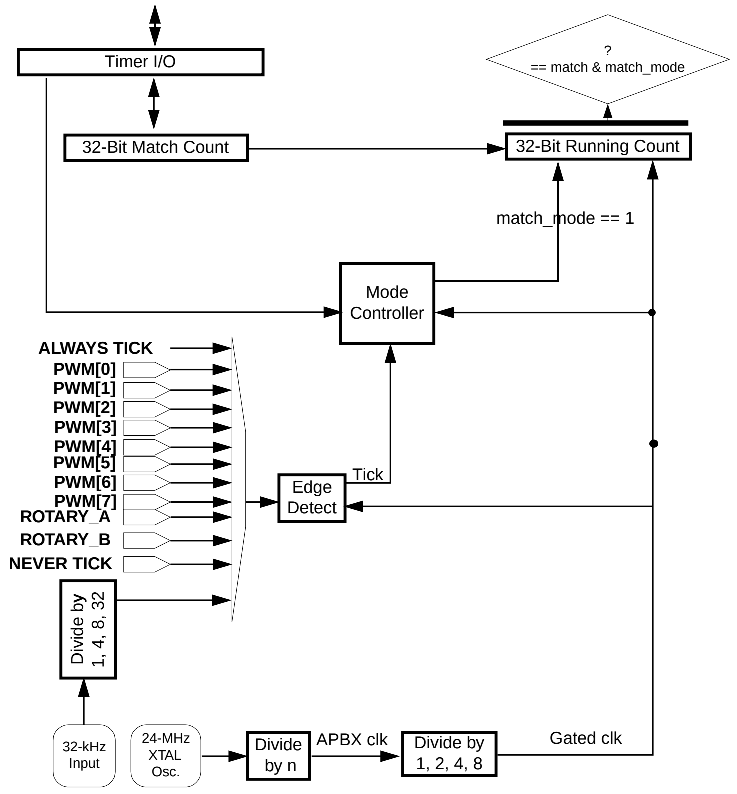


Figure 23-3. Timer 0, Timer 1, Timer 2 or Timer 3 Detail

In match mode, period timer interrupt can be triggered by modifying the match register to a proper value adequately smaller than the current value of the match register in every interrupt service routine, which is usually calculated by subtracting a delta from the current value of the match register. To get a group of interrupts with fixed interval, constant delta is used, and to get interrupts with variable interval, variable delta is used.

Any timer interrupt can be programmed as a FIQ or as a regular IRQ. See Interrupt Collector (ICOLL) Overview for programming details.

For proper detection of the input source signal, it should be much slower than the pre-scaled APBX clock (not greater than one-third the frequency of the pre-scaled APBX clock).

Selecting the ALWAYS tick causes the timer to decrement continuously at the rate established by the pre-scaled APBX clock. The NEVER TICK selection causes the timer to stall.

The state of the 32-bit free-running count can be read by the CPU for each timer.

23.2.3 Using External Signals as Inputs

External signals can be used as inputs to the block. They can be used as either the test signal or sampling input signals (duty cycle or normal timer mode). This can be accomplished by using the rotary input pins or any unused PWM pins. If PWM pins are being used for this purpose, conflicts with the PWM or other blocks that could drive the pins as outputs must be avoided. In this case, the PWM pins being used should be programmed as GPIO inputs. (See Pin Control and GPIO Overview for details.) Then, the external signal can be wired to the pin, and the PWM number selected in the appropriate TIMROT registers.

23.2.4 Timer 3 and Duty Cycle Mode

Timer 3 can operate in the same modes as Timer 0, Timer 1, and Timer 2. However, it has an additional duty cycle measurement mode. Figure 23-4 shows a detailed view of Timer 3 in duty cycle mode. This mode can be enabled by writing 1 to bit **DUTY_CYCLE** of Register **TIMER3 Control and Status Register**. When this bit is set, no matter what value **MATCH MODE** of Register **TIMER3 Control and Status Register** is, Timer 3 works in duty cycle mode.

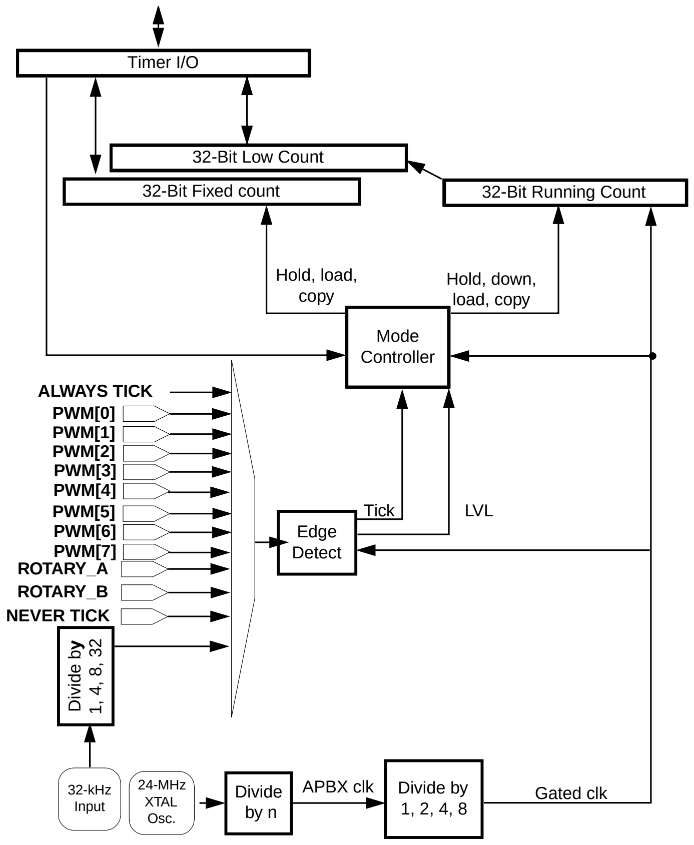


Figure 23-4. Timer 3 Detail

In the duty cycle mode, Timer 3 samples the free-running counter at the rising and falling edges of the input test signal, resetting the free-running counter on the same clock that is sampled.

- On the rising edge of the test signal, the free-running count is copied to the LOW_RUNNING_COUNT bit field of the HW_TIMROT_TIMCOUNT3 register.
- On the falling edge of the source clock, the free-running count is copied to the HIGH_FIXED_COUNT bit field (as shown in Figure 23-5).

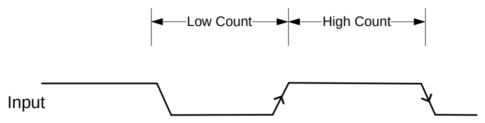


Figure 23-5. Pulse-Width Measurement Mode

- Once duty cycle mode is programmed and the input signal is stable, software should poll the DUTY_VALID bit in the HW_TIMROT_TIMCTRL3 register.
- This bit is automatically set and cleared by the hardware. When this bit is set, count values in the HW_TIMROT_TIMCOUNT3 register are stable and ready to be read.

Refer to the Timer 3 control and status register, HW_TIMROT_TIMCTRL3, where the DUTY_CYCLE bit controls whether HW_TIMROT_TIMCOUNT3 register's LOW_RUNNING_COUNT bit field reads back the running count or the low count of a duty cycle measurement. The DUTY_CYCLE bit also controls whether the HIGH_FIXED_COUNT bit field reads back the fixed-count value used in normal timer operations or the duty cycle high-time measurement.

It should be noted that for duty cycle mode to function properly, the timer tick source selected (SELECT field of the HW_TIMROT_TIMCTRL3 register) should be an appropriate frequency to sample the test signal. The NEVER_TICK value should never be used in this mode, as it yields incorrect count results.

23.3 Rotary Decoder

The rotary decoder uses two input selectors and edge detectors, as shown in Figure 23-6. It includes a debounce circuit for each input, as shown in Figure 23-7. This figure shows the debounce circuit for input A, though the circuit is identical for input B.

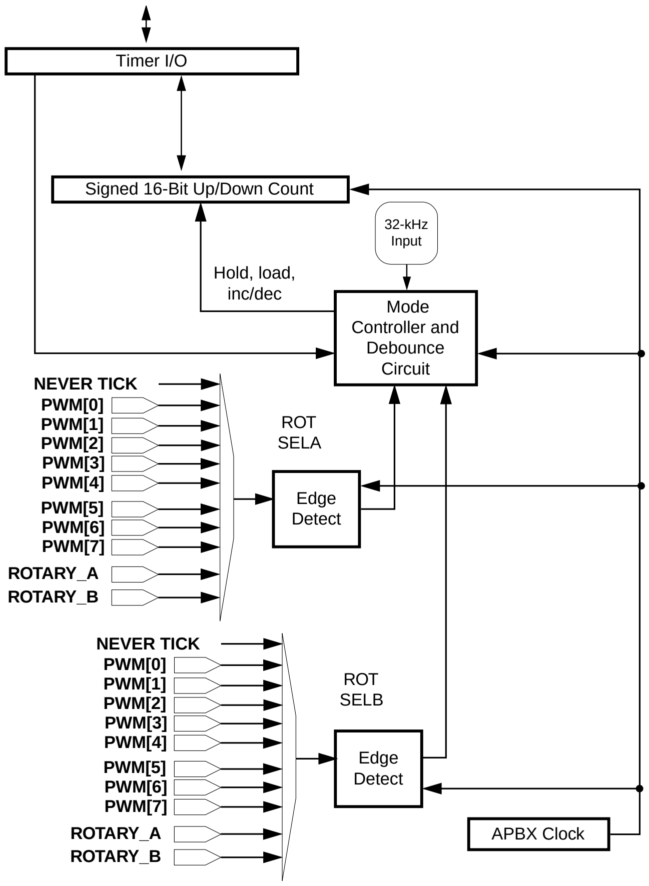


Figure 23-6. Detail of Rotary Decoder

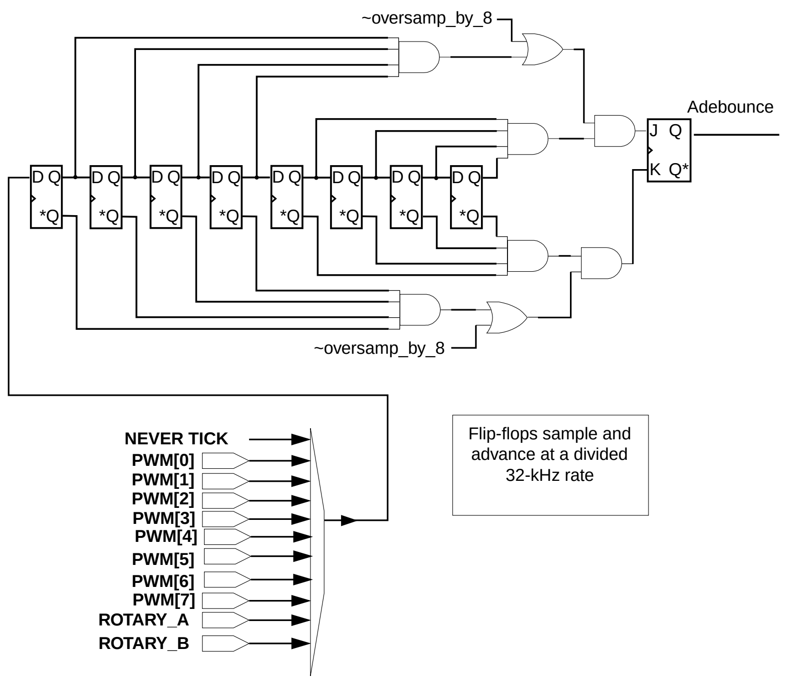


Figure 23-7. Rotary Decoding Mode-Debouncing Rotary A and B Inputs

A state machine following rotary decoder transition is provided to detect the direction of rotation and the time at which to increment or decrement the 16-bit signed counter in HW_TIMROT_ROTCOUNT. The updown counter can be treated as either a relative count or an absolute count, depending on the state of the HW_TIMROT_ROTCTRL_RELATIVE bit. When set to the relative mode, each read of the counter has the side effect of resetting it. The edge detectors respond to both edges of each input to determine the self-timed transition inputs to the state machine (see Figure 23-8).

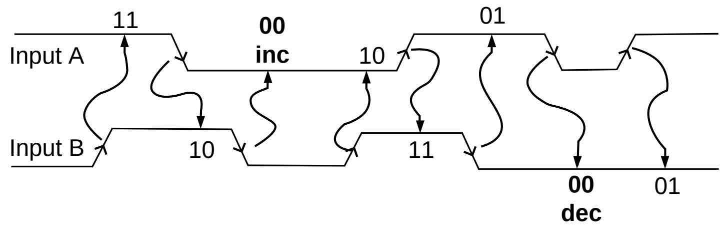


Figure 23-8. Rotary Decoding Mode-Input Transitions

Figure 23-8 shows that each detected edge causes a transition in the decoder state machine. Not all transitions are legal (see Table 23-2). For example, there is no legal way to transition directly from state 11 to 00 using normal inputs. In the cases where this occurs, the state machine goes to an alternate set of states and follows the input sequence until a valid sequence leading to state 00 is detected. No increment or decrement action is taken from the alternate state sequence.

Table 23-2. Rotary Decoder State Machine Transitions

CURRENT STATE	INPUT BA=00	INPUT BA=01	INPUT BA=10	INPUT BA=11
00	00	01	10	error
01	00, dec	01	error	11
10	00, inc	error	10	11
11	error	01	10	11

23.3.1 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

23.4 Programmable Registers

i.MX28 TIMROT Hardware Register Format Summary

HW_TIMROT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8006_8000	Rotary Decoder Control Register (HW_TIMROT_ROTCTRL)	32	R/W	FE00_0000h	23.4.1/1510
8006_8010	Rotary Decoder Up/Down Counter Register (HW_TIMROT_ROTCOUNT)	32	R	0000_0000h	23.4.2/1513
8006_8020	Timer 0 Control and Status Register (HW_TIMROT_TIMCTRL0)	32	R/W	0000_0000h	23.4.3/1513
8006_8030	Timer 0 Runing Count Register (HW_TIMROT_RUNNING_COUNT0)	32	R	0000_0000h	23.4.4/1515
8006_8040	Timer 0 Fixed Count Register (HW_TIMROT_FIXED_COUNT0)	32	R/W	0000_0000h	23.4.5/1516
8006_8050	Timer 0 Match Count Register (HW_TIMROT_MATCH_COUNT0)	32	R/W	0000_0000h	23.4.6/1516
8006_8060	Timer 1 Control and Status Register (HW_TIMROT_TIMCTRL1)	32	R/W	0000_0000h	23.4.7/1517
8006_8070	Timer 1 Runing Count Register (HW_TIMROT_RUNNING_COUNT1)	32	R	0000_0000h	23.4.8/1519
8006_8080	Timer 1 Fixed Count Register (HW_TIMROT_FIXED_COUNT1)	32	R/W	0000_0000h	23.4.9/1519

HW_TIMROT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8006_8090	Timer 1 Match Count Register (HW_TIMROT_MATCH_COUNT1)	32	R/W	0000_0000h	23.4.10/1520
8006_80A0	Timer 2 Control and Status Register (HW_TIMROT_TIMCTRL2)	32	R/W	0000_0000h	23.4.11/1520
8006_80B0	Timer 2 Runing Count Register (HW_TIMROT_RUNNING_COUNT2)	32	R	0000_0000h	23.4.12/1522
8006_80C0	Timer 2 Fixed Count Register (HW_TIMROT_FIXED_COUNT2)	32	R/W	0000_0000h	23.4.13/1523
8006_80D0	Timer 2 Match Count Register (HW_TIMROT_MATCH_COUNT2)	32	R/W	0000_0000h	23.4.14/1523
8006_80E0	Timer 3 Control and Status Register (HW_TIMROT_TIMCTRL3)	32	R/W	0000_0000h	23.4.15/1524
8006_80F0	Timer 3 Running Count Register (HW_TIMROT_RUNNING_COUNT3)	32	R	0000_0000h	23.4.16/1526
8006_8100	Timer 3 Count Register (HW_TIMROT_FIXED_COUNT3)	32	R/W	0000_0000h	23.4.17/1527
8006_8110	Timer 3 Match Count Register (HW_TIMROT_MATCH_COUNT3)	32	R/W	0000_0000h	23.4.18/1527
8006_8120	TIMROT Version Register (HW_TIMROT_VERSION)	32	R	0200_0000h	23.4.19/1528

23.4.1 Rotary Decoder Control Register (HW_TIMROT_ROTCTRL)

The Rotary Decoder Control Register specifies the reset state and the source selection for the rotary decoder. In addition, it specifies the polarity of any external input source that is used. This register also contains some general block controls including soft reset, clock gate, and present bits.

HW_TIMROT_ROTCTRL: 0x000

HW_TIMROT_ROTCTRL_SET: 0x004

HW_TIMROT_ROTCTRL_CLR: 0x008

HW_TIMROT_ROTCTRL_TOG: 0x00C

EXAMPLE

```
HW_TIMROT_ROTCTRL_WR(0x00000076);   // Set up rotary control fields.
```

Address: HW_TIMROT_ROTCTRL – 8006_8000h base + 0h offset = 8006_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	ROTARY_ PRESENT	TIM3_ PRESENT	TIM2_ PRESENT	TIM1_ PRESENT	TIM0_ PRESENT	STATE			DIVIDER					
W			▨	▨	▨	▨	▨	▨	▨	▨						
Reset	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3			RELATIVE	OVERSAMPLE		POLARITY_B	POLARITY_A	SELECT_B				SELECT_A			
W	▨	▨	▨													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_ROTCTRL field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable operation of any timer or the rotary decoder. When set to one, it forces a block-level reset and gates off the clocks to the block.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one, it gates off the clocks to the block.
29 ROTARY_ PRESENT	0= Rotary decoder is not present in this product. 1= Rotary decoder is present is in this product.
28 TIM3_PRESENT	0= TIMER3 is not present in this product. 1= TIMER3 is present is in this product.
27 TIM2_PRESENT	0= TIMER2 is not present in this product. 1= TIMER2 is present is in this product.
26 TIM1_PRESENT	0= TIMER1 is not present in this product. 1= TIMER1 is present is in this product.
25 TIM0_PRESENT	0= TIMER0 is not present in this product. 1= TIMER0 is present is in this product.
24–22 STATE	Read-only view of the rotary decoder transition detecting state machine.
21–16 DIVIDER	This bit field determines the divisor used to divide the 32-kHz on chip clock rate for oversampling (debouncing) the rotary A and B inputs. Note that the divider value is actually the (value of this field+1).

HW_TIMROT_ROTCTRL field descriptions (continued)

Field	Description
15–13 RSRVD3	Always write zeroes to this bit field.
12 RELATIVE	Set this bit to one to cause the rotary decoders updown counter to be reset to zero whenever it is read.
11–10 OVERSAMPLE	This bit field determines the oversample rate to use in debouncing Rotary A and B inputs. 0x0 **8X** — 8x Oversample: 8 successive ones or zeroes to transition. 0x1 **4X** — 4x Oversample: 4 successive ones or zeroes to transition. 0x2 **2X** — 2x Oversample: 2 successive ones or zeroes to transition. 0x3 **1X** — 1x Oversample: Transition on each first input change.
9 POLARITY_B	Set this bit to one to invert the input to the edge detector.
8 POLARITY_A	Set this bit to one to invert the input to the edge detector.
7–4 SELECT_B	Selects the source for the timer tick that increments the free-running counter that measures the A2B and B2A overlap counts. 0x0 **NEVER_TICK** — SelectB: Never tick. 0x1 **PWM0** — SelectB: Input from PWM0. 0x2 **PWM1** — SelectB: Input from PWM1. 0x3 **PWM2** — SelectB: Input from PWM2. 0x4 **PWM3** — SelectB: Input from PWM3. 0x5 **PWM4** — SelectB: Input from PWM4. 0x6 **PWM5** — SelectB: Input from PWM5. 0x7 **PWM6** — SelectB: Input from PWM6. 0x8 **PWM7** — SelectB: Input from PWM7. 0x9 **ROTARYA** — SelectB: Input from Rotary A. 0xA **ROTARYB** — SelectB: Input from Rotary B.
3–0 SELECT_A	Selects the source for the timer tick that increments the free-running counter that measures the A2B and B2A overlap counts. 0x0 **NEVER_TICK** — SelectA: Never tick. 0x1 **PWM0** — SelectA: Input from PWM0. 0x2 **PWM1** — SelectA: Input from PWM1. 0x3 **PWM2** — SelectA: Input from PWM2. 0x4 **PWM3** — SelectA: Input from PWM3. 0x5 **PWM4** — SelectA: Input from PWM4. 0x6 **PWM5** — SelectB: Input from PWM5. 0x7 **PWM6** — SelectB: Input from PWM6. 0x8 **PWM7** — SelectB: Input from PWM7. 0x9 **ROTARYA** — SelectB: Input from Rotary A. 0xA **ROTARYB** — SelectB: Input from Rotary B.

23.4.2 Rotary Decoder Up/Down Counter Register (HW_TIMROT_ROTCOUNT)

The Rotary Decoder Up/Down Counter Register contains the timer counter value that counts up or down as the rotary encoder is rotated.

This register contains the read-only current count for the rotary decoder.

EXAMPLE

```
count = HW_TIMROT_ROTCTRL_RD(); // Read count
```

Address: HW_TIMROT_ROTCOUNT – 8006_8000h base + 10h offset = 8006_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1																UPDOWN															
W	▨																▨															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_ROTCOUNT field descriptions

Field	Description
31–16 RSRVD1	Always write zeroes to this bit field.
15–0 UPDOWN	At each edge of the Rotary A input, the Rotary B value is sampled, similarly at each edge of the Rotary B input, the Rotary A input is sampled. These values drive a rotary decoder state machine that determines when this counter is incremented or decremetned. When set in the RELATIVE mode, reads from this register clear this register as a side effect. Counter values in this register are signed 16-bit values.

23.4.3 Timer 0 Control and Status Register (HW_TIMROT_TIMCTRL0)

The Timer 0 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 0.

HW_TIMROT_TIMCTRL0: 0x020

HW_TIMROT_TIMCTRL0_SET: 0x024

HW_TIMROT_TIMCTRL0_CLR: 0x028

HW_TIMROT_TIMCTRL0_TOG: 0x02C

EXAMPLE

```
HW_TIMROT_TIMCTRLn_WR(0, 0x00000008);   // Set up control fields for timer0
```

Address: HW_TIMROT_TIMCTRL0 – 8006_8000h base + 20h offset = 8006_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQ	IRQ_ EN	RSRVD2		MATCH_ MODE	RSRVD1		POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			
W			▨			▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_TIMCTRL0 field descriptions

Field	Description
31–16 RSRVD3	Always write zeroes to this bit field.
15 IRQ	This bit is set to one when Timer 0 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14 IRQ_EN	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13–12 RSRVD2	Always write zeroes to this bit field.
11 MATCH_MODE	Set this bit to one to enable timer match mode
10–9 RSRVD1	Always write zeroes to this bit field.
8 POLARITY	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Negative edge detection.
7 UPDATE	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6 RELOAD	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5–4 PRESCALE	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. 0x0 **DIV_BY_1** — PreScale: Divide the APBX clock by 1. 0x1 **DIV_BY_2** — PreScale: Divide the APBX clock by 2.

HW_TIMROT_TIMCTRL0 field descriptions (continued)

Field	Description
	0x2 **DIV_BY_4** — PreScale: Divide the APBX clock by 4. 0x3 **DIV_BY_8** — PreScale: Divide the APBX clock by 8.
3–0 SELECT	Selects the source for the timer tick that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. 0x0 **NEVER_TICK** — Never tick. 0x1 **PWM0** — Input from PWM0. 0x2 **PWM1** — Input from PWM1. 0x3 **PWM2** — Input from PWM2. 0x4 **PWM3** — Input from PWM3. 0x5 **PWM4** — Input from PWM4. 0x6 **PWM5** — Input from PWM5. 0x7 **PWM6** — Input from PWM6. 0x8 **PWM7** — Input from PWM7. 0x9 **ROTARYA** — Input from Rotary A. 0xA **ROTARYB** — Input from Rotary B. 0xB **32KHZ_XTAL** — Input from 32-kHz crystal. 0xC **8KHZ_XTAL** — Input from 8-kHz (divided from 32-kHz crystal). 0xD **4KHZ_XTAL** — Input from 4-kHz (divided from 32-kHz crystal). 0xE **1KHZ_XTAL** — Input from 1-kHz (divided from 32-kHz crystal). 0xF **TICK_ALWAYS** — Always tick.

23.4.4 Timer 0 Runing Count Register (HW_TIMROT_RUNNING_COUNT0)

The Timer 0 Running Count Register contains the timer running counter values for Timer 0.

Address: HW_TIMROT_RUNNING_COUNT0 – 8006_8000h base + 30h offset = 8006_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RUNNING_COUNT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_RUNNING_COUNT0 field descriptions

Field	Description
31–0 RUNNING_ COUNT	This bit field shows the current state of the running count as it decrements.

23.4.5 Timer 0 Fixed Count Register (HW_TIMROT_FIXED_COUNT0)

The Timer 0 Fixed Count Register contains the fixed timer counter values for Timer 0.

EXAMPLE

```
HW_TIMROT_FIXED_COUNTn_WR(0, 0x0000f0dd);  // Set up timer fixed count
```

Address: HW_TIMROT_FIXED_COUNT0 – 8006_8000h base + 40h offset = 8006_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIXED_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_FIXED_COUNT0 field descriptions

Field	Description
31–0 FIXED_COUNT	Software loads the fixed count bit field with the value to down count. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

23.4.6 Timer 0 Match Count Register (HW_TIMROT_MATCH_COUNT0)

The Timer 0 Match Count Register contains the match timer counter values for Timer 0.

EXAMPLE

```
HW_TIMROT_MATCH_COUNTn_WR(0, 0x0000f0dd);  // Set up timer match count
```

Address: HW_TIMROT_MATCH_COUNT0 – 8006_8000h base + 50h offset = 8006_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_MATCH_COUNT0 field descriptions

Field	Description
31–0 MATCH_COUNT	Software compares the match count bit field with the value to down count. If they equal to each other, interrupt is trigger.

23.4.7 Timer 1 Control and Status Register (HW_TIMROT_TIMCTRL1)

The Timer 1 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 1.

HW_TIMROT_TIMCTRL1: 0x060

HW_TIMROT_TIMCTRL1_SET: 0x064

HW_TIMROT_TIMCTRL1_CLR: 0x068

HW_TIMROT_TIMCTRL1_TOG: 0x06C

EXAMPLE

```
HW_TIMROT_TIMCTRLn_WR(1, 0x00000008);   // Set up control fields for timer1
```

Address: HW_TIMROT_TIMCTRL1 – 8006_8000h base + 60h offset = 8006_8060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQ	IRQ_EN	RSRVD2		MATCH_MODE	RSRVD1		POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			
W			▨			▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_TIMCTRL1 field descriptions

Field	Description
31–16 RSRVD3	Always write zeroes to this bit field.
15 IRQ	This bit is set to one when Timer 1 decrements to zero. Write a zero to clear it or use Clear SCT mode.

HW_TIMROT_TIMCTRL1 field descriptions (continued)

Field	Description
14 IRQ_EN	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13–12 RSRVD2	Always write zeroes to this bit field.
11 MATCH_MODE	Set this bit to one to enable timer match mode
10–9 RSRVD1	Always write zeroes to this bit field.
8 POLARITY	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7 UPDATE	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6 RELOAD	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5–4 PRESCALE	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. 0x0 **DIV_BY_1** — PreScale: Divide the APBX clock by 1. 0x1 **DIV_BY_2** — PreScale: Divide the APBX clock by 2. 0x2 **DIV_BY_4** — PreScale: Divide the APBX clock by 4. 0x3 **DIV_BY_8** — PreScale: Divide the APBX clock by 8.
3–0 SELECT	Selects the source for the timer tick that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. 0x0 **NEVER_TICK** — Never tick. 0x1 **PWM0** — Input from PWM0. 0x2 **PWM1** — Input from PWM1. 0x3 **PWM2** — Input from PWM2. 0x4 **PWM3** — Input from PWM3. 0x5 **PWM4** — Input from PWM4. 0x6 **PWM5** — Input from PWM5. 0x7 **PWM6** — Input from PWM6. 0x8 **PWM7** — Input from PWM7. 0x9 **ROTARYA** — Input from Rotary A. 0xA **ROTARYB** — Input from Rotary B. 0xB **32KHZ_XTAL** — Input from 32-kHz crystal. 0xC **8KHZ_XTAL** — Input from 8-kHz (divided from 32-kHz crystal). 0xD **4KHZ_XTAL** — Input from 4-kHz (divided from 32-kHz crystal). 0xE **1KHZ_XTAL** — Input from 1-kHz (divided from 32-kHz crystal). 0xF **TICK_ALWAYS** — Always tick.

23.4.8 Timer 1 Runing Count Register (HW_TIMROT_RUNNING_COUNT1)

The Timer 1 Running Count Register contains the timer running counter values for Timer 1.

Address: HW_TIMROT_RUNNING_COUNT1 – 8006_8000h base + 70h offset = 8006_8070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RUNNING_COUNT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_RUNNING_COUNT1 field descriptions

Field	Description
31–0 RUNNING_ COUNT	This bit field shows the current state of the running count as it decrements.

23.4.9 Timer 1 Fixed Count Register (HW_TIMROT_FIXED_COUNT1)

The Timer 1 Fixed Count Register contains the fixed timer counter values for Timer 1.

EXAMPLE

```
HW_TIMROT_FIXED_COUNTn_WR(1, 0x0000f0dd);  // Set up timer fixed count
```

Address: HW_TIMROT_FIXED_COUNT1 – 8006_8000h base + 80h offset = 8006_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIXED_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_FIXED_COUNT1 field descriptions

Field	Description
31–0 FIXED_COUNT	Software loads the fixed count bit field with the value to down count. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

23.4.10 Timer 1 Match Count Register (HW_TIMROT_MATCH_COUNT1)

The Timer 1 Match Count Register contains the match timer counter values for Timer 1.

EXAMPLE

```
HW_TIMROT_MATCH_COUNTn_WR(1, 0x0000f0dd);  // Set up timer match count
```

Address: HW_TIMROT_MATCH_COUNT1 – 8006_8000h base + 90h offset = 8006_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_MATCH_COUNT1 field descriptions

Field	Description
31–0 MATCH_COUNT	Software compares the match count bit field with the value to down count. If they equal to each other, interrupt is trigger.

23.4.11 Timer 2 Control and Status Register (HW_TIMROT_TIMCTRL2)

The Timer 2 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 2.

HW_TIMROT_TIMCTRL2: 0x0a0

HW_TIMROT_TIMCTRL2_SET: 0x0a4

HW_TIMROT_TIMCTRL2_CLR: 0x0a8

HW_TIMROT_TIMCTRL2_TOG: 0x0aC

EXAMPLE

```
HW_TIMROT_TIMCTRLn_WR(2, 0x00000008);   // Set up control fields for timer2
```

Address: HW_TIMROT_TIMCTRL2 – 8006_8000h base + A0h offset = 8006_80A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQ	IRQ_EN	RSRVD2		MATCH_MODE	RSRVD1		POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			
W			▨			▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_TIMCTRL2 field descriptions

Field	Description
31–16 RSRVD3	Always write zeroes to this bit field.
15 IRQ	This bit is set to one when Timer 2 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14 IRQ_EN	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13–12 RSRVD2	Always write zeroes to this bit field.
11 MATCH_MODE	Set this bit to one to enable timer match mode
10–9 RSRVD1	Always write zeroes to this bit field.
8 POLARITY	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7 UPDATE	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6 RELOAD	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5–4 PRESCALE	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. 0x0 **DIV_BY_1** — PreScale: Divide the APBX clock by 1. 0x1 **DIV_BY_2** — PreScale: Divide the APBX clock by 2. 0x2 **DIV_BY_4** — PreScale: Divide the APBX clock by 4. 0x3 **DIV_BY_8** — PreScale: Divide the APBX clock by 8.

HW_TIMROT_TIMCTRL2 field descriptions (continued)

Field	Description
3–0 SELECT	Selects the source for the timer tick that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. 0x0 **NEVER_TICK** — Never tick. 0x1 **PWM0** — Input from PWM0. 0x2 **PWM1** — Input from PWM1. 0x3 **PWM2** — Input from PWM2. 0x4 **PWM3** — Input from PWM3. 0x5 **PWM4** — Input from PWM4. 0x6 **PWM5** — Input from PWM5. 0x7 **PWM6** — Input from PWM6. 0x8 **PWM7** — Input from PWM7. 0x9 **ROTARYA** — Input from Rotary A. 0xA **ROTARYB** — Input from Rotary B. 0xB **32KHZ_XTAL** — Input from 32-kHz crystal. 0xC **8KHZ_XTAL** — Input from 8-kHz (divided from 32-kHz crystal). 0xD **4KHZ_XTAL** — Input from 4-kHz (divided from 32-kHz crystal). 0xE **1KHZ_XTAL** — Input from 1-kHz (divided from 32-kHz crystal). 0xF **TICK_ALWAYS** — Always tick.

23.4.12 Timer 2 Runing Count Register (HW_TIMROT_RUNNING_COUNT2)

The Timer 2 Running Count Register contains the timer running counter values for Timer 2.

Address: HW_TIMROT_RUNNING_COUNT2 – 8006_8000h base + B0h offset = 8006_80B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RUNNING_COUNT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_RUNNING_COUNT2 field descriptions

Field	Description
31–0 RUNNING_ COUNT	This bit field shows the current state of the running count as it decrements.

23.4.13 Timer 2 Fixed Count Register (HW_TIMROT_FIXED_COUNT2)

The Timer 2 Fixed Count Register contains the fixed timer counter values for Timer 2.

EXAMPLE

```
HW_TIMROT_FIXED_COUNTn_WR(2, 0x0000f0dd);  // Set up timer fixed count
```

Address: HW_TIMROT_FIXED_COUNT2 – 8006_8000h base + C0h offset = 8006_80C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FIXED_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_FIXED_COUNT2 field descriptions

Field	Description
31–0 FIXED_COUNT	Software loads the fixed count bit field with the value to down count. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count.

23.4.14 Timer 2 Match Count Register (HW_TIMROT_MATCH_COUNT2)

The Timer 2 Match Count Register contains the match timer counter values for Timer 2.

EXAMPLE

```
HW_TIMROT_MATCH_COUNTn_WR(2, 0x0000f0dd);  // Set up timer match count
```

Address: HW_TIMROT_MATCH_COUNT2 – 8006_8000h base + D0h offset = 8006_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_MATCH_COUNT2 field descriptions

Field	Description
31–0 MATCH_COUNT	Software compares the match count bit field with the value to down count. If they equal to each other, interrupt is trigger.

23.4.15 Timer 3 Control and Status Register (HW_TIMROT_TIMCTRL3)

The Timer 3 Control and Status Register specifies timer control parameters, as well as interrupt status and the enable for Timer 3.

HW_TIMROT_TIMCTRL3: 0x0e0

HW_TIMROT_TIMCTRL3_SET: 0x0e4

HW_TIMROT_TIMCTRL3_CLR: 0x0e8

HW_TIMROT_TIMCTRL3_TOG: 0x0eC

EXAMPLE

```
HW_TIMROT_TIMCTRLn_WR(3, 0x00000008);   // Set up control fields for timer3
```

Address: HW_TIMROT_TIMCTRL3 – 8006_8000h base + E0h offset = 8006_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2												TEST_SIGNAL			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IRQ	IRQ_EN	RSRVD1		MATCH_MODE	DUTY_VALID	DUTY_CYCLE	POLARITY	UPDATE	RELOAD	PRESCALE		SELECT			
W			▨			▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_TIMCTRL3 field descriptions

Field	Description
31–20 RSRVD2	Always write zeroes to this bit field.
19–16 TEST_SIGNAL	Selects the source of the signal to be measured in duty cycle mode. 0x0 **NEVER_TICK** — Never tick. Freeze the count. 0x1 **PWM0** — Input from PWM0.

HW_TIMROT_TIMCTRL3 field descriptions (continued)

Field	Description
	0x2 **PWM1** — Input from PWM1. 0x3 **PWM2** — Input from PWM2. 0x4 **PWM3** — Input from PWM3. 0x5 **PWM4** — Input from PWM4. 0x6 **PWM5** — Input from PWM5. 0x7 **PWM6** — Input from PWM6. 0x8 **PWM7** — Input from PWM7. 0x9 **ROTARYA** — Input from Rotary A. 0xA **ROTARYB** — Input from Rotary B. 0xB **32KHZ_XTAL** — Input from 32-kHz crystal. 0xC **8KHZ_XTAL** — Input from 8-kHz (divided from 32-kHz crystal). 0xD **4KHZ_XTAL** — Input from 4-kHz (divided from 32-kHz crystal). 0xE **1KHZ_XTAL** — Input from 1-kHz (divided from 32-kHz crystal). 0xF **TICK_ALWAYS** — Always tick.
15 IRQ	This bit is set to one when Timer 3 decrements to zero. Write a zero to clear it or use Clear SCT mode.
14 IRQ_EN	Set this bit to one to enable the generation of a CPU interrupt when the count reaches zero in normal counter mode.
13–12 RSRVD1	Always write zeroes to this bit field.
11 MATCH_MODE	Set this bit to one to enable timer match mode
10 DUTY_VALID	This bit is set and cleared by the hardware. It is set only when in duty cycle measuring mode and the HW_TIMROT_TIMCOUNT3 has valid duty cycle data to be read. This register will be cleared if not in duty cycle mode or on writes to this register. In the case that it is written while in duty cycle mode, this bit will clear but will again be set at the appropriate time for reading the count register.
9 DUTY_CYCLE	Set this bit to one to cause the timer to operate in duty cycle measuring mode.
8 POLARITY	Set this bit to one to invert the input to the edge detector. 0: Positive edge detection. 1: Invert to negative edge detection.
7 UPDATE	Set this bit to one to cause the running count to be written from the CPU at the same time a new fixed count register value is written.
6 RELOAD	Set this bit to one to cause the timer to reload its current count from its fixed count value whenever the current count decrements to zero. When set to zero, the timer enters a mode that freezes at a count of zero. When the fixed count is zero, setting this bit to one causes a continuous reload of the fixed count register so that writting a non-zero value will start the timer.
5–4 PRESCALE	Selects the divisor used for clock generation. The APBX clock is divided by the following amount. Note the APBX clock itself is initially divided down from the 24.0-MHz crystal clock frequency. 0x0 **DIV_BY_1** — PreScale: Divide the APBX clock by 1. 0x1 **DIV_BY_2** — PreScale: Divide the APBX clock by 2. 0x2 **DIV_BY_4** — PreScale: Divide the APBX clock by 4.

HW_TIMROT_TIMCTRL3 field descriptions (continued)

Field	Description
	0x3 **DIV_BY_8** — PreScale: Divide the APBX clock by 8.
3–0 SELECT	Selects the source for the timer tick that decrements the free running counter. Note: programming an undefined value will result in "always tick" behavior. In duty cycle mode it increments the counter used to calculate the high and low cycle counts. 0x0 **NEVER_TICK** — Never tick. Freeze the count. 0x1 **PWM0** — Input from PWM0. 0x2 **PWM1** — Input from PWM1. 0x3 **PWM2** — Input from PWM2. 0x4 **PWM3** — Input from PWM3. 0x5 **PWM4** — Input from PWM4. 0x6 **PWM5** — Input from PWM5. 0x7 **PWM6** — Input from PWM6. 0x8 **PWM7** — Input from PWM7. 0x9 **ROTARYA** — Input from Rotary A. 0xA **ROTARYB** — Input from Rotary B. 0xB **32KHZ_XTAL** — Input from 32-kHz crystal. 0xC **8KHZ_XTAL** — Input from 8-kHz (divided from 32-kHz crystal). 0xD **4KHZ_XTAL** — Input from 4-kHz (divided from 32-kHz crystal). 0xE **1KHZ_XTAL** — Input from 1-kHz (divided from 32-kHz crystal). 0xF **TICK_ALWAYS** — Always tick.

23.4.16 Timer 3 Running Count Register (HW_TIMROT_RUNNING_COUNT3)

The Timer 3 Running Count Register contains the timer runningcounter values for Timer 3. NOTE: This timer can be put in a special duty cycle mode that will measure the duty cycle of an input test signal.

The definitions of the fields change depending whether the timer is in normal or duty cycle mode.

Address: HW_TIMROT_RUNNING_COUNT3 – 8006_8000h base + F0h offset = 8006_80F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LOW_RUNNING_COUNT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_RUNNING_COUNT3 field descriptions

Field	Description
31–0 LOW_RUNNING_ COUNT	In duty cycle mode, this bit field is loaded from the running counter when it has just finished measuring the low portion of the duty cycle. In normal timer mode, it shows the running count as a read-only value.

23.4.17 Timer 3 Count Register (HW_TIMROT_FIXED_COUNT3)

The Timer 3 fixed Count Register contains the timer fixed counter values for Timer 3. NOTE: This timer can be put in a special duty cycle mode that will measure the duty cycle of an input test signal.

The definitions of the fields change depending whether the timer is in normal or duty cycle mode.

EXAMPLE

```
HW_TIMROT_FIXED_COUNTn_WR(3, 0x0000f0dd);  // Set up timer fixed count
```

Address: HW_TIMROT_FIXED_COUNT3 – 8006_8000h base + 100h offset = 8006_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIGH_FIXED_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_FIXED_COUNT3 field descriptions

Field	Description
31–0 HIGH_FIXED_ COUNT	Software loads the fixed count bit field with the value to down count. If the reload bit is set to one, then the new value will be loaded into the running count the next time it reaches zero. If the update bit is set to one, then the new value is also copied into the running count, immediately. If both the reload and update bits are set to zero, then the new value is never picked up by the running count. In duty cycle mode, this bit field is loaded from the running counter when it has finished measuring the high portion of the duty cycle.

23.4.18 Timer 3 Match Count Register (HW_TIMROT_MATCH_COUNT3)

The Timer 3 Match Count Register contains the match timer counter values for Timer 3.

EXAMPLE

```
HW_TIMROT_MATCH_COUNTn_WR(0, 0x0000f0dd);  // Set up timer match count
```

Address: HW_TIMROT_MATCH_COUNT3 – 8006_8000h base + 110h offset = 8006_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH_COUNT																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_MATCH_COUNT3 field descriptions

Field	Description
31–0 MATCH_COUNT	Software compares the match count bit field with the value to down count. If they equal to each other, interrupt is trigger.

23.4.19 TIMROT Version Register (HW_TIMROT_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_TIMROT_VERSION.B.MAJOR != 1) Error();
```

Address: HW_TIMROT_VERSION – 8006_8000h base + 120h offset = 8006_8120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_TIMROT_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 24
Debug UART (DUART)

24.1 Debug UART Overview

The Debug UART performs:

- Serial-to-parallel conversion on data received from a peripheral device
- Parallel-to-serial conversion on data transmitted to the peripheral device

The CPU reads and writes data and control/status information through the APBX interface. The transmit and receive paths are buffered with internal FIFO memories, enabling up to 32 bytes to be stored independently in both transmit and receive modes.

The Debug UART includes a programmable baud rate generator that creates a transmit and receive internal clock from the 24-MHz UART internal reference clock input UARTCLK. XCLK is not tied to the UARTCLK in the i.MX28.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 115 Kb/s. Figure 24-1 shows a block diagram of the Debug UART. The Debug UART operation and baud rate values are controlled by the line control register (HW_UARTDBGLCR_H, HW_UARTDBGIBRD, and HW_UARTDBGFBRD).

The Debug UART can generate a single combined interrupt, so output is asserted if any individual interrupt is asserted and unmasked. Interrupt sources include the receive (including timeout), transmit, modem status, and error conditions.

If a framing, parity, or break error occurs during reception, the appropriate error bit is set, and is stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately, and FIFO data is prevented from being overwritten. You can program the FIFOs to be one-byte deep, providing a conventional double-buffered UART interface.

The modem status input signal Clear To Send (CTS) and output modem control line Request To Send (RTS) are supported. A programmable hardware flow control feature uses the nUARTCTS input and the nUARTRTS output to automatically control the serial data flow.

Note

Names for the external pins used by the Debug UART begin with "UART1". Pin names beginning with "UART2" are used by the Application UART.

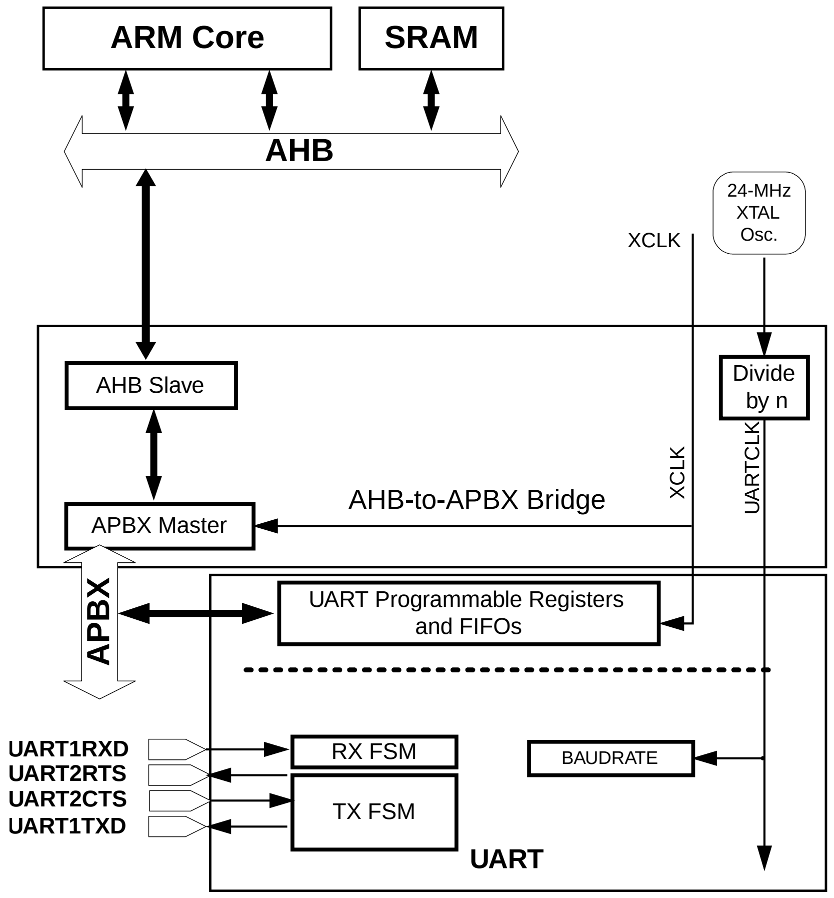


Figure 24-1. Debug UART Block Diagram

24.2 Operation

Control data is written to the Debug UART line control register. This register defines:

- Transmission parameters
- Word length
- Buffer mode
- Number of transmitted stop bits
- Parity mode

- Break generation
- Baud rate divisor

24.2.1 Fractional Baud Rate Divider

The baud rate divisor is calculated from the frequency of UARTCLK and the desired baud rate by using the following formula:

```
divisor = (UARTCLK * 4) / baud rate, rounded to the nearest integer
```

The divisor must be between 0x00000040 and 0x003FFFC0, inclusive. Program the lowest 6 bits of the divisor into BAUD_DIVFRAC, and the next 16 bits of the divisor into BAUD_DIVINT.

In the debug UART, HW_UARTDBGLCR_H, HW_UARTDBGIBRD, and HW_UARTDBGFBRD form a single 30-bit wide register (UARTLCR) that is updated on a single write strobe generated by an HW_UARTDBGLCR_H write. So, in order to internally update the contents of HW_UARTDBGIBRD or HW_UARTDBGFBRD, a write to HW_UARTDBGLCR_H must always be performed at the end.

24.2.2 UART Character Frame

Figure 24-2 illustrates the UART character frame.

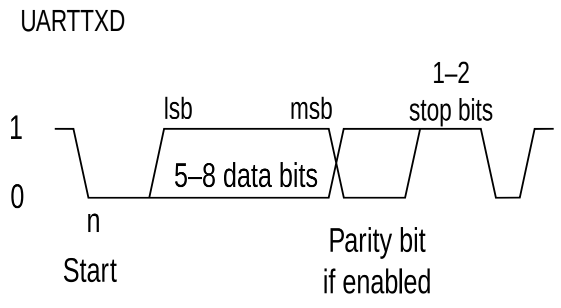


Figure 24-2. Debug UART Character Frame

24.2.3 Data Transmission or Reception

Data received or transmitted is stored in two 32-byte FIFOs, though the receive FIFO has an extra four bits per character for status information.

For transmission, data is written into the transmit FIFO. If the Debug UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UARTLCR_H. Data continues to be transmitted until there is no data left in the transmit FIFO.

The BUSY signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. BUSY is negated only when the transmit FIFO is empty and the last character has been transmitted from the shift register, including the stop bits. BUSY can be asserted HIGH even though the Debug UART might no longer be enabled.

For each sample of data, three readings are taken and the majority value is kept. In the following paragraphs, the middle sampling point is defined and one sample is taken either side of it.

- When the receiver is idle (UARTRXD continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter in normal UART mode to allow for the shorter logic 0 pulses (half way through a bit period).
- The start bit is valid if UARTRXD is still LOW on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored. If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked, if parity mode was enabled.
- Lastly, a valid stop bit is confirmed if UARTRXD is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word (see Table 24-1).

24.2.4 Error Bits

Three error bits are stored in bits [10:8] of the receive FIFO and are associated with a particular character. An additional error indicating an overrun error is stored in bit 11 of the receive FIFO.

24.2.5 Overrun Bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full, and the next character is completely received in the shift register. The data in the shift register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO, and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared. Table 24-1 shows the bit functions of the receive FIFO.

Table 24-1. Receive FIFO Bit Functions

FIFO bit	Function
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

24.2.6 Disabling the FIFOs

FIFOs can be disabled. In this case, the transmit and receive sides of the UART have one-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read.

In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a one-byte register.

24.3 Programmable Registers

UARTDBG Hardware Register Format Summary

HW_UARTDBG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8007_4000	UART Data Register (HW_UARTDBG_DR)	32	R/W	0000_0000h	24.3.1/1534
8007_4004	UART Receive Status Register (Read) / Error Clear Register (Write) (HW_UARTDBG_ECR)	32	R/W	0000_0000h	24.3.2/1535
8007_4018	UART Flag Register (HW_UARTDBG_FR)	32	R	0000_0090h	24.3.3/1536
8007_4020	UART IrDA Low-Power Counter Register (HW_UARTDBG_ILPR)	32	R/W	0000_0000h	24.3.4/1537
8007_4024	UART Integer Baud Rate Divisor Register (HW_UARTDBG_IBRD)	32	R/W	0000_0000h	24.3.5/1537
8007_4028	UART Fractional Baud Rate Divisor Register (HW_UARTDBG_FBRD)	32	R/W	0000_0000h	24.3.6/1538
8007_402C	UART Line Control Register, HIGH Byte (HW_UARTDBG_H)	32	R/W	0000_0000h	24.3.7/1539
8007_4030	UART Control Register (HW_UARTDBG_CR)	32	R/W	0000_0300h	24.3.8/1539
8007_4034	UART Interrupt FIFO Level Select Register (HW_UARTDBG_IFLS)	32	R/W	0000_0012h	24.3.9/1541

HW_UARTDBG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8007_4038	UART Interrupt Mask Set/Clear Register (HW_UARTDBG_IMSC)	32	R/W	0000_0000h	24.3.10/1542
8007_403C	UART Raw Interrupt Status Register (HW_UARTDBG_RIS)	32	R	0000_0000h	24.3.11/1543
8007_4040	UART Masked Interrupt Status Register (HW_UARTDBG_MIS)	32	R	0000_0000h	24.3.12/1545
8007_4044	UART Interrupt Clear Register (HW_UARTDBG_ICR)	32	R/W	0000_0000h	24.3.13/1546
8007_4048	UART DMA Control Register (HW_UARTDBG_DMACR)	32	R/W	0000_0000h	24.3.14/1547

24.3.1 UART Data Register (HW_UARTDBG_DR)

Debug Uart Data Register. For words to be transmitted: 1) If the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO 2) If the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the PrimeCell UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted. For received words: 1) If the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) are pushed onto the 12-bit wide receive FIFO 2) If the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data byte is read by performing reads from the DR register along with the corresponding status information. The status information can also be read by a read of the RSR_ECR register.

Address: HW_UARTDBG_DR – 8007_4000h base + 0h offset = 8007_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																RESERVED				OE	BE	PE	FE	DATA							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_DR field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–12 Reserved	This bitfield is reserved. Reserved.
11 OE	Overrun Error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.

HW_UARTDBG_DR field descriptions (continued)

Field	Description
10 BE	Break Error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.
9 PE	Parity Error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the LCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8 FE	Framing Error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.
7–0 DATA	Receive (read) data character. Transmit (write) data character.

24.3.2 UART Receive Status Register (Read) / Error Clear Register (Write) (HW_UARTDBG_ECR)

The RSR_ECR register is the receive status register/error clear register. Receive status can also be read from RSR_ECR. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from DR prior to reading RSR_ECR. The status information for overrun is set immediately when an overrun condition occurs. A write to RSR_ECR clears the framing, parity, break, and overrun errors.

Address: HW_UARTDBG_ECR – 8007_4000h base + 4h offset = 8007_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																								EC				OE	BE	PE	FE
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_ECR field descriptions

Field	Description
31–8 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
7–4 EC	Error Clear. Any write to this bitfield clears the framing, parity, break, and overrun errors. The value is unpredictable when read.
3 OE	Overrun Error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by any write to RSR_ECR. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.

HW_UARTDBG_ECR field descriptions (continued)

Field	Description
2 BE	Break Error.
1 PE	Parity Error.
0 FE	Framing Error.

24.3.3 UART Flag Register (HW_UARTDBG_FR)

The FR register is the flag register.

Address: HW_UARTDBG_FR – 8007_4000h base + 18h offset = 8007_4018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED							RI	TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS
W																
Reset	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0

HW_UARTDBG_FR field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–9 Reserved	This bitfield is reserved. Reserved, do not modify, read as zero.
8 RI	Ring Indicator. This bit is the complement of the UART ring indicator (nUARTRI) modem status input. That is, the bit is 1 when the modem status input is 0.
7 TXFE	Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the LCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.
6 RXFF	Receive FIFO Full.
5 TXFF	Transmit FIFO Full.

HW_UARTDBG_FR field descriptions (continued)

Field	Description
4 RXFE	Receive FIFO Empty.
3 BUSY	UART Busy.
2 DCD	Data Carrier Detect.
1 DSR	Data Set Ready.
0 CTS	Clear To Send.

24.3.4 UART IrDA Low-Power Counter Register (HW_UARTDBG_ILPR)

The ILPR register is the IrDA Low-Power Counter Register. This is an 8-bit read/write register which stores a low-power counter divisor value used to divde down the UARTCLK to generate the IrLPBaud16 signal.

Address: HW_UARTDBG_ILPR – 8007_4000h base + 20h offset = 8007_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																								ILPDVSR							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_ILPR field descriptions

Field	Description
31–8 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
7–0 ILPDVSR	IrDA Low Power Divisor [7:0]. 8-bit low-power divisor value.

24.3.5 UART Integer Baud Rate Divisor Register (HW_UARTDBG_IBRD)

The IBRD register is the integer part of the baud rate divisor value.

Address: HW_UARTDBG_IBRD – 8007_4000h base + 24h offset = 8007_4024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																BAUD_DIVINT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_IBRD field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–0 BAUD_DIVINT	Baud Rate Integer [15:0]. The integer baud rate divisor.

24.3.6 UART Fractional Baud Rate Divisor Register (HW_UARTDBG_FBRD)

The FBRD register is the fractional part of the baud rate divisor value.

Address: HW_UARTDBG_FBRD – 8007_4000h base + 28h offset = 8007_4028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE[15:8]								RESERVED		BAUD_DIVFRAC					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_FBRD field descriptions

Field	Description
31–8 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
7–6 Reserved	This bitfield is reserved. Not documented.
5–0 BAUD_DIVFRAC	Baud Rate Fraction [5:0]. The fractional baud rate divisor.

24.3.7 UART Line Control Register, HIGH Byte (HW_UARTDBG_H)

The LCR_H is the Line Control Register.

Address: HW_UARTDBG_H – 8007_4000h base + 2Ch offset = 8007_402Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																RESERVED								SPS	WLEN		FEN	STP2	EPS	PEN	BRK
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_H field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–8 Reserved	This bitfield is reserved. Reserved, do not modify, read as zero.
7 SPS	Stick Parity Select. When bits 1, 2, and 7 of the LCR_H register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled.
6–5 WLEN	Word length [1:0]. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits, 10 = 7 bits, 01 = 6 bits, 00 = 5 bits.
4 FEN	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, the FIFOs are disabled (character mode); that is, the FIFOs become 1-byte-deep holding registers.
3 STP2	Two Stop Bits Select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2 EPS	Even Parity Select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (PEN, bit 1) being cleared to 0.
1 PEN	Parity Enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.
0 BRK	Send Break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

24.3.8 UART Control Register (HW_UARTDBG_CR)

The CR is the Control Register.

Address: HW_UARTDBG_CR – 8007_4000h base + 30h offset = 8007_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTSEN	RTSEN	OUT2	OUT1	RTS	DTR	RXE	TXE	LBE	RESERVED				SIRLP	SIREN	UARTEN
W										▨						
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

HW_UARTDBG_CR field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15 CTSEN	CTS Hardware Flow Control Enable.
14 RTSEN	RTS Hardware Flow Control Enable.
13 OUT2	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. Not Implemented.
12 OUT1	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. Not Implemented.
11 RTS	Request To Send.
10 DTR	Data Transmit Ready. Not Implemented.
9 RXE	Receive Enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for the UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping.
8 TXE	Transmit Enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for the UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7 LBE	Loop Back Enable. This feature reduces the amount of external coupling required during system test. If this bit is set to 1, the UARTTXD path is fed through to the UARTRXD path. When this bit is set, the modem outputs are also fed through to the modem inputs.
6–3 Reserved	This bitfield is reserved. Reserved, do not modify, read as zero.
2 SIRLP	IrDA SIR low-power mode. Not Supported.

HW_UARTDBG_CR field descriptions (continued)

Field	Description
1 SIREN	SIR Enable. Not Supported.
0 UARTEN	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for the UART signals. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

24.3.9 UART Interrupt FIFO Level Select Register (HW_UARTDBG_IFLS)

The IFLS register is the Interrupt FIFO Level Select Register. You can use the IFLS register to define the FIFO level at which the UARTTXINTR and UARTRXINTR are triggered. The interrupts are generated based on a transition through a level rather than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level. The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

Address: HW_UARTDBG_IFLS – 8007_4000h base + 34h offset = 8007_4034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UNAVAILABLE																RESERVED										RXIFLSEL			TXIFLSEL		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0

HW_UARTDBG_IFLS field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–6 Reserved	This bitfield is reserved. Reserved, do not modify, read as zero.
5–3 RXIFLSEL	Receive Interrupt FIFO Level Select. The trigger points for the receive interrupt are as follows: 0x0 **ONE_EIGHT** — Trigger when FIFO becomes at least one-eight full. 0x1 **ONE_QUARTER** — Trigger when FIFO becomes at least one-quarter full. 0x2 **ONE_HALF** — Trigger when FIFO becomes at least one-half full. 0x3 **THREE_QUARTERS** — Trigger when FIFO becomes at least three-quarters full. 0x4 **SEVEN_EIGHTHS** — Trigger when FIFO becomes at least seven-eights full. 0x5 **INVALID5** — Reserved. 0x6 **INVALID6** — Reserved. 0x7 **INVALID7** — Reserved.
2–0 TXIFLSEL	Transmit Interrupt FIFO Level Select. The trigger points for the transmit interrupt are as follows: 0x0 **ONE_EIGHT** — Trigger when FIFO becomes equal or less than one-eight full. 0x1 **ONE_QUARTER** — Trigger when FIFO becomes equal or less than one-quarter full.

HW_UARTDBG_IFLS field descriptions (continued)

Field	Description
	0x2 **ONE_HALF** — Trigger when FIFO becomes equal or less than one-half full. 0x3 **THREE_QUARTERS** — Trigger when FIFO becomes equal or less than three-quarters full. 0x4 **SEVEN_EIGHTHS** — Trigger when FIFO becomes equal or less than seven-eights full. 0x5 **INVALID5** — Reserved. 0x6 **INVALID6** — Reserved. 0x7 **INVALID7** — Reserved.

24.3.10 UART Interrupt Mask Set/Clear Register (HW_UARTDBG_IMSC)

The IMSC register is the Interrupt Mask Set/Clear Register. On a read, this register gives the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask. Special Note: Here Mask means Enable.Mask=1 means that bit interrupt is enabled

Address: HW_UARTDBG_IMSC – 8007_4000h base + 38h offset = 8007_4038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED					OEIM	BEIM	PEIM	FEIM	RTIM	TXIM	RXIM	DSRMIM	DCDMIM	CTSMIM	RIMIM
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_IMSC field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–11 Reserved	This bitfield is reserved. Reserved, do not modify, read as zero.
10 OEIM	Overrun Error Interrupt Mask. On a read, the current mask for the OEIM interrupt is returned. On a write of 1, the mask of the OEIM interrupt is set. A write of 0 clears the mask.
9 BEIM	Break Error Interrupt Mask.
8 PEIM	Parity Error Interrupt Mask.

HW_UARTDBG_IMSC field descriptions (continued)

Field	Description
7 FEIM	Framing Error Interrupt Mask.
6 RTIM	Receive Timeout Interrupt Mask.
5 TXIM	Transmit Interrupt Mask.
4 RXIM	Receive Interrupt Mask.
3 DSRMIM	nUARTDSR Modem Interrupt Mask.
2 DCDMIM	nUARTDCD Modem Interrupt Mask.
1 CTSMIM	nUARTCTS Modem Interrupt Mask.
0 RIMIM	nUARTRI Modem Interrupt Mask.

24.3.11 UART Raw Interrupt Status Register (HW_UARTDBG_RIS)

The RIS register is the Raw Interrupt Status Register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect. All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Address: HW_UARTDBG_RIS – 8007_4000h base + 3Ch offset = 8007_403Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED					OERIS	BERIS	PERIS	FERIS	RTRIS	TXRIS	RXRIS	DSRRMIS	DCDRMIS	CTSRMIS	RIRMIS
W	▨					▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_RIS field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–11 Reserved	This bitfield is reserved. Reserved, read as zero, do not modify.
10 OERIS	Overrun Error Interrupt Status.
9 BERIS	Break Error Interrupt Status.
8 PERIS	Parity Error Interrupt Status.
7 FERIS	Framing Error Interrupt Status.
6 RTRIS	Receive Timeout Interrupt Status.
5 TXRIS	Transmit Interrupt Status.
4 RXRIS	Receive Interrupt Status.
3 DSRRMIS	nUARTDSR Modem Interrupt Status.
2 DCDRMIS	nUARTDCD Modem Interrupt Status.
1 CTSRMIS	nUARTCTS Modem Interrupt Status.
0 RIRMIS	nUARTRI Modem Interrupt Status.

24.3.12 UART Masked Interrupt Status Register (HW_UARTDBG_MIS)

The MIS register is the Masked Interrupt Status Register. It is a read-only register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect. All the bits except for the modem status interrupt bits (bits 3 to 0) are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Address: HW_UARTDBG_MIS – 8007_4000h base + 40h offset = 8007_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED					OEMIS	BEMIS	PEMIS	FEMIS	RTMIS	TXMIS	RXMIS	DSRMMIS	DCDMMIS	CTSMMIS	RIMMIS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_MIS field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–11 Reserved	This bitfield is reserved. Reserved, read as zero, do not modify.
10 OEMIS	Overrun Error Masked Interrupt Status.
9 BEMIS	Break Error Masked Interrupt Status.
8 PEMIS	Parity Error Masked Interrupt Status.
7 FEMIS	Framing Error Masked Interrupt Status.
6 RTMIS	Receive Timeout Masked Interrupt Status.

HW_UARTDBG_MIS field descriptions (continued)

Field	Description
5 TXMIS	Transmit Masked Interrupt Status.
4 RXMIS	Receive Masked Interrupt Status.
3 DSRMMIS	nUARTDSR Modem Masked Interrupt Status.
2 DCDMMIS	nUARTDCD Modem Masked Interrupt Status.
1 CTSMMIS	nUARTCTS Modem Masked Interrupt Status.
0 RIMMIS	nUARTRI Modem Masked Interrupt Status.

24.3.13 UART Interrupt Clear Register (HW_UARTDBG_ICR)

The ICR register is the Interrupt Clear Register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect.

Address: HW_UARTDBG_ICR – 8007_4000h base + 44h offset = 8007_4044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED															
W						OEIC	BEIC	PEIC	FEIC	RTIC	TXIC	RXIC	DSRMIC	DCDMIC	CTSMIC	RIMIC
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_ICR field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.

HW_UARTDBG_ICR field descriptions (continued)

Field	Description
15–11 Reserved	This bitfield is reserved. Reserved, read as zero, do not modify.
10 OEIC	Overrun Error Interrupt Clear.
9 BEIC	Break Error Interrupt Clear.
8 PEIC	Parity Error Interrupt Clear.
7 FEIC	Framing Error Interrupt Clear.
6 RTIC	Receive Timeout Interrupt Clear.
5 TXIC	Transmit Interrupt Clear.
4 RXIC	Receive Interrupt Clear.
3 DSRMIC	nUARTDSR Modem Interrupt Clear.
2 DCDMIC	nUARTDCD Modem Interrupt Clear.
1 CTSMIC	nUARTCTS Modem Interrupt Clear.
0 RIMIC	nUARTRI Modem Interrupt Clear.

24.3.14 UART DMA Control Register (HW_UARTDBG_DMACR)

This register is reserved in this system.

Address: HW_UARTDBG_DMACR – 8007_4000h base + 48h offset = 8007_4048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	UNAVAILABLE															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RESERVED													DMAONERR	TXDMAE	RXDMAE
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTDBG_DMACR field descriptions

Field	Description
31–16 UNAVAILABLE	The UART IP only implements 16 and 8-bit registers, so the top 2 or 3 bytes of every 32-bit register are always unavailable.
15–3 Reserved	This bitfield is reserved. Reserved.
2 DMAONERR	Reserved.
1 TXDMAE	Reserved.
0 RXDMAE	Reserved.

Chapter 25
Controller Area Network (FlexCAN)

25.1 FlexCAN Introduction

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. A general block diagram is shown in Figure 25-1, which describes the main sub-blocks implemented in the FlexCAN module, including two embedded memories, one for storing Message Buffers (MB) and another one for storing Rx Individual Mask Registers. The functions of the sub-modules are described in subsequent sections.

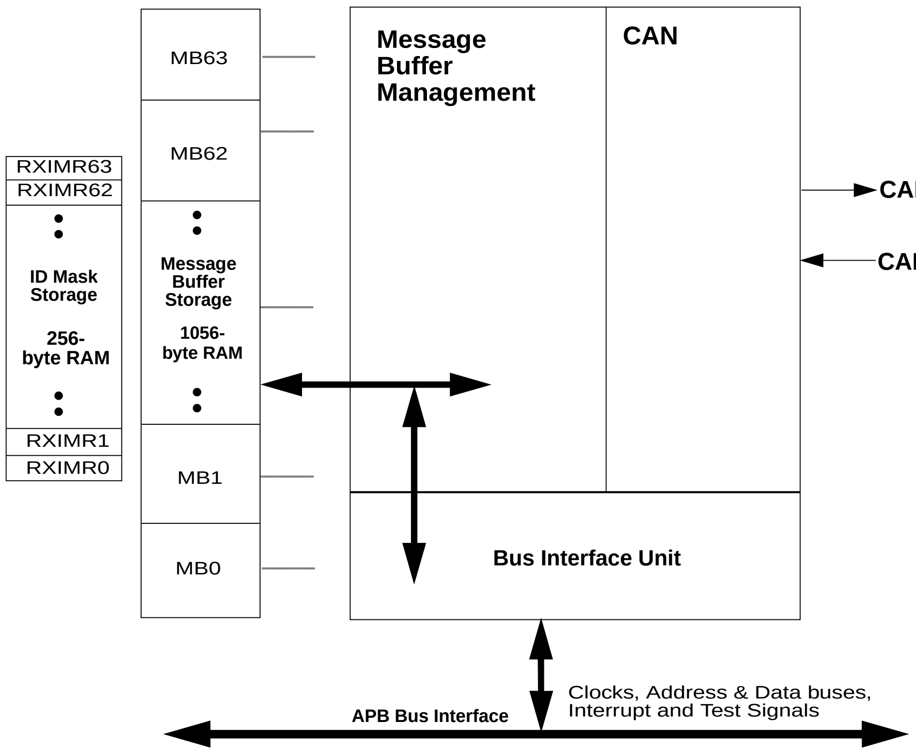


Figure 25-1. FlexCAN Block Diagram

25.2 Overview

The CAN protocol was primarily, but not only, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FlexCAN module is a full implementation of the CAN protocol specification, Version 2.0 B, which supports both standard and extended message frames. The Message Buffers are stored in an embedded RAM dedicated to the FlexCAN module.

The CAN Protocol Interface (CPI) sub-module manages the serial communication on the CAN bus, requesting RAM access for receiving and transmitting message frames, validating received messages and performing error handling. The Message Buffer Management (MBM) sub-module handles Message Buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms. The Bus Interface Unit (BIU) sub-module controls the access to and from the internal interface bus, in order to establish connection to the CPU and other blocks. Clocks, address and data buses, interrupt outputs and test signals are accessed through the Bus Interface Unit, which is compliant to APB Bus Specification.

25.2.1 Features

The FlexCAN module includes the following distinctive Features:

- Full Implementation of the CAN protocol specification, Version 2.0B
    - Standard data and remote frames
    - Extended data and remote frames
    - Zero to eight bytes data length
    - Programmable bit rate up to 1 Mb/sec
    - Content-related addressing
- 64 Message Buffers of zero to eight bytes data length
- Each MB configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Message Buffer
- Includes either 1056 bytes (64 MBs) of RAM used for MB storage
- Includes either 256 bytes (64 MBs) of RAM used for individual Rx Mask Registers
- Full featured Rx FIFO with storage capacity for six frames and internal pointer handling

- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 8 extended, 16 standard or 32 partial (8 bits) IDs, with individual masking capability
- Selectable backward compatibility with previous FlexCAN version
- CAN Protocol Interface with clock only from crystal oscillator
- Unused MB and Rx Mask Register space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity
- Configurable Glitch filter width to filter the noise on CAN bus when waking up

25.2.2 Modes of Operation

The FlexCAN module has four functional modes: Normal Mode, Freeze Mode, Listen-Only Mode and Loop-Back Mode. There are also two low power modes: Disable Mode, Stop Mode.

- Normal Mode :

  In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled.

- Freeze Mode:

  It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze Mode for more information.

- Listen-Only Mode:

  The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode:

  The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode:

  This low power mode is entered when the MDIS bit in the MCR Register is asserted. When disabled, the module shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See Module Disable Mode for more information.

- Stop Mode:

  This low power mode is entered when Stop Mode is requested at ARM chip level.When in Stop Mode, the module puts itself in an inactive state and then informs the ARM that the clocks can be shut down globally. Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop Mode for more information.

25.3 Message Buffer Structure

The Message Buffer structure used by the FlexCAN module is represented in Table 25-1. Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification (Version 2.0 Part B) are represented.

Table 25-1. Message Buffer Structure

	31 … 28	27 … 24		22	21	20	19 … 16	15 … 8	7 … 0
$0	▨	CODE	▨	SRR	IDE	RTR	LENGTH	TIME STAMP	
$4	PRIO	ID (Standard/Extended)						ID (Extended)	
$8	Data Byte 0		Data Byte 1					Data Byte 2	Data Byte 3
$C	Data Byte 4		Data Byte 5					Data Byte 6	Data Byte 7
	▨	= Unimplemented or Reserved							

CODE—Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the Flexcan module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 25-2 and Table 25-3. See Overview for additional information.

Table 25-2. Message Buffer Code for Rx buffers

Rx Code BE-FORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0000	INACTIVE: MB is not active.	—	MB does not participate in the matching process.
0100	EMPTY: MB is active and empty.	0010	MB participates in the matching process. When a frame is received successfully, the code is automatically updated to FULL.
0010	FULL: MB is full.	0010	The act of reading the C/S word followed by unlocking the MB does not make the code return to EMPTY. It remains FULL. If a new frame is written to the MB after the C/S word was read and the MB was unlocked, the code still remains FULL.
		0110	If the MB is FULL and a new frame is overwritten to this MB before the CPU had time to read it, the code is automatically updated to OVERRUN. Refer to Matching Process for details about overrun behavior.
0110	OVERRUN: a frame was overwritten into a full buffer.	0010	If the code indicates OVERRUN but the CPU reads the C/S word and then unlocks the MB, when a new frame is written to the MB the code returns to FULL.
		0110	If the code already indicates OVERRUN, and yet another new frame must be written, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Matching Process for details about overrun behavior.

Rx Code BEFORE Rx New Frame	Description	Rx Code AFTER Rx New Frame	Comment
0XY1[1]	BUSY: Flexcan is updating the contents of the MB. The CPU must not access the MB.	0010	An EMPTY buffer was written with a new frame (XY was 01).
		0110	A FULL/OVERRUN buffer was overwritten (XY was 11).

1. For Tx MBs (see Table 25-3), the BUSY bit should be ignored upon read, except when the AEN bit is set in the MCR register.

Table 25-3. Message Buffer Code for Tx buffers

RTR	Initial Tx code	Code after successful transmission	Description
X	1000	—	INACTIVE: MB does not participate in the arbitration process.
X	1001	—	ABORT: MB was configured as Tx and CPU aborted the transmission. This code is only valid when AEN bit in MCR is asserted. MB does not participate in the arbitration process.
0	1100	1000	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1	1100	0100	Transmit remote frame unconditionally once. After transmission, the MB automatically becomes an Rx MB with the same ID.
0	1010	1010	Transmit a data frame whenever a remote request frame with the same ID is received. This MB participates simultaneously in both the matching and arbitration processes. The matching process compares the ID of the incoming remote request frame with the ID of the MB. If a match occurs this MB is allowed to participate in the current arbitration process and the Code field is automatically updated to '1110' to allow the MB to participate in future arbitration runs. When the frame is eventually transmitted successfully, the Code automatically returns to '1010' to restart the process again.
0	1110	1010	This is an intermediate code that is automatically written to the MB by the MBM as a result of match to a remote request frame. The data frame will be transmitted unconditionally once and then the code will automatically return to '1010'. The CPU can also write this code with the same effect.

SRR—Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as arbitration loss.

1 = Recessive value is compulsory for transmission in Extended Format frames

0 = Dominant is not a valid value for transmission in Extended Format frames

IDE—ID Extended Bit

This bit identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR—Remote Transmission Request

This bit is used for requesting transmissions of a data frame. If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission.

1 = Indicates the current MB has a Remote Frame to be transmitted

0 = Indicates the current MB has a Data Frame to be transmitted

LENGTH—Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset $8 through $F of the MB space (see Table 25-1). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field.

TIME STAMP—Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO—Local priority

This 3-bit field is only used when LPRIO_EN bit is set in MCR and it only makes sense for Tx buffers. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration Process.

ID—Frame Identifier

In Standard Frame format, only the eleven most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.

DATA—Data Field

Up to eight bytes can be used for a data frame. For Rx frames, the data is stored as it is received from the CAN bus. For Tx frames, the CPU prepares the data field to be transmitted within the frame.

25.3.1 Rx FIFO Structure

When the FEN bit is set in the MCR, the memory area from $80 to $FF (which is normally occupied by MBs 0 to 7) is used by the reception FIFO engine. Table 25-4 shows the Rx FIFO data structure. The region $0-$C contains a MB structure which is the port through which the CPU reads data from the FIFO (the oldest frame received and not read yet). The region $10-$DF is reserved for internal use of the FIFO engine. The region $E0-$FF contains an 8-entry ID table that specifies filtering criteria for accepting frames into the FIFO. Table 25-5 shows the three different formats that the elements of the ID table can assume, depending on the IDAM field of the MCR. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 25-4. Rx FIFO Structure

	31			28	27			24		22	21	20	19			16	15							8	7							0
$0	▨									SRR	IDE	RTR	LENGTH				TIME STAMP															
$4	▨			ID (Standard/Extended)												ID (Extended)																
$8	Data Byte 0								Data Byte 1								Data Byte 2								Data Byte 3							
$C	Data Byte 4								Data Byte 5								Data Byte 6								Data Byte 7							
$10 to $DF	Reserved																															
$E0	ID Table 0																															
$E4	ID Table 1																															
$E8	ID Table 2																															
$EC	ID Table 3																															
$F0	ID Table 4																															
$F4	ID Table 5																															
$F8	ID Table 6																															
$FC	ID Table 7																															

▨ = Unimplemented or Reserved

Table 25-5. ID Table 0 - 7

Format	31	30	29–24	23–16	15	14	13–8	7–1	0
A	REM	EXT	RXIDA (Standard = 29-19, Extended = 29-1)						▨
B	REM	EXT	RXIDB_0 (Standard = 29-19, Extended = 29-16)		REM	EXT	RXIDB_1 (Standard = 13-3, Extended = 13-0)		
C	RXIDC_0 (Std/Ext = 31-24)			RXIDC_1 (Std/Ext = 23-16)	RXIDC_2 (Std/Ext = 15-8)			RXIDC_3 (Std/Ext = 7-0)	
	▨	▨	= Unimplemented or Reserved						

REM — Remote Frame

This bit specifies if Remote Frames are accepted into the FIFO if they match the target ID.

1 = Remote Frames can be accepted and data frames are rejected

0 = Remote Frames are rejected and data frames can be accepted

EXT — Extended Frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the eleven most significant bits are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the eleven most significant bits (a full standard ID) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all eight bits of the field are compared to the eight most significant bits of the received ID.

25.4 Functional Description

25.4.1 Overview

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed by a set of 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message Buffer Structure). The memory corresponding to the first 8 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 8 extended IDs or 16 standard IDs or 32 8-bit ID slices), each one with its own individual mask register. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed on its ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by three local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be active at a given time if it can participate in the matching and arbitration algorithms that are happening at that time. An Rx MB with a 0000 code is inactive (refer to Table 25-2). Similarly, a Tx MB with a 1000 or 1001 code is also inactive (refer to Table 25-3). An MB not programmed with 0000, 1000 or 1001 is temporarily deactivated (does not participate in the current arbitration or matching run) when the ARM writes to the C/S field of that MB (see Message Buffer Deactivation).

The FlexCAN also provide a glitch filter which can filter the noises on CAN bus when the FlexCAN is in the STOP mode. The glitch filter width is configurable by the register HW_CAN_GFWR.

25.4.2 Transmit Process

In order to transmit a CAN frame, the ARM must prepare a Message Buffer for transmission by executing the following procedure:

- If the MB is active (transmission pending), write an ABORT code ('1001') to the Code field of the Control and Status word to request an abortion of the transmission, then read back the Code field and the IFLAG register to check if the transmission was aborted (see Transmission Abort Mechanism). If backwards compatibility is desired (AEN in MCR negated), just write '1000' to the Code field to inactivate the MB but then the pending frame may be transmitted without notification (see Message Buffer Deactivation).
- Write the ID word.
- Write the data bytes.
- Write the Length, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated in the fourth step, it will participate in the arbitration process and eventually be transmitted according to its priority. At the end of the successful transmission, the value of the Free Running Timer is written into the Time Stamp field, the Code field in the Control and Status word is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new Code field after transmission depends on the code that was used to activate the MB in step four (see Table 25-2 and Table 25-3 in Section Message Buffer Structure). When the Abort feature is enabled (AEN in MCR is asserted), after the Interrupt Flag is asserted for a MB configured as transmit buffer, the MB is blocked, therefore the ARM is not able to update it until the Interrupt Flag be negated by ARM. It means that the ARM must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

25.4.3 Arbitration Process

The arbitration process is an algorithm executed by the MBM that scans the whole MB memory looking for the highest priority message to be transmitted. All MBs programmed as transmit buffers will be scanned to find the lowest ID[1] or the lowest MB number or the highest priority, depending on the LBUF and LPRIO_EN bits on the Control Register. The arbitration process is triggered in the following events:

- During the CRC field of the CAN frame
- During the error delimiter field of the CAN frame

1. Actually, if LBUF is negated, the arbitration considers not only the ID, but also the RTR and IDE bits placed inside the ID at the same positions they are transmitted in the CAN frame.

- During Intermission, if the winner MB defined in a previous arbitration was deactivated, or if there was no MB to transmit, but the ARM wrote to the C/S word of any MB after the previous arbitration finished
- When MBM is in Idle or Bus Off state and the ARM writes to the C/S word of any MB
- Upon leaving Freeze Mode

When LBUF is asserted, the LPRIO_EN bit has no effect and the lowest number buffer is transmitted first. When LBUF and LPRIO_EN are both negated, the MB with the lowest ID is transmitted first. But, if LBUF is negated and LPRIO_EN is asserted, the PRIO bits augment the ID used during the arbitration process. With this extended ID concept, arbitration is performed based on the full 32-bit ID and the PRIO bits define which MB should be transmitted first, therefore MBs with PRIO = 000 have higher priority. If two or more MBs have the same priority, the regular ID will determine the priority of transmission. If two or more MBs have the same priority (3 extra bits) and the same regular ID, the lowest MB will be transmitted first.

Once the highest priority MB is selected, it is transferred to a temporary storage space called Serial Message Buffer (SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in HALT or BUS OFF
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) value is bigger.

25.4.4 Receive Process

To be able to receive CAN frames into the mailbox MBs, the ARM must prepare one or more Message Buffers for reception by executing the following steps:

- If the MB has a pending transmission, write an ABORT code ('1001') to the Code field of the Control and Status word to request an abortion of the transmission, then read back the Code field and the IFLAG register to check if the transmission was aborted (see Transmission Abort Mechanism). If backwards compatibility is desired (AEN in

MCR negated), just write '1000' to the Code field to inactivate the MB, but then the pending frame may be transmitted without notification (see Message Buffer Deactivation). If the MB already programmed as a receiver, just write '0000' to the Code field of the Control and Status word to keep the MB inactive.

- Write the ID word
- Write '0100' to the Code field of the Control and Status word to activate the MB

Once the MB is activated in the third step, it will be able to receive frames that match the programmed ID. At the end of a successful reception, the MB is updated by the MBM as follows:

- The value of the Free Running Timer is written into the Time Stamp field
- The received ID, Data (8 bytes at most) and Length fields are stored
- The Code field in the Control and Status word is updated (see Table 25-2 and Table 25-3 in Section Message Buffer Structure)
- A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit

Upon receiving the MB interrupt, the ARM should service the received frame using the following procedure:

- Read the Control and Status word (mandatory – activates an internal lock for this buffer)
- Read the ID field (optional – needed only if a mask was used)
- Read the Data field
- Read the Free Running Timer (optional – releases the internal lock)

Upon reading the Control and Status word, if the BUSY bit is set in the Code field, then the ARM should defer the access to the MB until this bit is negated. Reading the Free Running Timer is not mandatory. If not executed the MB remains locked, unless the ARM reads the C/S word of another MB. Note that only a single MB is locked at a time. The only mandatory ARM read operation is the one on the Control and Status word to assure data coherency (see Data Coherence).

The ARM should synchronize to frame reception by the status flag bit for the specific MB in one of the IFLAG Registers and not by the Code field of that MB. Polling the Code field does not work because once a frame was received and the ARM services the MB (by reading the C/S word followed by unlocking the MB), the Code field will not return to EMPTY. It will remain FULL, as explained in Table 25-2. If the ARM tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the MB, the MB is actually deactivated from any currently ongoing matching process. As a result, a newly received frame matching the ID of that MB may be lost. In summary: *never do polling by reading directly the C/S word of the MBs. Instead, read the IFLAG registers*.

Note that the received ID field is always stored in the matching MB, therefore the contents of the ID field in an MB may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists an Rx matching MB, provided the SRX_DIS bit in the MCR is not asserted. If SRX_DIS is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN frames through the FIFO, the ARM must enable and configure the FIFO during Freeze Mode (see Rx FIFO). Upon receiving the frames available interrupt from FIFO, the ARM should service the received frame using the following procedure:

- Read the Control and Status word (optional – needed only if a mask was used for IDE and RTR bits)
- Read the ID field (optional – needed only if a mask was used)
- Read the Data field
- Clear the frames available interrupt (mandatory – release the buffer and allow the ARM to read the next FIFO entry)

25.4.5 Matching Process

The matching process is an algorithm executed by the MBM that scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the 8-entry ID table from FIFO is scanned first and then, if a match is not found within the FIFO table, the other MBs are scanned. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

When the frame is received, it is temporarily stored in a hidden auxiliary MB called Serial Message Buffer (SMB). The matching process takes place during the CRC field of the received frame. If a matching ID is found in the FIFO table or in one of the regular MBs, the contents of the SMB will be transferred to the FIFO or to the matched MB during the sixth bit of the End-Of-Frame field of the CAN protocol. This operation is called move-in. If any protocol error (CRC, ACK, and so on) is detected, than the move-in operation does not happen.

For the regular mailbox MBs, an MB is said to be free to receive a new frame if the following conditions are satisfied:

- The MB is not locked (see Message Buffer Lock Mechanism)
- The Code field is either EMPTY or it is FULL or OVERRUN but the ARM has already serviced the MB (read the C/S word and then unlocked the MB)

If the first MB with a matching ID is not free to receive the new frame, then the matching algorithm keeps looking for another free MB until it finds one. If it can not find one that is free, then it will overwrite the last matching MB (unless it is locked) and set the Code field to OVERRUN (see Table 25-2 and Table 25-3). If the last matching MB is locked, then the new message remains in the SMB, waiting for the MB to be unlocked (see Message Buffer Lock Mechanism).

Suppose, for example, the FIFO is disabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not free to receive, so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are free to receive, so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the Code field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the ARM to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. The ARM can examine the Time Stamp field of the MBs to determine the order in which the messages are arrived.

The matching algorithm described above can be changed to be the same one used in previous versions of the FlexCAN module. When the BCC bit in MCR is negated, the matching algorithm stops at the first MB with a matching ID that it founds, whether this MB is free or not. As a result, the message queueing feature does not work if the BCC bit is negated.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". The Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed if the BCC bit is asserted and while the module is in Freeze Mode.

FlexCAN also supports an alternate masking scheme with only three mask registers (RGXMASK, RX14MASK and RX15MASK) for backwards compatibility. This alternate masking scheme is enabled when the BCC bit in the MCR Register is negated.

25.4.6 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the ARM must obey the rules described in Transmit Process and Receive Process. Any form of ARM accessing a MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

25.4.6.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform the ARM if the transmission was aborted or if the frame could not be aborted and was transmitted instead. In order to maintain backwards compatibility, the abort mechanism must be explicitly enabled by asserting the AEN bit in the MCR.

In order to abort a transmission, the ARM must write a specific abort code (1001) to the Code field of the Control and Status word. When the abort mechanism is enabled, the active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a MB that is currently being transmitted, or to an MB that was already loaded into the SMB for transmission, the write operation is blocked and the MB is not deactivated, but the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze Mode

If none of conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register and an interrupt to the ARM is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the Code field, the interrupt flag is set in the IFLAG and an interrupt is (optionally) generated to the ARM.

If the ARM writes the abort code before the transmission begins internally, then the write operation is not blocked, therefore the MB is updated and no interrupt flag is set. In this way the ARM just needs to read the abort code to make sure the active MB was deactivated. Although the AEN bit is asserted and the ARM wrote the abort code, in this case the MB is deactivated and not aborted, because the transmission did not start yet. One MB is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

- ARM writes 1001 into the code field of the C/S word
- ARM reads the CODE field and compares it to the value that was written
- If the CODE field that was read is different from the value that was written, the ARM must read the corresponding IFLAG to check if the frame was transmitted or it is being currently transmitted. If the corresponding IFLAG is set, the frame was transmitted. If the corresponding IFLAG is reset, the ARM must wait for it to be set, and then the ARM must read the CODE field to check if the MB was aborted (CODE=1001) or it was transmitted (CODE=1000).

25.4.7 Message Buffer Deactivation

Deactivation is a mechanism provided to maintain data coherence when the ARM writes to the Control and Status word of active MBs out of Freeze Mode. Any ARM write access to the Control and Status word of a MB causes that MB to be excluded from the transmit or receive processes during the current matching or arbitration round. The deactivation is temporary, affecting only for the current match/arbitration round.

The purpose of deactivation is data coherency. The match/arbitration process scans the MBs to decide which MB to transmit or receive. If the ARM updates the MB in the middle of a match or arbitration process, the data of that MB may no longer be coherent, therefore deactivation of that MB is done.

Even with the coherence mechanism described above, writing to the Control and Status word of active MBs when not in Freeze Mode may produce undesirable results. Examples are:

- Matching and arbitration are one-pass processes. If MBs are deactivated after they are scanned, no re-evaluation is done to determine a new match/winner. If a Rx MB with a matching ID is deactivated during the matching process after it was scanned, then this MB is marked as invalid to receive the frame, and FlexCAN will keep looking for another matching MB within the ones it has not scanned yet. If it can not find one, then the message will be lost. Suppose, for example, that two MBs have a matching ID to a received frame, and the user deactivated the first matching MB after FlexCAN has scanned the second. The received frame is lost even if the second matching MB was free to receive.

- If a Tx MB containing the lowest ID is deactivated after FlexCAN has scanned it, then FlexCAN will look for another winner within the MBs that it has not scanned yet. Therefore, it may transmit a MB with ID that may not be the lowest at the time because a lower ID might be present in one of the MBs that it had already scanned before the deactivation.
- There is a point in time until which the deactivation of a Tx MB causes it not to be transmitted (end of move-out). After this point, it is transmitted but no interrupt is issued and the Code field is not updated. In order to avoid this situation, the abort procedures described in Transmission Abort Mechanism should be used.

25.4.8 Message Buffer Lock Mechanism

Besides MB deactivation, FlexCAN has another data coherence mechanism for the receive process. When the ARM reads the Control and Status word of an "active not empty" Rx MB, FlexCAN assumes that the ARM wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the ARM reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB. The MB locking is done to prevent a new frame to be written into the MB while the ARM is reading it.

Note

The locking mechanism only applies to Rx MBs which have a code different than INACTIVE ('0000') or EMPTY[2]*('0100'). Also, Tx MBs can not be locked.*

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the ARM decides to read MB number 5 and at the same time another message with the same ID is arriving. When the ARM reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no MBs that are free to receive, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only

2. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior will be honoured when the BCC bit is negated.

then will be written to the MB. If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the Code field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the Code field is asserted. If the ARM reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Deactivation takes precedence over locking. If the ARM deactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the MB.

25.4.9 Rx FIFO

The receive-only FIFO is enabled by asserting the FEN bit in the MCR. The reset value of this bit is zero to maintain software backwards compatibility with previous versions of the module that did not have the FIFO feature. When the FIFO is enabled, the memory region normally occupied by the first 8 MBs ($80-$FF) is now reserved for use of the FIFO engine (see Rx FIFO Structure). Management of read and write pointers is done internally by the FIFO engine. The ARM can read the received frames sequentially, in the order they were received, by repeatedly accessing a Message Buffer structure at the beginning of the memory.

The FIFO can store up to 6 frames pending service by the ARM. An interrupt is sent to the ARM when new frames are available in the FIFO. Upon receiving the interrupt, the ARM must read the frame (accessing an MB in the $80 address) and then clear the interrupt. The act of clearing the interrupt triggers the FIFO engine to replace the MB in $80 with the next frame in the queue, and then issue another interrupt to the ARM. If the FIFO is full and more frames continue to be received, an OVERFLOW interrupt is issued to the ARM and subsequent frames are not accepted until the ARM creates space in the FIFO by reading one or more frames. A warning interrupt is also generated when 4frames are accumulated in the FIFO.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of 8 32-bit registers that can be configured to one of the following formats (see also Rx FIFO Structure):

- Format A: 8 extended or standard IDs (including IDE and RTR)

- Format B: 16 standard IDs or 16 extended 14-bit ID slices (including IDE and RTR)
- Format C: 32 standard or extended 8-bit ID slices

Note

A chosen format is applied to all 8 registers of the filter table. It is not possible to mix formats within the table.

The eight elements of the filter table are individually affected by the first eight Individual Mask Registers (RXIMR0 - RXIMR7), allowing very powerful filtering criteria to be defined. The rest of the RXIMR, starting from RXIM8, continue to affect the regular MBs, starting from MB8. If the BCC bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO filter table is affected by the legacy mask registers as follows: element 6 is affected by RX14MASK, element 7 is affected by RX15MASK and the other elements (0 to 5) are affected by RXGMASK.

25.4.10.1 Remote Frames

Remote frame is a special kind of frame. The user can program a MB to be a Request Remote Frame by writing the MB as Transmit with the RTR bit set to '1'. After the Remote Request frame is transmitted successfully, the MB becomes a Receive Message Buffer, with the same ID as before.

When a Remote Request frame is received by FlexCAN, its ID is compared to the IDs of the transmit message buffers with the Code field '1010'. If there is a matching ID, then this MB frame will be transmitted. Note that if the matching MB has the RTR bit set, then FlexCAN will transmit a Remote Frame as a response.

A received Remote Request Frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match.

In the case that a Remote Request Frame was received and matched an MB, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx MB, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.

If the Rx FIFO is enabled (bit FEN set in MCR), FlexCAN will not generate an automatic response for Remote Request Frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the ARM. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).

25.4.10.2 Overload Frames

FlexCAN does transmit overload frames due to detection of the following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the seventh bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the eighth bit (last) of Error Frame Delimiter or Overload Frame Delimiter

25.4.10.3 Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of move-in in the TIME STAMP field, providing network behavior with respect to time.

The Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization.

25.4.10.4 Protocol Timing

The FlexCAN only supports the crystal oscillator clock as the CPI clock.

The crystal oscillator clock should be selected whenever a tight tolerance (up to 0.1%) is required in the CAN bus timing. The crystal oscillator clock has better jitter performance than PLL generated clocks.

The FlexCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW.

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(PrescalerValue)}$$

A bit time is subdivided into three segments[3] (reference Figure 25-2 and Table 25-6):

3. For further explanation of the underlying concepts please refer to ISO/DIS 11519–1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 4 to 16 time quanta
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long

$$\text{BitRate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

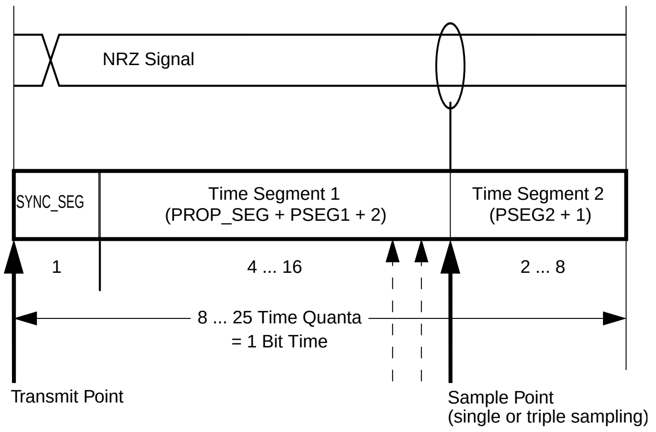


Figure 25-2. Segments within the Bit Time

Table 25-6. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 25-7 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 25-7. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

It is the user's responsibility to ensure the bit time settings are in compliance with the CAN standard. For bit time calculations, use an IPT (Information Processing Time) of 2, which is the value implemented in the FlexCAN module.

25.4.10.5 Arbitration and Matching Timing

During normal transmission or reception of frames, the arbitration, matching, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in Figure 25-3.

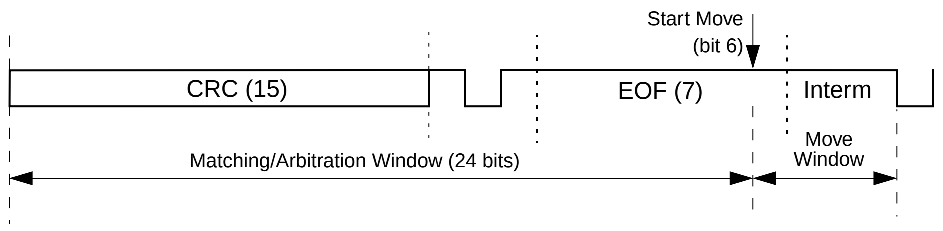


Figure 25-3. Arbitration, Match and Move Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time slot. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 25-7
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, that is the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in Table 25-8

Table 25-8. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	Minimum Ratio
16	8
32	8
64	16

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least eight times the CAN bit rate. The minimum frequency ratio specified in Table 25-8 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

25.4.11.1 Freeze Mode

This mode is entered by asserting the HALT bit in the MCR Register or when the i.MX28 is put into Debug Mode. In both cases, it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable, Stop). When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores the Rx input pin and drives the Tx pin as recessive
- Stops the prescaler, therefore halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible.

Exiting Freeze Mode is done in one of the following ways:

- ARM negates the FRZ bit in the MCR Register
- The MCU is removed from Debug Mode (negating ipg_debug) and/or the HALT bit is negated

Once out of Freeze Mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

25.4.11.2 Module Disable Mode

This low power mode is entered when the MDIS bit in the MCR Register is asserted. If the module is disabled during Freeze Mode, it shuts down the clocks to the CPI and MBM sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or otherwise waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Shuts down the clocks to the CPI and MBM sub-modules
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the ARM to access memory mapped registers, except the Free Running Timer, the Error Counter Register and the Message Buffers, which cannot be accessed when the module is in Disable Mode. Exiting from this mode is done by negating the MDIS bit, which will resume the clocks and negate the LPM_ACK bit.

25.4.11.3 Stop Mode

This is a system low power mode in which all i.MX28 clocks are stopped for maximum power savings. If FlexCAN receives the global Stop Mode request (through ipg_stop) during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends a Stop Acknowledge signal to the ARM, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or otherwise waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the ARM, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- ARM resuming the clocks and removing the Stop Mode request
- ARM resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the ARM. Upon receiving the interrupt, the ARM should resume the clocks and remove the Stop Mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. Table 25-9 details the effect of SLF_WAK and WAK_MSK upon wake-up from Stop Mode. Note that wake-up from Stop Mode only works when both bits are asserted.

Table 25-9. Wake-up from Stop Mode

SLF_WAK	WAK_MSK	i.MX28 Clocks Enabled	Wake-up Interrupt Generated
0	0	No	No
0	1	No	No
1	0	No	No
1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line while in Stop Mode. This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

25.4.11.4 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to ORed interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up).

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the ARM writes it to 1 (unless another interrupt is generated at the same time).

If the Rx FIFO is enabled (bit FEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the FIFO Overflow flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the Frames Available in FIFO flag and bits 4-0 are unused.

A combined interrupt for all MBs is also generated by an OR of all the interrupt sources from MBs. This interrupt gets generated when any of the MBs generates an interrupt. In this case the ARM must read the IFLAG Registers to determine which MB caused the interrupt.

The other 5 interrupt sources (Bus Off, Error, Tx Warning, Rx Warning and Wake Up) generate interrupts like the MB ones, and can be read from the Error and Status Register. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control Register, and the Wake-Up interrupt mask bit is located in the MCR.

25.5 Initialization/Application Information

This section provide instructions for initializing the FlexCAN module.

25.5.1 FlexCAN Initialization Sequence

The FlexCAN module may be reset in three ways:

- i.MX28 level hard reset using ipg_hard_async_reset_b, which resets all memory mapped registers asynchronously
- i.MX28 level soft reset, which resets some of the memory mapped registers synchronously
- SOFT_RST bit in MCR, which has the same effect as the i.MX28 level soft reset

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

The clock source (CLK_SRC bit) should be selected while the module is in Disable Mode. After the clock source is selected and the module is enabled (MDIS bit negated), FlexCAN automatically goes to Freeze Mode. In Freeze Mode, FlexCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FlexCAN is put into Freeze Mode (see Freeze Mode). The following is a generic initialization sequence applicable to the FlexCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the BCC bit
  - Enable the warning interrupts by setting the WRN_EN bit
  - If required, disable frame self reception by setting the SRX_DIS bit
  - Enable the FIFO by setting the FEN bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW

- Determine the bit rate by programming the PRESDIV field
- Determine the internal arbitration mode (LBUF bit)

- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If FIFO was enabled, the 8-entry ID table must be initialized
  - Other entries in each Message Buffer should be initialized as required

- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FlexCAN attempts to synchronize to the CAN bus.

25.6 Programmable Registers

CAN Hardware Register Format Summary

There are Two CAN in chip. CAN0 base address is 0x80032000 and CAN1 base address is 0x80034000

HW_CAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8003_2000	Module Configuration Register (HW_CAN_MCR)	32	R/W	5890_000Fh	25.6.1/1578
8003_2004	Control Register (HW_CAN_CTRL)	32	R/W	0000_0000h	25.6.2/1580
8003_2008	Free Running Timer (HW_CAN_TIMER)	32	R/W	0000_0000h	25.6.3/1582
8003_2010	Rx Global Mask (HW_CAN_RXGMASK)	32	R/W	FFFF_FFFFh	25.6.4/1583
8003_2014	Rx 14 Mask (HW_CAN_RX14MASK)	32	R/W	FFFF_FFFFh	25.6.5/1583
8003_2018	Rx 15 Mask (HW_CAN_RX15MASK)	32	R/W	FFFF_FFFFh	25.6.6/1584
8003_201C	Error Counter Register (HW_CAN_ECR)	32	R/W	0000_0000h	25.6.7/1584
8003_2020	Error and Status Register (HW_CAN_ESR)	32	R/W	0000_0000h	25.6.8/1586
8003_2024	Interrupt Masks 2 Register (HW_CAN_IMASK2)	32	R/W	0000_0000h	25.6.9/1587
8003_2028	Interrupt Masks 1 Register (HW_CAN_IMASK1)	32	R/W	0000_0000h	25.6.10/1588

HW_CAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8003_202C	Interrupt Flags 2 Register (HW_CAN_IFLAG2)	32	R/W	0000_0000h	25.6.11/1588
8003_2030	Interrupt Flags 1 Register (HW_CAN_IFLAG1)	32	R/W	0000_0000h	25.6.12/1589
8003_2034	Glitch Filter Width Register (HW_CAN_GFWR)	32	R/W	0000_007Fh	25.6.13/1590
8003_2080	CAN Messager Buffer Registers (HW_CAN_MBn)	32	R/W	0000_0000h	25.6.14/1590
8003_2880	Rx Individual Mask Registers (HW_CAN_RXIMRn)	32	R/W	0000_0000h	25.6.15/1591

25.6.1 Module Configuration Register (HW_CAN_MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power)and maximum message buffer configuration.Most of the fields in this register can be accessed at any time, except the MAXMB field, which should only be changed while the module is in Freeze Mode. The default value of this register is 0x5890000f.

The MCR defines global system configurations such as the module operation mode (for example, low-power mode) and maximum message buffer configuration. The MAXMB field must only be changed while the module is in freeze mode: all other fields in this register can be accessed at any time.

Address: HW_CAN_MCR – 8003_2000h base + 0h offset = 8003_2000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDIS	FRZ	FEN	HALT	NOT_ RDY	WAK_MSK	SOFT_RST	FRZ_ ACK	SUPV	SLF_ WAK	WRN_EN	LPM_ ACK	WAK_SRC	RSVD3	SRX_ DIS	BCC
W					▨			▨				▨				
Reset	0	1	0	1	1	0	0	0	1	0	0	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2		LPRIO_EN	AEN	RSVD1		IDAM		RSVD0		MAXMB					
W	▨	▨			▨	▨			▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

HW_CAN_MCR field descriptions

Field	Description
31 MDIS	This bit controls whether CAN is enabled or not.When disabled, CAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. 1 Disable the FlexCAN module

HW_CAN_MCR field descriptions (continued)

Field	Description
	0 Enable the FlexCAN module
30 FRZ	The FRZ bit specifies the CAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at MCU level (through assertion of the ipg_debug signal on the IP Interface). When FRZ is asserted, CAN is enabled to enter Freeze Mode. Negation of this bit field causes CAN to exit from Freeze Mode.
29 FEN	This bit controls whether the FIFO feature is enabled or not. When FEN is set, MBs 0 to 7 cannot be used for normal reception and transmission because the corresponding memory region ($80-$FF) is used by the FIFO engine.
28 HALT	Assertion of this bit puts the CAN module into Freeze Mode. The ARM should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by CAN before this bit is cleared. While in Freeze Mode, the ARM has write access to the Error Counter Register, that is otherwise read-only. Freeze Mode can not be entered while CAN is in any of the low power modes.
27 NOT_RDY	This read-only bit indicates that CAN is either in DisableMode, StopMode or Freeze Mode. It is negated once CAN has exited these modes.
26 WAK_MSK	This bit enables the Wake Up Interrupt generation.
25 SOFT_RST	When this bit is asserted, CAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, TCR, ECR, ESR, IMASK1, IMASK2, IFLAG1, IFLAG2. Configuration registers that control the interface to the CAN bus are not affected by soft reset.
24 FRZ_ACK	This read-only bit indicates that CAN is in Freeze Mode and its prescaler is stopped.
23 SUPV	This bit configures some of the CAN registers to be either in Supervisor or Unrestricted memory space.
22 SLF_WAK	This bit enables the Self Wake Up feature when CAN is in Stop Mode. If this bit had been asserted by the time CAN entered StopMode, then CAN will look for a recessive to dominant transition on the bus during these modes.
21 WRN_EN	When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated.
20 LPM_ACK	This read-only bit indicates that CAN is either in Disable Mode or Stop Mode.
19 WAK_SRC	This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wake up.
18 RSVD3	This bit field is reserved.
17 SRX_DIS	This bit defines whether CAN is allowed to receive frames transmitted by itself.
16 BCC	This bit is provided to support Backwards Compatibility with previous CAN versions.

HW_CAN_MCR field descriptions (continued)

Field	Description
15–14 RSVD2	Reserved.
13 LPRIO_EN	This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not.
12 AEN	This bit is supplied for backwards compatibility reasons.
11–10 RSVD1	Reserved.
9–8 IDAM	This 2-bit field identifies the format of the elements of the Rx FIFO filter table
7–6 RSVD0	Reserved.
5–0 MAXMB	This 6-bit field defines the maximum number of message buffers that will take part in the matching and arbitration processes.

25.6.2 Control Register (HW_CAN_CTRL)

This register is defined for specific CAN control features related to the CAN bus. The default value of this register is 0x00000000.

This register is defined for specific FlexCAN control features related to the CAN bus, such as bit rate, programmable sampling point within an Rx bit, loopback mode, listen-only mode, bus-off recovery behavior and interrupt enabling (bus-off, error, warning). It also determines the division factor for the clock prescaler. Most of the fields in this register can only be changed while the module is in disable mode or in freeze mode. Exceptions are the BOFF_MSK, ERR_MSK, TWRN_MSK, RWRN_MSK and BOFF_REC bits, that can be accessed at any time.

Address: HW_CAN_CTRL – 8003_2000h base + 4h offset = 8003_2004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BOFF_MSK	ERR_MSK	CLK_SRC	LPB	TWRN_MSK	RWRN_MSK	RSVD		SMP	BOFF_REC	TSYN	LBUF	LOM	PROP_SEG		
W							▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_CTRL field descriptions

Field	Description
31–24 PRESDIV	This 8-bit field defines the ratio between the CPI clock frequency and the serial clock (SCLK) frequency.
23–22 RJW	This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one RJW re-synchronization.
21–19 PSEG1	This 3-bit field defines the length of phase buffer segment 1 in the bit time.
18–16 PSEG2	This 3-bit field defines the length of phase buffer segment 2 in the bit time.
15 BOFF_MSK	This bit provides a mask for the bus-off interrupt.
14 ERR_MSK	This bit provides a mask for the error interrupt.
13 CLK_SRC	This bit selects the clock source to the CAN Protocol Interface (CPI) to be either the peripheral clock (driven by the PLL) or the crystal oscillator clock. In i.MX28, the clock source is just from the crystal oscillator clock. Setting this bit has no effects.
12 LPB	This bit configures CAN to operate in Loop-Back Mode.
11 TWRN_MSK	This bit provides a mask for the TxWarning Interrupt associated with the TWRN_INT flag in the Error and Status Register.
10 RWRN_MSK	This bit provides amask for the RxWarning Interrupt associated with the RWRN_INT flag in the Error and Status Register.
9–8 RSVD	Reserved.
7 SMP	This bit defines the sampling mode of CAN bits at the Rx input.
6 BOFF_REC	This bit defines how CAN recovers from the bus-off state.
5 TSYN	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0.
4 LBUF	This bit defines the ordering mechanism for Message Buffer transmission.
3 LOM	This bit configures CAN to operate in Listen Only Mode.

HW_CAN_CTRL field descriptions (continued)

Field	Description
2–0 PROP_SEG	This 3-bit field defines the length of the propagation segment in the bit time.

25.6.3 Free Running Timer (HW_CAN_TIMER)

This register represents a 16-bit free-running counter that can be read and written by the ARM. The timer starts at 0x0000 upon reset, counts linearly to 0xFFFF, then wraps back to 0x0000. The default value of this register is 0x00000000.

This register represents a 16-bit free-running counter that can be read and written by the ARM. The timer starts at 0x0000 upon reset, counts linearly to 0xFFFF, then wraps back to 0x0000. The timer is clocked by the FlexCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments each time a bit is received or transmitted. When there is no message on the bus, it counts using the previously-programmed baud rate. During freeze mode, the timer is not incremented. The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the time stamp entry in a message buffer after a successful reception or transmission of a message. Writing to the timer is an indirect operation. The data is first written to an auxiliary register and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data takes some time to actually be written to the register. If desired, software can poll the register to discover when the data was actually written.

Address: HW_CAN_TIMER – 8003_2000h base + 8h offset = 8003_2008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																TIMER															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_TIMER field descriptions

Field	Description
31–16 RSVD	Reserved .
15–0 TIMER	each time a message is received in Message Buffer

25.6.4 Rx Global Mask (HW_CAN_RXGMASK)

The default value of this register is 0xffffffff.

This register is provided for legacy support and for low cost MCUs that do not have the individual masking per message buffer feature. For MCUs supporting individual masks per message buffer, setting the BCC bit in the MCR causes the RXGMASK register to have no effect on the module operation. For MCUs not supporting individual masks per message buffer, this register is always effective. RXGMASK is used as acceptance mask for all Rx message buffers, excluding message buffers 14C15, which have individual mask registers. When the FEN bit in the MCR is set (FIFO enabled), the RXGMASK also applies to all elements of the ID filter table, except elements 6-7, which have individual masks. Setting the BCC bit in MCR causes the RXGMASK register to have no effect on the modules operation. The contents of this register must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

Address: HW_CAN_RXGMASK – 8003_2000h base + 10h offset = 8003_2010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_CAN_RXGMASK field descriptions

Field	Description
31–0 MI	This register is provided for legacy support and for low cost MCUs that do not have the individual masking per message buffer feature.

25.6.5 Rx 14 Mask (HW_CAN_RX14MASK)

The default value of this register is 0xffffffff.

RX14MASK is used as acceptance mask for the identifier in message buffer 14. When the FEN bit in the MCR is set (FIFO enabled), the RXG14MASK also applies to element 6 of the ID filter table. Setting the BCC bit in the MCR causes the RX14MASK register to have no effect on the module operation. This register has the same structure as RXGMASK. It must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

Address: HW_CAN_RX14MASK – 8003_2000h base + 14h offset = 8003_2014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_CAN_RX14MASK field descriptions

Field	Description
31–0 MI	This register is provided for legacy support and for low cost MCUs that do not have the individual masking per message buffer feature.

25.6.6 Rx 15 Mask (HW_CAN_RX15MASK)

The default value of this register is 0xffffffff.

When the BCC bit is cleared, RX15MASK is used as acceptance mask for the identifier in message buffer 15. When the FEN bit in the MCR is set (FIFO enabled), the RXG14MASK also applies to element 7 of the ID filter table. Setting the BCC bit in the MCR causes the RX15MASK register to have no effect on the module operation. This register has the same structure as the RXGMASK. It must be programmed while the module is in freeze mode, and must not be modified when the module is transmitting or receiving frames.

Address: HW_CAN_RX15MASK – 8003_2000h base + 18h offset = 8003_2018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_CAN_RX15MASK field descriptions

Field	Description
31–0 MI	This register is provided for legacy support and for low cost MCUs that do not have the individual masking per message buffer feature.

25.6.7 Error Counter Register (HW_CAN_ECR)

This register has two 8-bit fields which reflect the value of the transmit error counter (tx_err_counter field) and receive error counter (rx_err_counter field).

This register has two 8-bit fields which reflect the value of the transmit error counter (tx_err_counter field) and receive error counter (rx_err_counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module. Both counters are read only except in freeze mode, where they can be written by the ARM. Writing to the error counter register while in freeze mode is an indirect operation. The data is first written to an auxiliary register, and then an internal request/acknowledge procedure across clock domains is executed. All this is transparent to the user, except for the fact that the data takes some time to actually be written to the register. If desired, software can poll the register to discover when the data was actually written. FlexCAN responds to any bus state as described in the protocol, for example, by transmitting an error active or error passive flag, delaying its transmission start time (error passive), and avoiding any influence on the bus when in the bus-off state. The following are the basic rules for FlexCAN bus state transitions: If the value of tx_err_counter or rx_err_counter increases to exceed 127, the FLT_CONF field in the error and status register is updated to reflect error passive state. If the FlexCAN state is error passive, and either tx_err_counter or rx_err_counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the error and status register is updated to reflect error active state. If the value of tx_err_counter increases to be greater than 255, the FLT_CONF field in the error and status register is updated to reflect the bus-off state, and an interrupt is issued. The value of tx_err_counter is then reset to zero. If FlexCAN is in the bus-off state, then tx_err_counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, tx_err_counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the tx_err_counter. when tx_err_counter reaches the value of 128, the FLT_CONF field in the error and status register is updated to be error active and both error counters are reset to zero. at any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the tx_err_counter value. If during system start-up, only one node is operating, then its tx_err_counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the error and status register). After the transition to error passive state, the tx_err_counter does not increment anymore by acknowledge errors. Therefore the device never goes to the bus-off state. If the rx_err_counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to resume to error active state.

Address: HW_CAN_ECR – 8003_2000h base + 1Ch offset = 8003_201Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RX_ERR_COUNTER								TX_ERR_COUNTER							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_ECR field descriptions

Field	Description
31–16 RSVD	Reserved .
15–8 RX_ERR_ COUNTER	Receive error counter
7–0 TX_ERR_ COUNTER	Transmit error counter

25.6.8 Error and Status Register (HW_CAN_ESR)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the ARM.

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the ARM. The reported error conditions are those that occurred since the last time the ARM read this register. The ARM read action clears bits. Bits are status bits. Most bits in this register are read-only, except TWRN_INT, RWRN_INT, BOFF_INT, WAK_INT and ERR_INT, which are interrupt flags that can be cleared by writing 1 to them (writing 0 has no effect).

Address: HW_CAN_ESR – 8003_2000h base + 20h offset = 8003_2020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1														TWRN_ INT	RWRN_ INT
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1_ ERR	BIT0_ ERR	ACK_ ERR	CRC_ ERR	FRM_ ERR	STF_ ERR	TX_ WRN	RX_ WRN	IDLE	TXRX	FLT_CONF		RSVD0	BOFF_INT	ERR_ INT	WAK_INT
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_ESR field descriptions

Field	Description
31–18 RSVD1	Reserved .
17 TWRN_INT	If theWRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from 0 to 1, meaning that the Tx error counter reached
16 RWRN_INT	If theWRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from 0 to 1, meaning that the Rx error counters reached
15 BIT1_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.
14 BIT0_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message.
13 ACK_ERR	This bit indicates that an Acknowledge Error has been detected by the transmitter node
12 CRC_ERR	This bit indicates that a CRC Error has been detected by the receiver node
11 FRM_ERR	This bit indicates that a Form Error has been detected by the receiver node
10 STF_ERR	This bit indicates that a Stuffing Error has been detected.
9 TX_WRN	This bit indicates when repetitive errors are occurring during message transmission.
8 RX_WRN	This bit indicates when repetitive errors are occurring during message reception.
7 IDLE	This bit indicates when CAN bus is in IDLE state.
6 TXRX	This bit indicates if CAN is transmitting or receiving a message when the CAN bus is not in IDLE state.
5–4 FLT_CONF	This 2-bit field indicates the Confinement State of the CAN module
3 RSVD0	Reserved.
2 BOFF_INT	This bit is set when CAN enters Bus Off state
1 ERR_INT	This bit indicates that at least one of the Error Bits (bits 15-10) is set.
0 WAK_INT	When CAN is Stop Mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR Register is set, an interrupt is generated to the ARM.

25.6.9 Interrupt Masks 2 Register (HW_CAN_IMASK2)

The default value of this register is 0x00000000.

This register allows any number of a range of 32 message buffer interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFLAG2 bit is set).

Address: HW_CAN_IMASK2 – 8003_2000h base + 24h offset = 8003_2024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_IMASK2 field descriptions

Field	Description
31–0 BUFM	Each bit enables or disables the respective FlexCAN message buffer interrupt (message buffers 32~63).

25.6.10 Interrupt Masks 1 Register (HW_CAN_IMASK1)

The default value of this register is 0x00000000.

This register enables or disables any number of a range of 32 message buffer interrupts. It contains one interrupt mask bit per buffer, enabling the ARM to determine which buffer generates an interrupt after a successful transmission or reception (that is, when the corresponding IFLAG1 bit is set)

Address: HW_CAN_IMASK1 – 8003_2000h base + 28h offset = 8003_2028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFM																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_IMASK1 field descriptions

Field	Description
31–0 BUFM	This register allows any number of a range of 32 message buffer interrupts to be enabled or disabled.

25.6.11 Interrupt Flags 2 Register (HW_CAN_IFLAG2)

The default value of this register is 0x00000000.

This register defines the flags for 32 message buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1; writing 0 has no effect. When the AEN bit in the MCR is set (abort enabled), while the IFLAG2 bit is set for a message buffer configured as Tx, ARM write access to the corresponding message buffer is blocked.

Address: HW_CAN_IFLAG2 – 8003_2000h base + 2Ch offset = 8003_202Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_IFLAG2 field descriptions

Field	Description
31–0 BUFI	This register defines the flags for 32Message Buffer interrupts.

25.6.12 Interrupt Flags 1 Register (HW_CAN_IFLAG1)

The default value of this register is 0x00000000.

This register defines the flags for 32 message buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt is generated. The interrupt flag must be cleared by writing a 1 to it. Writing 0 has no effect. Setting the abort enabled (AEN) bit in the MCR while the IFLAG1 bit is set for a message buffer configured as Tx blocks the ARMs write access to the corresponding message buffer. Setting the FIFO enable (FEN) bit in the MCR changes the function of the 8 least significant interrupt flags (BUF7ICBUF0I) to support the FIFOs operation. BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO; BUF4ICBUF0I are not used.

Address: HW_CAN_IFLAG1 – 8003_2000h base + 30h offset = 8003_2030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUFI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_IFLAG1 field descriptions

Field	Description
31–0 BUFI	This register defines the flags for 32 Message Buffer interrupts.

25.6.13 Glitch Filter Width Register (HW_CAN_GFWR)

The default value of this register is 0x0000007f.

The Glitch Filter just takes effects when the FlexCAN enters the STOP mode.

Address: HW_CAN_GFWR – 8003_2000h base + 34h offset = 8003_2034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0																								GFWR							
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

HW_CAN_GFWR field descriptions

Field	Description
31–8 RSVD0	Reserved
7–0 GFWR	The Glitch Filter Width Register defines the glitch width which will be filtered.

25.6.14 CAN Messager Buffer Registers (HW_CAN_MBn)

The default value of this register is 0x00000000. There are 64 MB registers (MB0~MB63) n denotes the number from 0 to 63.

See the Messager Buffer structure in previous section.

Address: HW_CAN_MBn – 8003_2000h base + 80h offset = 8003_2080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MBn																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_MBn field descriptions

Field	Description
31–0 MBn	These registers are used as acceptance storage for messages.

25.6.15 Rx Individual Mask Registers (HW_CAN_RXIMRn)

The default value of this register is 0x00000000. There are 64 Rx Individual Mask registers (RXIMR0¨RXIMR63) n denotes the number from 0 to 63.

These registers are used as acceptance masks for ID filtering in Rx message buffers and the FIFO. If the FIFO is not enabled, one mask register is provided for each available message buffer, providing ID masking capability on a per message buffer basis. When the FIFO is enabled (FEN bit in the MCR is set), the first 8 mask registers apply to the 8 elements of the FIFO filter table (on a one-to-one correspondence), while the rest of the registers apply to the regular message buffers, starting from message buffer 8. The individual Rx mask registers are implemented in RAM, so they are not affected by reset and must be explicitly initialized prior to any reception. Furthermore, they can only be accessed by the ARM while the module is in freeze mode. Outside of freeze mode, write accesses are blocked and read accesses return all zeros. Furthermore, if the BCC bit in the MCR is cleared, any read or write operation to these registers results in an access error.

Address: HW_CAN_RXIMRn – 8003_2000h base + 880h offset = 8003_2880h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_CAN_RXIMRn field descriptions

Field	Description
31–0 MI	These registers are used as acceptance masks for ID filtering in Rx message buffers and the FIFO.

Chapter 26
Ethernet Controller (ENET)

26.1 Overview

The ethernet controller (ENET) consists of two MACs (media access controllers), each with it's own dedicated uDMA (unified DMA) module. The MAC module is third party IP from "More Than IP" (MTIP). The MACs interface to 10 Mbps and 100 Mbps Ethernet/IEEE 802.3™ networks. For Ethernet, an external PHY or SerDes device is required to complete the interface to the media. Each MAC supports multiple standard media-independent interfaces. An integrated 3-Port Switch allows the macs to filter and forward traffic at wire-speed to each other or the core, or operate as two independent ports.

The ENET assembly provides two master ports on the AHB Bus. IP provides interface for connecting various FIFOs for data storage (for MAC and Switch).

The ENET MAC is backward compatible with the Freescale FEC controller (Such as the FEC used in i.MX25).

26.1.1 Block Diagram

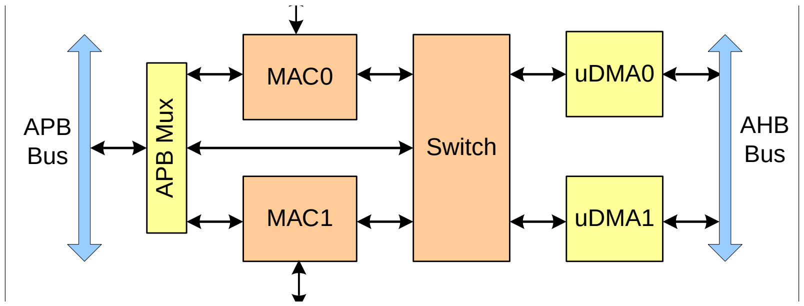


Figure 26-1. ENET Overview Block Diagram

26.1.2 Features

- MAC supports full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking

- Dynamically configurable to support 10/100 Mbps
- Supports full duplex and configurable half duplex operations
- Supports AMD magic packet detection with interrupt for node remote power management
- Supports interface to Fast Ethernet Phy devices through Medium independent interface (MII) working at 25 MHz and Reduced Medium independent interface (RMII) working at 50 MHz
- Provides 64 bit FIFO interface (transmit and receive)
- When operating in Full Duplex mode, implements automated Pause Frame (802.3 x 31A) generation and termination providing flow control without user application intervention
- Implements standard flow control mechanism in full duplex operation mode
- In half duplex mode, provides full collision support, including jamming, backoff and automatic retransmission
- Support for VLAN tagged frames as per IEEE 802.1Q
- Programmable MAC address
- Multicast and unicast address filtering on receive based on 64 entries hash table thus reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistic indicators for frame traffic as well as errors (alignment, CRC error and so on) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and Remote network monitoring (RFC 2819)
- Multiple internal loopback options
- MDIO master interface for PHY device configuration
- Support for all IEEE 1588 frames
- Reference clock can be chosen independently of network speed
- Software programmable precise time stamping of Ingress frames and Egress frames
- Hardware and software controllable timer synchronization

26.2 Unified DMA Block Guide

This section includes an introduction, descriptions of signals, a description of the control interface, and a functional description.

26.2.1 Introduction

This section includes an overview, description of features, and a description of the modes of operation.

26.2.1.1 Overview

The Unified DMA (uDMA) block is connected with the ENET-MAC and through the 3PSWITCH. This is NOT a general purpose DMA but is highly specific to support Ethernet traffic.

Fundamentally, there are two basic modes of operation. The uDMA supports a legacy mode which enables data to be transferred in a format significantly consistent with the legacy FEC device that exists on several parts through the Legacy Buffer Descriptor (LBD). Additionally the uDMA supports an Enhanced Buffer Descriptor (EBD) which in turn provides the ability to make use of several new features including (but not limited to) IEEE-1588 support, Gigabit Ethernet, transmit error checking and various off loading features.

The uDMA blocks are highlighted in Figure 26-2.

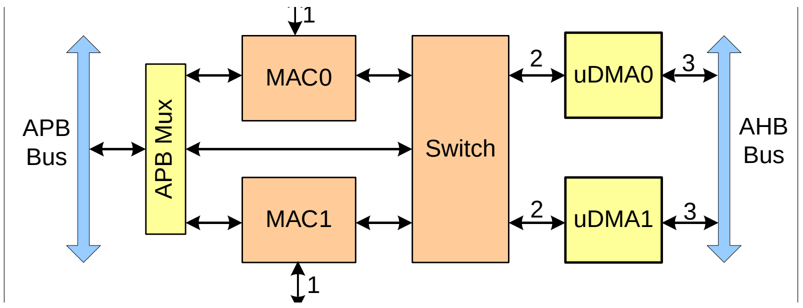


Figure 26-2. Typical uDMA SoC Connection

In reference to the figure:

1 - The interface between the MAC-NET and the SoC Pad Logic. All pin multiplexing takes place outside the ENET-MAC, including RMII and MII.

2 - The ENET-MAC communicates with the Unified DMA through two FIFO interfaces. One of the FIFO interfaces is for RX the other is for TX. These interfaces also contain a set of sideband signals.

3 - The Unified DMA communicates with the SoC Platform through this interface.

26.2.1.2 Features

The primary features of the uDMA are listed below:

- Only Surport big-endian.
- Separate TX and RX interfaces to allow for minimum space or maximum data throughput.

- Supports Legacy Buffer Descriptor programming models and functionality.
- Enables an Enhanced Buffer Descriptor programming model to support new Ethernet functionality.

26.2.1.3 Modes of Operation

The primary modes of operation of the uDMA are described in this section:

- Legacy mode
- Enhanced mode

26.2.1.3.1 Legacy Mode

While in legacy mode, the uDMA reads and generates Legacy Buffer Descriptors (LBD). Additionally, great effort is made to ensure that support signals such as interrupts behave consistently with the legacy programming model used with the FEC. This is the default mode of operation.

26.2.1.3.2 Enhanced Mode

While in enhanced mode, the uDMA reads and generates Enhanced Buffer Descriptors (EBD). The descriptors are based on the LBD but allow for the support of additional features such as IEEE-1588 support, additional interrupts and error checking among other features. While in this mode, there is no hard requirement that the programming model is consistent with the legacy programming model; however where reasonable the same look and feel is accomplished.

26.2.2 Functional Description

This section provides the detailed functional description of the uDMA. The uDMA is a data mover between the FIFO interface and the bus master and vice-versa. Additionally, the uDMA is responsible for formatting the data being transferred into and out of buffer descriptors. Further, the uDMA is responsible for the generation of various control signals (that is, interrupts) that may occur during the process. The uDMA is the key link for enabling the legacy programming format of the FEC module to work with the ENET-MAC.

This section provides the technical details for the uDMA. Additional details required for legacy mode compatibility is provided in the FEC specification (FEC_BlockGuide.pdf).

There are two FIFO interfaces, one for transmit and one for receive. Please refer to the documentation in the reference section for the detailed specification for these FIFO interfaces.

26.2.2.1 Legacy Buffer Descriptor Models

Table 26-1 and Table 26-2 show the legacy buffer descriptor (LBD) models.

26.2.2.1.1 Legacy FEC Receive Buffer Descriptor

This section discusses the legacy FEC receive buffer descriptors.

Table 26-1. Legacy FEC Receive Buffer Descriptor

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	RX Data Buffer Pointer - A[31:16]															
Offset + 6	RX Data buffer Pointer - A[15:0]															

The following sections reference Table 26-1.

26.2.2.1.1.1 Bit-15 E

Empty. Written by the uDMA (=0) and user (=1). The functionality and operation of this bit does not change from the FEC. The uDMA clears this bit to indicate that the buffer descriptor is complete and ready for the driver to process.

26.2.2.1.1.2 Bit-14 RO1

Receive software ownership. This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.

26.2.2.1.1.3 Bit-13 W

Wrap. Written by user. The functionality and operation of this bit does not change from the FEC. It indicates the next buffer descriptor the uDMA is supposed to use. If 0 the next buffer descriptor is found in the consecutive location, and if 1 the next buffer descriptor is found at the location defined in ERDSR.

26.2.2.1.1.4 Bit-12 RO2

Receive software ownership.This field is reserved for use by software. This read/write bit will not be modified by hardware, nor will its value affect hardware.

26.2.2.1.1.5 Bit-11 L

Last in frame. Written by the uDMA. Setting this bit to '1' indicates that the buffer descriptor is the last buffer descriptor in the frame.

26.2.2.1.1.6 Bit-10 and Bit 9 —

These bits are reserved and must not be modified.

26.2.2.1.1.7 Bit-8 M

Miss. Written by the uDMA. This bit is set by the uDMA for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, the user can use the M-bit to quickly determine whether the frame was destined to this station. This bit is only valid if the L-bit is set and the PROM bit is set inside the ENET-MAC. If 0,the frame was received because of an address recognition hit. If 1, the frame was received because of promiscuous mode. This field is only valid if the BD is the last in the frame, that is, L-bit is set.

26.2.2.1.1.8 Bit-7 BC

Set if the DA is broadcast (FF-FF-FF-FF-FF-FF).

26.2.2.1.1.9 Bit-6 MC

Set if the DA is multicast and not BC.

26.2.2.1.1.10 Bit-5 LG

Rx frame length violation, written by the uDMA. A frame length greater than MAX_FL was recognized. This bit is valid only if the L-bit is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.

26.2.2.1.1.11 Bit-4 NO

Receive non-octet aligned frame, written by the uDMA. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or Phy error occurred. This bit is valid only if the L-bit is set. If this bit is set the CR bit will not be set.

26.2.2.1.1.12 Bit-3 —

Reserved.

26.2.2.1.1.13 Bit-2 CR

Rx CRC or Rx Frame error, written by the uDMA. This frame contains a Frame with Phy error or CRC error and is an integral number of octets in length. This bit is valid only if the L-bit is set.

26.2.2.1.1.14 Bit-1 OV

Overrun, written by the uDMA. A receive FIFO overrun occurred during frame reception. If this bit is set, the other status bits, M, LG, NO, SH, CR, and CL lose their normal meaning and will be zero. This bit is valid only if the L-bit is set.

26.2.2.1.1.15 Bit-0 TR

Will be set if the receive frame is truncated (frame length > TRUNC_FL). If the TR bit is set, the frame should be discarded and the other error bits should be ignored as they may be incorrect.

26.2.2.1.1.16 Data Length

Data length. Written by the uDMA. Data length is the number of octets written by the uDMA into this BD's data section. If L equals 0, the value written will be equal to EMRBR. If L equals 1, complete length of the frame including the CRC. It is written by the uDMA once as the BD is closed.

26.2.2.1.1.17 RX Data Buffer Pointer - A[31:16]

RX data buffer pointer, bits [31:16].

26.2.2.1.1.18 RX Data Buffer Pointer - A[15:0]

RX data buffer pointer, bits [15:0]. The receive buffer pointer, which always points to the first location of the associated data buffer, must always be evenly divisible by 16.

26.2.2.1.20 Legacy FEC Transmit Buffer Descriptor

This section discusses the legacy FEC transmit buffer descriptors.

Table 26-2. Legacy FEC Transmit Buffer Descriptor

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	R	TO1	W	TO2	L	TC	ABC	—	—	—	—	—	—	—	—	—
Offset + 2	Data length															
Offset + 4	TX Data Buffer Pointer A[31:16]															
Offset + 6	TX Data buffer Pointer A[15:0]															

The following sections reference Table 26-2.

26.2.2.1.20.1 Bit-15 R

Ready. Written by the uDMA (=0) and the user (=1). It is set by the user to indicate the buffer descriptor is ready for transmission. The uDMA clears this bit at the start of data transmission.

26.2.2.1.20.2 Bit-14 TO1

Transmit software ownership. This field is reserved for software use. This read/write bit will not be modified by hardware, nor will its value affect the hardware.

26.2.2.1.20.3 Bit-13 W

Wrap. Written by user. The functionality and operation of this bit does not change from the FEC.

- 0 The next buffer descriptor is found in the consecutive location.
- 1 The next buffer descriptor is found at the location defined in ETDSR.

26.2.2.1.20.4 Bit-12 TO2

Transmit software ownership. This field is reserved for software use. This read/write bit will not be modified by hardware, nor will its value affect hardware.

26.2.2.1.20.5 Bit-11 L

Last in frame. Written by the user. The functionality and operation of this bit does not change from the FEC.

- 0 The buffer descriptor is not the last in the transmit frame.
- 1 The buffer descriptor is the last in the transmit frame.

26.2.2.1.20.6 Bit-10 TC

Tx CRC. This bit is written by the user. If '0' this indicates that the frame is sent as-is and it is up to the software to provide a valid frame with CRC field. If '1' the MAC will calculate and append the CRC field to the frame.

26.2.2.1.20.7 Bit-9 ABC

Append bad CRC. This bit is not supported in the uDMA legacy or enhanced modes and is ignored.

26.2.2.1.20.8 Bit 8-0 —

These values are reserved and should not be modified.

26.2.2.1.20.9 Data Length

Datalength. Written by the user. The functionality and operation of this field does not change from the FEC.

26.2.2.1.20.10 TX Data Buffer Pointer A[31:16]

TX data buffer pointer. A[31:0] contains the address of the associated data buffer and must always be evenly divisible by 4. The functionality and operation of this field does not change from the FEC.

26.2.2.1.20.11 TX Data Buffer Pointer A[15:0]

TX data buffer pointer. The functionality and operation of this field does not change from the FEC.

26.2.2.31 Enhanced Buffer Descriptor Models

In order to support various new functionality now available in the ENET-MAC the uDMA also supports an enhanced buffer descriptor model (EBD). Table 26-3 and Table 26-4 shows what is being referred to as the EBD models. To promote the maximum reuse of driver software where possible the LBD fields were reused.

26.2.2.31.1 Enhanced uDMA Receive Buffer Descriptor

This section discusses the enhanced uDMA receive buffer descriptors.

Table 26-3. Enhanced uDMA Receive Buffer Descriptor

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 2	Data length															
Offset + 4	RX Data Buffer Pointer A[31:16]															
Offset + 6	RX Data buffer Pointer A[15:0]															
Offset + 8	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—
Offset + A	—	—	—	—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + C	Header Length					—	—	—	Protocol Type							
Offset + E	Payload Checksum															
Offset + 10	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 Timestamp[31:16]															
Offset + 16	1588 Timestamp[15:0															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

26.2.2.31.1.1 Offset + 0 Bit 15:0

This set of bits behaves exactly as in legacy mode with the exception of bit 15 E. See Legacy FEC Receive Buffer Descriptor. The Last Buffer Descriptor is already updated when bit E=0 and bit BDU=1.

26.2.2.31.1.2 Data Length

This field behaves exactly the same as in legacy mode. See Legacy FEC Receive Buffer Descriptor .

26.2.2.31.1.3 RX Data Buffer Pointer A[31:16]

This field behaves exactly the same as in legacy mode. See Legacy FEC Receive Buffer Descriptor.

26.2.2.31.1.4 RX Data Buffer Pointer A[15:0]

This field behaves exactly the same as in legacy mode. See Legacy FEC Receive Buffer Descriptor.

26.2.2.31.1.5 Offset + 8 Bit 15 ME

MAC error. This bit is written by the uDMA. This bit means that the frame stored in the system memory was received with an error. This bit is only valid when the L-bit is set.

26.2.2.31.1.6 Offset + 8 Bit 14 —

This bit is not used (reserved) since the uDMA aborts the operation when a DMA detects an error during the data transfer.

26.2.2.31.1.7 Offset + 8 Bit 10 PE

PHY Error. This bit is written by the uDMA. Set to 1 when the frame was received with an Error character on the PHY interface. The frame is invalid. This bit is valid only when the L-bit is set.

26.2.2.31.1.8 Offset + 8 – Bit 9 CE

Collision. This bit is written by the uDMA. Set to 1 when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This bit is valid only when the L-bit is set.

26.2.2.31.1.9 Offset + 8 – Bit 8 UC

Unicast. This bit is written by the uDMA. This bit means that the frame is unicast. This bit is valid regardless of if the L-bit is set.

26.2.2.31.1.10 Offset + 8 – Bit 7 INT

Generate RXB/RXF interrupt. This bit is set by the user. This bit indicates that the uDMA is to generate an interrupt on the dma_int_rxb / dma_int_rxf event.

26.2.2.31.1.11 Offset + A – Bit 5 ICE

IP header checksum error. This is an accelerator option. This bit is written by the uDMA. Set to 1 when either not an IP frame is received, or the IP header checksum was invalid. This bit is only valid if the L-bit is set.

26.2.2.31.1.12 Offset + A – Bit 4 PCR

Protocol checksum error. This is an accelerator option. This bit is written by the uDMA. Set to 1 when the checksum of the protocol is invalid, or an unknown protocol is found and checksumming could not be performed. This bit is only valid if the L-bit is set.

26.2.2.31.1.13 Offset + A – Bit 3 —

Type removed. This is an accelerator option. This bit is written by the uDMA. If set, this bit indicates that the frame's type feld has been removed: The frames payload starts immediately after the MAC source address within the frame, or after the VLAN tag if present. This bit is only valid if the L-bit is set. The date for this bit comes from the MTIP sideband signal ff_rx_ip_stat[7].

26.2.2.31.1.14 Offset + A – Bit 2 VLAN

VLAN. This is an accelerator option. This bit is written by the uDMA. This bit means that the frame has a VLAN tag. This bit is valid only if the L-bit is set.

26.2.2.31.1.15 Offset + A – Bit 1 IPV6

IPV6 Frame. This bit is written by the uDMA. This bit indicates that the frame has a IPv6 frame type. If this bit is not set, it means that an IPv4 or other protocol frame was received. This bit is valid only if the L-bit is set.

26.2.2.31.1.16 Offset + A – Bit 0 FRAG

Pv4 Fragment.This is an accelerator option.This bit is written by the uDMA.This bit indicates that the frame is an IPv4 fragment frame. This bit is only valid when the L-bit is set.

26.2.2.31.1.17 Offset + C – Bit [15:11] Header Length

Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32 bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers:

- ICMP/IP: 6 (5 IP header, 1 ICMP header)
- UDP/IP: 7 (5 IP header, 2 UDP header)
- TCP/IP: 10 (5 IP header, 5 TCP header)

This field is only valid if the L-bit is set.

26.2.2.31.1.18 Offset + C – Bit [7:0] Protocol Type

Protocol type. This is an accelerator option. The 8-Bit protocol field found within the IP header of the frame. Only valid if "ICE" bit is 0. This bit is only valid if the L-bit is set.

26.2.2.31.1.19 Offset + E – Bit [15:0] Payload Checksum

Internet payload checksum. This is an accelerator option. The one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L-bit is set.

26.2.2.31.1.20 Offset + 0x10 – Bit 15 BDU

Last Buffer Descriptor Update Done. This bit indicates that all the last BD data has been updated by uDMA. This bit is written by the user (=0) and uDMA (=1).

26.2.2.31.1.21 1588 Timestamp [31:0]

It is only valid if the L-bit is set.

26.2.2.31.23 Enhanced uDMA Transmit Buffer Descriptor

This section discusses the enhanced uDMA transmit buffer descriptors.

Table 26-4. Enhanced uDMA Transmit Buffer Descriptor

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	R	TO1	W	TO2	L	TC	-	-	-	-	-	-	-	-	-	-
Offset + 2	Data length															
Offset + 4	TX Data Buffer Pointer - A[31:16]															
Offset + 6	TX Data buffer Pointer - A[15:0]															
Offset + 8	-	INT	TS	PINS	IINS	-	-	-	-	-	-	-	-	-	-	-
Offset + A	TXE	-	UE	EE	FE	LCE	OE	TSE	-	-	-	-	-	-	-	-
Offset + C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + E	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 10	BDU	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 14	1588 Timestamp [31:16]															
Offset + 16	1588 Timestamp [15:0]															
Offset + 18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 1A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 1C	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Offset + 1E	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

The following sections reference Table 26-4.

26.2.2.31.23.1 Offset + 0 - Bit 15:0

This functionality does not change from the legacy programming model with the exception of bit 9 "ABC" which has been explicitly removed and bit 15 "R". See Legacy FEC Transmit Buffer Descriptor.

26.2.2.31.23.2 Offset + 2 - Data Length

This functionality does not change from the legacy programming model. See Legacy FEC Transmit Buffer Descriptor .

26.2.2.31.23.3 Offset + 4 and Offset + 6 - TX Data Buffer Pointer

This functionality does not change from the legacy programming model. See Legacy FEC Transmit Buffer Descriptor.

26.2.2.31.23.4 Offset + 8 Bit 15 "-"

Error Indication. This bit is written by the user. This bit is used by the application software to indicate that the frame stored in the system memory contains an error. This bit is valid regardless of if the L-bit is set and must be the same for all EBD for a given frame. The uDMA does not update this value.

26.2.2.31.23.5 Offset + 8 Bit 14 "INT"

Generate interrupt. This bit is written by the user. This indicates that the uDMA is to generate a dma_int_txb / dma_int_txf interrupt in relation to this frame / buffer descriptor. This bit is valid regardless of if the L-bit is set and must be the same for all EBD for a given frame. The uDMA does not update this value.

26.2.2.31.23.6 Offset + 8 Bit 13 "TS"

Timestamp. This bit is written by the user. This indicates that the uDMA is to generate a timestamp frame. This bit is valid regardless of if the L-bit is set and must be the same for all EBD for the given frame. The uDMA does not update this value.

26.2.2.31.23.7 Offset + 8 Bit 12 "PINS"

Insert protocol specifc checksum. This bit is written by the user. If '1' the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum feld with the calculated value. The checksum feld must be set to 0 by the application generating the frame. The uDMA does not update this value. This bit is valid regardless of if the L-bit is set and must be the same for all EBD for a given frame.

26.2.2.31.23.8 Offset + 8 Bit 11 "IINS"

Insert IP header checksum. This bit is written by the user. If '1' the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header feld with the calculated value. The checksum feld must be set to 0 by the application generating the frame. The uDMA does not update this value. This bit is valid regardless of if the L-bit is set and must be the same for all EBD for a given frame.

26.2.2.31.23.9 Offset + A Bit 15 "TXE"

Transmit error occurred. This bit is written by the uDMA. This bit indicates that there was a transmit error reported with the frame. Effectively this bit is the "or" of all other error bits including UE, EE, FE, LCE, OE, and TSE. This bit is only valid when the L-bit is set.

26.2.2.31.23.10 Offset + A Bit 14 "-"

This bit is not used (reserved) since the uDMA will abort the operation when a DMA detects an error during the data transfer.

26.2.2.31.23.11 Offset + A Bit 13 "UE"

Underflow error. This bit is written by the uDMA. This bit indicates that the MAC reported an underfow error on transmit.

26.2.2.31.23.12 Offset + A Bit 12 "EE"

Excess Collision error. This bit is written by the uDMA. This bit indicates that the MAC reported an excess collision error on transmit. This bit is only valid when the L-bit is set.

26.2.2.31.23.13 Offset + A Bit 11 "FE"

Frame with error. This bit is written by the uDMA. This bit indicates that the MAC reported that the uDMA reported an error when providing the packet. This bit is only valid when the L-bit is set.

26.2.2.31.23.14 Offset + A Bit 10 "LCE"

Late collision error. This bit is written by the uDMA. This bit indicates that the MAC reported that there was a Late Collision on transmit. This bit is only valid when the L-bit is set.

26.2.2.31.23.15 Offset + A Bit 9 "OE"

Overfow error. This bit is written by the uDMA. This bit indicates that the MAC reported that there was a FIFO overfow condition on transmit. This bit is only valid when the L-bit is set.

26.2.2.31.23.16 Offset + A Bit 8 "TSE"

Timestamp error. This bit is written by the uDMA. This bit indicates that the MAC reported a different frame type then a timestamp frame. This bit is only valid when the L-bit is set.

26.2.2.31.23.17 Offset + 0x10 - Bit 15 "BDU"

Last Buffer Descriptor Update Done. This bit indicates that all the last BD data has been updated by uDMA. This bit is written by the user (=0) and uDMA (=1).

26.2.2.31.23.18 1588 Timestamp [31:0]

This value is written by the uDMA when switch is in Bypass mode. It is only valid if the L-bit is set.

26.3 Ethernet ENET-MAC Core

This section describes the ENET-MAC core, including the block diagram and formats.

26.3.1 Introduction

Ethernet is available in different speeds (10/100 Mbps) and provides connectivity to meet a wide range of needs and from desktop to switches.

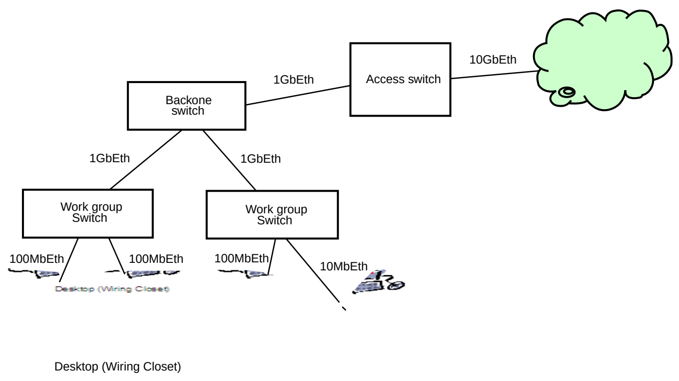


Figure 26-3. Enterprise LAN Topology Example

The ENET-MAC Core implements, in conjunction with a dual-speed 10/100 MAC, Layer 3 network acceleration functions, which are designed to accelerate the processing of various common networking protocols such as IP, TCP, UDP and ICMP providing wire speed services to Client applications.

The Core implements a dual speed 10/100 Mbps Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with Half or Full Duplex 10/100 Mbps Ethernet and Fast Ethernet LANs.

The MAC operation is fully programmable and can be used in NIC (Network Interface Card), bridging or switching applications. The Core implements the Remote Network Monitoring (RMON) counters according to IETF RFC 2819. All registers are accessible through a 32-Bit APB interface.

The Core also implements a Hardware acceleration block to optimize the performance of network controllers providing IP and TCP, UDP, ICMP protocol services. The acceleration block performs, in hardware, critical functions, which are typically implemented with large software overhead.

The Core implements programmable embedded FIFOs that can provide, on the Receive path, buffering for loss-less flow control.

Advanced Power Management features are available with Magic Packet detection and programmable power down modes.

For industrial automation application, the IEEE 1588 standard is becoming the main technology for precise time synchronization on Ethernet networks providing accurate clock synchronization for distributed control nodes to overcome one of the drawbacks of Ethernet.

The programmable 10/100 Ethernet MAC with IEEE 1588 support integrates a standard IEEE 802.3 Ethernet MAC with a time stamping module to support Ethernet applications requiring precise timing references for incoming and outgoing frames to implement a distributed time synchronization protocol such as the IEEE 1588.

26.3.2 10 100Mbps Ethernet ENET-MAC Core Features

26.3.2.1 Ethernet MAC Features

- Implements the full 802.3 specification with preamble / SFD generation, frame padding generation, CRC generation and checking
- Dynamically configurable to support 10 Mbps and 100 Mbps operation
- Supports full duplex and configurable half duplex operation
- Supports AMD Magic Packet detection with interrupt for node remote power management
- Seamless interface to commercial Fast Ethernet PHY device through a 4-Bit Media Independent Interface (MII) operating at 25MHz
- Simple 64-Bit FIFO interface to user application
- CRC-32 checking at full speed with optional forwarding of the FCS field to client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- When operating in Full Duplex mode, implements automated Pause Frame (802.3 x31A) generation and termination providing flow control without user application intervention
- When operating in full duplex mode, pause quanta used to form Pause frames, dynamically programmable
- Pause frame generation additionally controllable by user application offering flexible traffic flow control

- Optional forwarding of received pause frames to the user application when operating in Full Duplex mode
- Implements standard flow-control mechanism in full-duplex operation mode
- In half-duplex mode, provides full collision support, including jamming, backoff, and automatic retransmission
- Support for VLAN tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable group of four supplemental MAC addresses that can be used to filter Unicast traffic
- Programmable Promiscuous mode support to omit MAC destination address checking on receive
- Multicast and Unicast address filtering on receive based on 64 entries hash table reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic as well as errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory Management Information Database (MIB) package and Remote Network Monitoring (RFC 2819)
- Simple handshake user application FIFO interface with fully programmable depth and threshold levels ensuring data rates of 1Gbps
- 64-Bit Client FIFO interface
- Separate status word available for each received frame on the user interface providing information such as frame length, frame type, VLAN tag and error information
- Multiple internal loopback options
- MDIO Master interface for PHY device configuration and management with two programmable MDIO base addresses

26.3.2.2 IP Protocol Performance Optimization Features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing

- IPv4 and IPv6 support
- Transparent passing of frames of other types and protocols
- Support for VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and automatic insertion on transmit configurable on a per-frame basis
- Support for IP and TCP, UDP, ICMP data for checksum generation and checking
- Full header options support for IPv4 and TCP protocol headers
- IPv6 support limited to datagrams with base header only. Datagrams with extension headers are passed transparently unmodifed/unchecked
- Statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic Host-to-Network (RX) and Network-to-Host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive
- Configurable Ethernet Payload alignment to allow for 32-bit word aligned header and payload processing
- Programmable Store & Forward operation with clock and rate decoupling FIFOs

26.3.2.3 IEEE 1588 Functions

- Support for all IEEE 1588 Frames
- Reference Clock can be chosen independently of the Network speed
- Software Programmable Precise Time-Stamping of Ingress Frames and Egress Frames
- Timer monitoring capabilities for System calibration and timing accuracy management
- Precise time stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Hardware and Software controllable timer synchronization

26.3.3 ENET-MAC Core Block Diagram

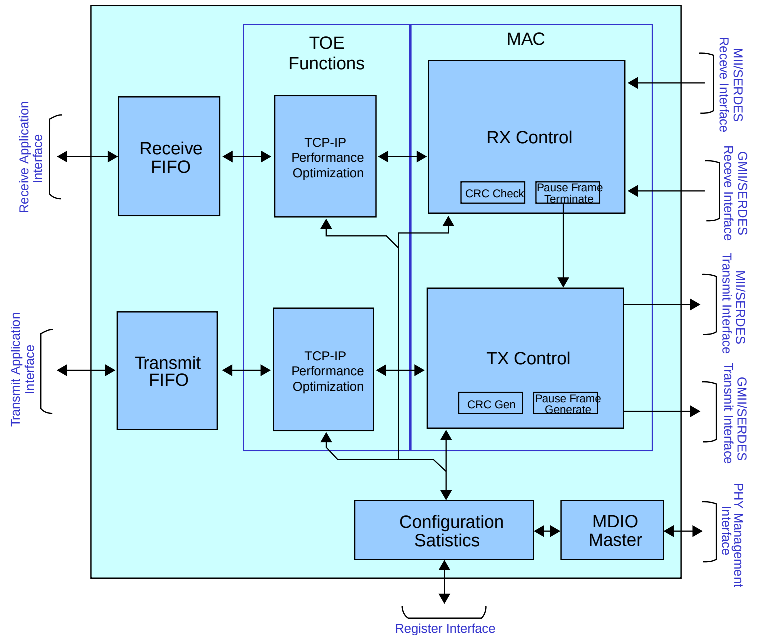

TOE Functions
MAC
Receive Application Interface
Receive FIFO
TCP-IP Performance Optimization
RX Control
CRC Check
Pause Frame Terminate
MII/SERDES Receive Interface
GMII/SERDES Receive Interface
Transmit Application Interface
Transmit FIFO
TCP-IP Performance Optimization
TX Control
CRC Gen
Pause Frame Generate
MII/SERDES Transmit Interface
GMII/SERDES Transmit Interface
Configuration Satistics
MDIO Master
PHY Management Interface
Register Interface

Figure 26-4. ENET-MAC Core Overview

26.3.4 Ethernet MAC Frame Formats

26.3.4.1 Overview

The IEEE 802.3 Standard defines the Ethernet frame format as follows: An Ethernet frame has a minimum length of 64 bytes and a maximum length of 1518 bytes or more if jumbo frames are supported, excluding the preamble and the start frame delimiter bytes. An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields
- Length or type field
- Data field
- Frame check sequence (CRC value)
- An EXTENSION field is defined only for gigabit Ethernet half-duplex implementations and is not supported by the MAC core

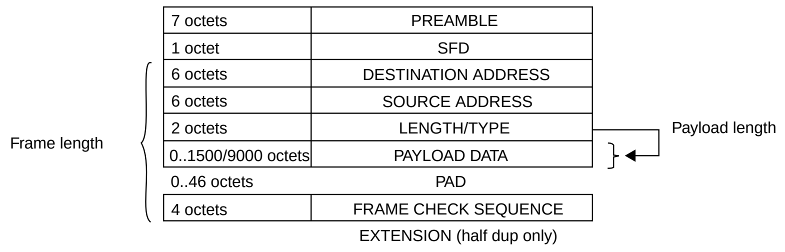


Figure 26-5. MAC Frame Format Overview

Optionally MAC frames can be VLAN-tagged with an additional 4-byte field (VLAN Tag and VLAN Info) inserted between the MAC source address and the Type/Length Field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

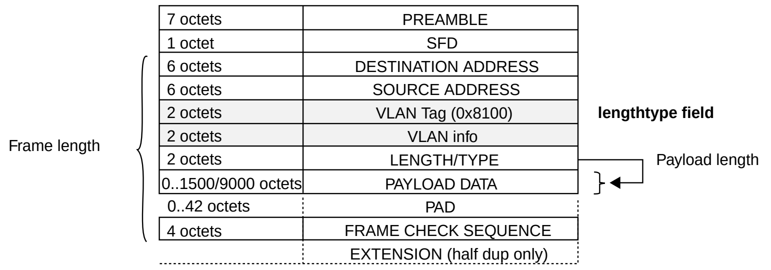


Figure 26-6. VLAN Tagged MAC Frame Format Overview

Table 26-5. MAC Frame Definition

Term	Description
Frame length	The length, in octets, defines the length of the complete Frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length (typical 1518).
Payload Length	The length / type field indicates the length of the frame's payload section. The most significant byte is sent/received first. If the Length/type field is set to a value lower than 46, the payload is padded so that the minimum frame length requirement (64 Bytes) is met. For VLAN tagged frames, a value less than 42 indicates a padded frame. If the Length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and Source Address	48-bit MAC addresses. The least significant byte is sent/received first and the two first bits (two least significant bits) of the MAC address are used to distinguish MAC frames.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

26.3.4.2 Pause Frames

A Pause Frame is generated by the receiving device to indicate a congestion to the emitting device which should stop sending data.

Pause Frames are indicated by the Length/Type set to 0x8808. The two first bytes of a Pause Frame following the type, defines a 16-Bit opcode field set to 0x0001 always. A 16-Bit Pause Quanta is defined in the Frame payload Bytes 2 (Byte P1) and 3 (Byte P2) as defined in Table 26-6. The pause quanta byte P1 is the most significant.

Table 26-6. Pause Frame Format (values in hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27-68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no Payload Length field found within a Pause Frame and a Pause Frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater zero (XOFF Condition) is received, the MAC stops transmitting data as soon the current Frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON Condition) is received, the transmitter is allowed to send data immediately (See chapter 10 for details).

26.3.4.3 Magic Packets

A Magic Packet can be a Unicast, Multicast or Broadcast packet, which carries a defined sequence in the payload section. Magic Packet are received and inspected only under specific conditions as described in Magic Packet Detection

The defined sequence used to decode a Magic Packet is formed with a synchronization stream (Six consecutive 0xFF bytes) followed by a sequence of six consecutive Unicast MAC addresses (The Unicast Address of the Node to be awakened).

The sequence can be located anywhere in the Magic Packet payload and the Magic Packet is formed with standard Ethernet header optional padding and CRC.

26.3.5 IP and Higher Layers Frame Format

26.3.5.1 Definitions

The following chapters use the term datagram to describe the protocol specific data unit, which is found within the payload section of its container entity.

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

26.3.5.2 Ethernet Types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field is used to discriminate several datagram types. The following table lists the types of interest:

Table 26-7. Ethernet Type Value Examples

Type	Description
0x8100	VLAN tagged frame. The actual type is found 4 octets later in the frame
0x0800	IP
0x0806	ARP
0x86dd	IPv6

26.3.5.3 IPv4 Datagram Format

The following Figure 6 shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The most significant bit is bit 31 in the following figure. The first byte sent/received is the leftmost byte of the first word (that is, Version/IHL field).

The IP Header can contain further options, which are always padded if necessary, to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is followed by the payload immediately, which can contain further protocol headers like for example, TCP or UDP as indicated by the protocol field value. The complete IP datagram is transported in the payload section of an Ethernet frame.

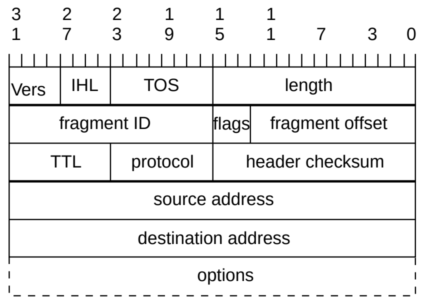


Figure 26-7. IPv4 Header Format

Table 26-8. IPv4 Header Fields

Field Name	Description
Vers	4-bit IP version information. It is 4 for IPv4 frames
IHL	4-bit internet header length information. Determines number of 32-bit words found within the IP header. The default value, if no options are present is 5
TOS	Type of Service / DiffServ field
length	Total length of the datagram in bytes. It includes all octets of header and payload
fragment ID, flags, fragment offset	Fields used for IP fragmentation
TTL	Time-to-live. If zero, datagram must be discarded
protocol	Protocol Identifier of protocol that follows in the datagram
header checksum	Checksum over all IP header fields
source address	Source IP address
destination address	Destination IP address

26.3.5.4 IPv6 Datagram Format

Figure 26-8 shows the IP Version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of 10 words (40 byte). The next header field identifies the type of the header to follow the IPv6 header. It is defined identical to the protocol identifier within IPv4 with new definitions for identifying so-called extension headers, which can be inserted between the IPv6 header and the protocol header, shifting the protocol header accordingly. The accelerator currently only supports IPv6 without extension headers (that is, next header identifies TCP or UDP or ICMP protocol).

The most significant bit is bit 31 in the following figure. The first byte sent/received is the leftmost byte of the first word (that is, Version/Traffic class fields).

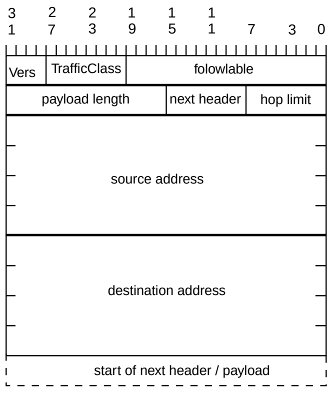


Figure 26-8. IPv6 Header Format

Table 26-9. IPv6 Header Fields

Field Name	Description
Vers	4-bit IP version information. It is 6 for all IPv6 frames
Traffic Class	8-bit field defining the traffic class
flow label	20-bit flow label identifying frames of the same flow
payload length	16-Bit Length of the datagram payload (!) in bytes. It includes all octets following the IPv6 header.
next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.
hop limit	Hop counter, decremented by 1 by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
source address	128-bit IPv6 source address
destination address	128-bit IPv6 destination address

26.3.5.5 ICMP Datagram Format

Following the IP Header, an Internet Control Message Protocol (ICMP) datagram is found when the protocol identifier has a decimal value of 1. The ICMP datagram has 4 octets header followed by the additional message data.

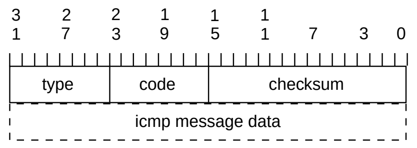


Figure 26-9. ICMP Header Format

Table 26-10. IP Header Fields

Field Name	Description
type	8-bit type information
code	8-bit code that is related to the message type
checksum	16-bit one's complement checksum over the complete ICMP datagram

26.3.5.6 UDP Datagram Format

Following the IP Header, a user datagram protocol header is found when the protocol identifier has a decimal value of 17.

Following the UDP header is the payload of the datagram. The header byte order follows the conventions given for the IP header above.

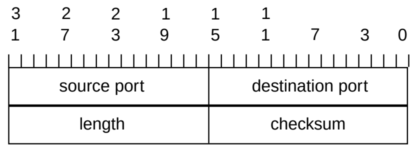


Figure 26-10. UDP Header Format

Table 26-11. UDP Header Fields

Field Name	Description
source port	Source application port
destination port	Destination application port
length	Length of user data which follows immediately the header including the UDP header. That is, the minimum value is 8.
checksum	Checksum over the complete datagram and some IP header information

26.3.5.7 TCP Datagram Format

Following the IP Header, a TCP header is found when the protocol identifier has a decimal value of 6.

The TCP payload follows immediately the TCP header.

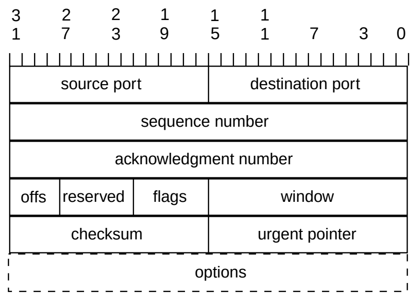


Figure 26-11. TCP Header Format

Table 26-12. TCP Header Fields

Field Name	Description
source port	Source application port
destination port	Destination application port
sequence number	Transmit sequence number
ack. number	Receive sequence number
offs	Data offset. Number of 32-bit words within the TCP header. If no options, a value of 5.
flags	URG, ACK, PSH, RST, SYN, FIN flags
window	TCP receive window size information
checksum	Checksum over the complete datagram (TCP Header and data) and IP header information
options	Additional 32-bit words for protocol options

26.3.6 IEEE 1588 Message Formats

26.3.6.1 Transport Encapsulation

The Precision-Time-Protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or, optionally, with the newer 1588v2 directly in Ethernet frames (Layer 2). Typically multicast addresses are used to allow efficient distribution of the synchronization messages.

26.3.6.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages. The following IP multicast groups are defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last 3 octets of IP follow the fixed value of 01-00-5e).

Table 26-13. UDP/IP Multicast Domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5e-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5e-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5e-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5e-00-01-84

Table 26-14. UDP Portnumbers

Message Type	UDP Port	Note
event	319	Used for SYNC and DELAY_REQUEST messages
general	320	All other messages (for example, follow-up, delay-response)

26.3.6.1.2 Native Ethernet (PTPv2)

In addition to the usage of UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype=0x88f7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Beside others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are additionally defined in PTPv2.

Table 26-15. PTPv2 Multicast Domains

Name	MAC Address
Normal messages	01-1b-19-00-00-00
peer delay messages	01-80-c2-00-00-0e

26.3.6.2 PTP Header

All PTP frames contain a common header, which is used to determine the protocol version as well as the type of message, which defines the further content of the message.

All multi-octet fields are transmitted in big-endian order meaning, the most significant byte is transmitted/received first.

The version field's (versionPTP) last four bits are at the same position (that is, 2nd byte) for both PTPv1 and PTPv2 headers, allowing a correct identification by inspecting the first two bytes of the message.

26.3.6.2.1 PTPv1 Header

Table 26-16. Common PTPv1 message header

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
versionPTP = 0x0001								2	0
versionNetwork								2	2
subdomain								16	4
messageType								1	20
sourceCommunicationTechnology								1	21
sourceUuid								6	22
sourcePortId								2	28
sequenceId								2	30
control								1	32
0x00								1	33
flags								2	34
reserved								4	36

The type of message is encoded in the fields messageType and control as follows:

Table 26-17. PTPv1 Message Type Identification

messageType	control	Message Name	Message
0x01	0	SYNC	event message
0x01	1	DELAY_REQ	event message
0x02	2	FOLLOW_UP	general message
0x02	3	DELAY_RESP	general message
0x02	4	MANAGEMENT	general message

messageType	control	Message Name	Message
other	other		reserved

The field sequenceId is used to non-ambiguously identify a message.

26.3.6.2.2 PTPv2 Header

Table 26-18. Common PTPv2 message Header

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageId				1	0
reserved				versionPTP = 0x2				1	1
messageLength								2	2
domainNumber								1	4
reserved								1	5
flags								2	6
correctionField								8	8
reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
control								1	32
logMeanMessageInterval								1	33

The type of message is encoded in the field messageId which is as follows:

Table 26-19. PTPv2 Message Type Identification

messageId	Message Name	Message
0x0	SYNC	event message
0x1	DELAY_REQ	event message
0x2	PATH_DELAY_REQ	event message
0x3	PATH_DELAY_RESP	event message
0x4 - 0x7		reserved
0x8	FOLLOW_UP	general message

messageId	Message Name	Message
0x9	DELAY_RESP	general message
0xa	PATH_DELAY_FOLLOW_UP	general message
0xb	ANNOUNCE	general message
0xc	SIGNALING	general message
0xd	MANAGEMENT	general message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The flags field consists of two octets with the following meanings for the bits. Reserved bits are set to 0 (false).

Table 26-20. PTPv2 Message Flags Field Definitions

Octet Offset	bit	Name	Description
6 (first)	0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
	1	TWO_STEP	True (1) for two-step clock False (0) for one-step clock
	2	UNICAST	True (1) if transport layer address uses a unicast destination address, false (0) if multicast is used.
	3	reserved	
	4	reserved	
	5	profile specific	
	6	profile specific	
	7	reserved	

26.3.7 MAC Receive

26.3.7.1 Overview

The MAC receive engine performs the following tasks:

- Check Frame Framing
- Remove Frame preamble and Frame SFD field
- Frame Discarding based on Frame Destination address field

- Terminate Pause Frames
- Check Frame Length
- Remove payload padding if it exists
- Calculate and verify CRC-32
- Write received Frames in the Core receive FIFO

In the MAC is programmed to operate in half duplex mode, the MAC performed the following additional action:

- Check if the frame is received with a collision

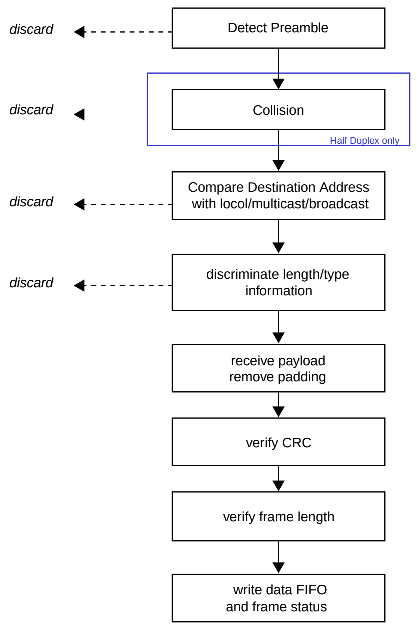

discard
Detect Preamble
discard
Collision
Half Duplex only
discard
Compare Destination Address with locol/multicast/broadcast
discard
discriminate length/type information
receive payload remove padding
verify CRC
verify frame length
write data FIFO and frame status

Figure 26-12. MAC Receive Flow

26.3.7.2 Collision Detection in Half Duplex Mode

If the packet is received with a collision detected during the reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and status bit ff_rx_err_stat(24) set to '1'.

26.3.7.3 Preamble Processing

The IEEE 802.3 Standard allows a maximum size of 56 bits (7 bytes) for the preamble, while the MAC Core allows any arbitrary preamble length. The MAC Core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame will be discarded.

Although the IEEE specification specifies that Frames should be separated at least by 96 bit (Inter Packet Gap or IPG), the MAC Core is designed to accept frames only separated by 64 MII (10/100Mbps operation) bits.

The MAC Core removes all the preamble bytes and the SFD byte.

26.3.7.4.1 Overview

The destination address bit 0 is used to differentiate Multicast and Unicast Addresses:

- If bit 0 is set '0' the MAC address is an individual (Unicast) address.
- If bit 0 is set '1' the MAC address defines a group address (Multicast Address).
- If all 48 bits of the MAC address are set to '1' it indicates a Broadcast address.

26.3.7.4.2 Unicast Address Check

If a Unicast address is received, the destination MAC address is compared to the Node MAC address programmed by the host in the registers PADDR1/PADDR2. In addition, it is compared to the supplemental MAC addresses programmed in the registers SMAC_0_0/SMAC_0_1, SMAC_1_0/SMAC_1_1, SMAC_2_0/SMAC_2_1 and SMAC_3_0/SMAC_3_1. If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If only one MAC address is required, all the supplemental MAC addresses should be programmed with the Node MAC address.

If promiscuous mode is enabled (configuration register RCR(PROM) set to '1') no address checking is performed and all Unicast frames are accepted.

26.3.7.4.3 Multicast and Unicast Address Resolution

The hash table algorithm used in the group and individual hash filtering operates as follows. The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits stored in GAUR, GALR (group address hash match), or IAUR, IALR (individual address hash match). This mapping is performed by passing the 48-bit address through the on-chip 32 bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63. The msb of the CRC result selects GAUR (msb equals 1) or GALR (msb equals 0). The least significant five bits of the hash result select the bit within the selected register. If the CRC generator selects a bit set in the hash table, the frame is accepted; otherwise, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- FCS(X) = X 32 +X 26 +X 23 +X 22 +X 16 +X 12 +X 11 +X 10 +X 8 +X 7 +X 5 +X 4 +X 2 +X 1 +1

If promiscuous mode is enabled (Configuration register bit PROM set to '1'), all Unicast and all Multicast frames are accepted regardless of GAUR / GALR and IAUR / IALR settings.

26.3.7.4.4 Broadcast Address Reject

All Broadcast Frames are accepted if the register bit BC_REJ is set to '0' or if the register bit PROM is set to '1'. If PROM is set to '0' when BC_REJ set to '1', all Broadcast Frames are rejected.

Table 26-21. Broadcast Address Reject Programming

PROM	BC_REJ	Broadcast Frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

26.3.7.4.5 Miss-Bit Implementation

For higher layer filtering purposes, the receive FIFO interface provides a status bit (ff_rx_err_stat[26]) indicating an address miss when the MAC operates in promiscuous mode and accepted a frame that would otherwise be rejected.

If neither a hash match (group or individual) nor an exact match (group or individual) occur, and if promiscuous mode is enabled (RCR[PROM] equals 1), the frame is accepted and the MISS bit is set; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during promiscuous mode:

- BC_REJ=1. A broadcast frame is received in promiscuous mode while broadcast reject is set.
- A unicast is received that does not match with any of the following:
    - PADDR1/ 2 (Node address, PALR/PAUR)
    - SMAC_0/1/2/3 (supplemental unicast addresses)
    - IADDR1/2 hash table entries (hash table for unicast, IALR/IAUR)
- A multicast is received that does not match the GADDR1/2 hash table entries (GALR/IAUR)

26.3.7.5.1 Payload Length Check

If the Length / Type has a value lower than 0x600 and if the register bit NO_LGTH_CHECK is configured to perform the check, the MAC checks the payload length and reports any error in the frame status word (Bit 1 of Core signal ff_rx_err_stat(0) and interrupt bit PLR).

If the Length / Type field has a value greater or equal 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames.

If a padded frame is received, no length check can be performed due to the extended frame payload (that is, padded frames never can have a payload length error).

26.3.7.5.2 Frame Length Check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt bit is generated. The LG register bit and ff_rx_err_stat(1) are set to '1'.

The frame is not truncated unless the frame length exceeds the value programmed in TRUNC_FL. If the frame is truncated, ff_rx_err_stat(2) is set to '1'. In addition, a truncated frame will always have the CRC error indication set (ff_rx_err_stat(3)).

26.3.7.6 VLAN Frames Processing

VLAN frames have a Length / Type field set to 0x8100 immediately followed by a 16-Bit VLAN Control Information field. VLAN tagged frames are received as normal frames (VLAN Tag not interpreted by the MAC function) and are completely (Including the VLAN tag) pushed to the user application. If the length/type field of the VLAN tagged frame, which is found four octets later in the frame is less than 42, padding will be removed. In addition, the frame status word (Core signal ff_rx_err_stat(7)) indicates that the current frame is VLAN tagged.

26.3.7.7 Pause Frame Termination

Pause frames are terminated within the receive engine and not transferred to the receive FIFO. The Quanta is extracted and sent to the MAC Transmit path through a small internal Clock Rate decoupling asynchronous FIFO.

The Quanta is written only if a correct CRC and a correct frame length are detected by the control state machine. If not, the Quanta is discarded and the MAC Transmit path is not paused.

Good Pause Frames are ignored if the register bit FCE is set to '0' and are forwarded to the Client interface when register bit PAUSE_FWD is set to '1'.

26.3.7.8 CRC Check

The CRC-32 field is checked and is forwarded to the Core FIFO interface if the Core configuration registers CRC_FWD and PAD_EN are set to '0' and '1' respectively. When the Core register CRC_FWD is set to '1' (Regardless of PAD_EN register value), the CRC-32 field is checked and terminated (Not transmitted to the Core FIFO).

The CRC polynomial, as specified in the 802.3 Standard, is as follows:

- FCS(X) = X 32 +X 26 +X 23 +X 22 +X 16 +X 12 +X 11 +X 10 +X 8 +X 7 +X 5 +X 4 +X 2 +X 1 +1

The 32 bits of the CRC value are placed in the FCS field so that the X31 term is the right-most bit of the first octet. The CRC bits are thus received in the following order: X31, X30,..., X1, X0.

If a CRC error is detected, the frame is marked invalid and ff_rx_err_stat(3) is set °Æ1°Ø.

26.3.7.9 Frame Padding Remove

When a frame is received with a Payload length field set to lower than 46 (42 for VLAN tagged and 38 for frames with stacked VLANs) the '0' padding can be optionally removed before the Frame is written into the data FIFO as configured by the configuration bit PAD_EN.

Note

If a frame is received with excess padding (that is, length field as mentioned above, but frame has more than 64 octets) and padding removal is enabled, the padding is removed as normal and no error is caused if the frame is otherwise correct (for example, good CRC and less maximum length and no other error).

26.3.8 MAC Transmit

26.3.8.1 Overview

Frame transmission starts when the Transmit FIFO holds enough data. Once a transfer has started, the MAC transmit function performs the following tasks:

- Generate Preamble and SFD field before Frame transmission.
- Generate XOFF pause frames if the Receive FIFO reports a congestion or if register TFC_PAUSE signal is asserted with PAUSE_DUR set to a non zero value.
- Generate XON pause frames if the Receive FIFO congestion condition is cleared or if register TFC_PAUSE signal is asserted with PAUSE_DUR set to 0.
- Suspend Ethernet Frame transfer (XOFF) if a non zero Pause Quanta is received from the MAC receive path.
- Add padding to the frame if required.
- Calculate and append CRC-32 to the transmitted frame.
- Send Frame with correct Inter Packet Gap (IPG) (Deferring).

When the MAC is configured to operate in Half Duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer has expired

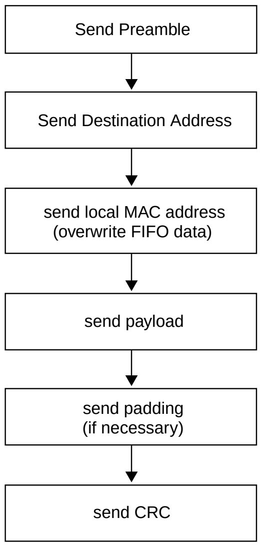


Figure 26-13. Frame Transmit Overview

26.3.8.2 Frame Payload Padding

The IEEE specification defines a minimum frame length of 64 bytes. If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) so that frames transmitted to the Ethernet link do not violate the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (ff_tx_crc_fwd=1) the user application is responsible for providing frames with a minimum length of 64 octets.

26.3.8.3 MAC Address Insertion

On each frame received from the Core transmit FIFO interface, the source MAC address is optionally replaced by the address programmed on the configuration registers PADDR1 and PADDR2 (Core Configuration register TX_ADDR_INS set to '1') or is transparently forwarded to the Ethernet line (Core Configuration register TX_ADDR_INS set to '0').

26.3.8.4 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a Frame if the Frame is transmitted to the Core with the Frame status bit ff_tx_crc_fwd set to '0' when ff_tx_eop is asserted.

The CRC polynomial, as specified in the 802.3 Standard, is as follows:

- FCS(X) = X 32 +X 26 +X 23 +X 22 +X 16 +X 12 +X 11 +X 10 +X 8 +X 7 +X 5 +X 4 +X 2 +X 1 +1

The 32 bits of the CRC value are placed in the FCS field so that the X31 term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: X31, X30,..., X1, X0.

26.3.8.5 Inter Packet Gap

In full duplex mode, after frame transmission and before transmission of a new frame, an Inter Packet Gap (IPG), programmed with register TX_IPG_LENGTH, is maintained. The minimum IPG can be programmed to any value between 8 and 27 byte-times (64 and 216 bit-times).

In half duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period and any backoff time requirements have been satisfied. In accordance with the standard, the Core begins to measure the IPG from mii_crs / gmii_crs de-assertion.

26.3.8.6 Collision Detection and Handling - Half Duplex Operation Only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the Core monitors the line condition and detects a collision when the PHY device asserts the MII mii_col signal.

When the Core detects a collision while transmitting, it stops the transmission of data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before the transmission of 64 bytes (Including preamble and SFD), the MAC Core waits for the backoff period and retransmits the packet data (Stored in a 64-Bytes re-transmit buffer) already sent on the line. The backoff period is generated from a pseudo random process (Truncated binary exponential backoff).

If a collision occurs after the transmission of 64 bytes (Including preamble and SFD), the MAC discards the remaining of the Frame, optionally sets the interrupt bit LC and sets the transmit status bit tx_ts_stat(1).

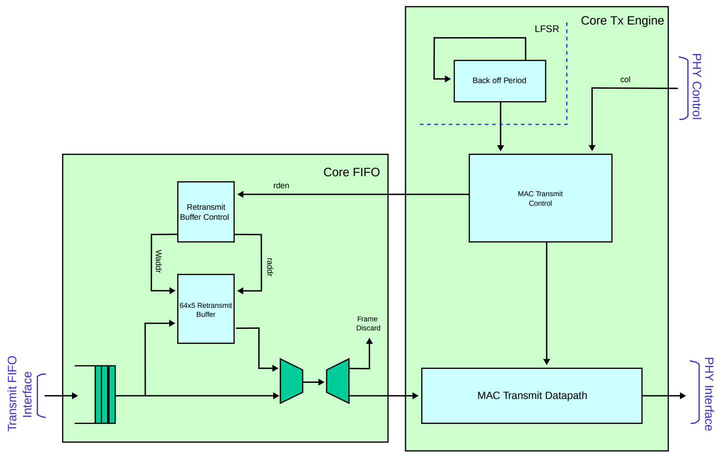


Figure 26-14. Packet Re-Transmit Overview

The backoff time is represented by an integer multiple of slot times (1 slot is equal to a 512-bit time period). The number of delay slot times, before the nth retransmission attempt, is chosen as a uniformly distributed random integer in the range:

- 0 < r < 2^k
- k = min(n, N) where n is the number of retransmissions and N = 10

For example, after the first collision, the backoff period (in slot time) is 0 or 1, if a collision occurs on the 1st retransmission, the backoff period (in slot time) is 0, 1, 2 or 3 and so on.

The maximum backoff time (in 512-Bit time slots) is limited by N set to 10 as specified in the IEEE 802.3 Standard.

If a collision occurs after 16 consecutive retransmissions, the Core reports an excessive collision condition (Interrupt bit RL and transmit status bit tx_ts_stat(2)) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after the transmission of first 64 bytes. In this case, the Core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The Core resumes transmission with the next packet available in the Core transmit FIFO.

26.3.9 Full Duplex Flow Control Operation

26.3.9.1 Overview

Three conditions are handled by the Core's Flow Control engine:

- Remote Device Congestion: The Remote device connected to the same Ethernet segment as the Core reports a Congestion requesting the Core to stop sending Data.
- Core FIFO Congestion: When the Core's receive FIFO reaches a user programmable threshold (rx section empty), the Core sends a Pause Frame back to the Remote device requesting data transfer to be stopped.
- Local Device Congestion: Any device connected to the Core can request (Typically through the Host Processor) the remote device to stop transmitting data.

26.3.9.2 Remote Device Congestion

When the MAC Transmit control gets a valid Pause Quanta from the receive path and if the bit FCE is set to '1', the MAC Transmit logic completes the transfer of the current Frame, stops sending data for the amount of time specified by the Pause Quanta in 512 bit time increments and asserts the bit RFC_PAUSE.

Frame transfer resumes when the time specified by the Quanta has expired and if no new Quanta value has been received or if a new Pause Frame with a Quanta value set to 0x0000 is received. The MAC also resets the bit RFC_PAUSE to '0'.

If the register bit FCE is set to '0', Pause Frames received by the MAC are ignored.

Optionally and independently of the FCE register bit setting, Pause Frame are forwarded to the Client interface if the register bit PAUSE_FWD is set to '1'.

26.3.9.3 Local Device / FIFO Congestion

Pause Frames are generated by the MAC transmit engine, when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when a Pause Frame generation is requested by the Local Host Processor:

- To generate a Pause Frame, the Host Processor asserts the register TFC_PAUSE. A single pause frame is generated when the current Frame transfer is completed and the register bit TFC_PAUSE is automatically cleared. Optionally, an interrupt () is generated.
- A XOFF Pause Frame is generated when the Receive FIFO asserts its section empty flag (internal). A XOFF Pause Frame is generated automatically, when the current Frame transfer is completed.
- A XON Pause Frame is generated when the Receive FIFO de-asserts its section empty flag (internal). A XON Pause Frame is generated automatically, when the current Frame transfer is completed.

When a XOFF Pause Frame is generated, the Pause Quanta (Payload Byte P1 and P2) is filled with the value programmed in the Core register PAUSE_DUR.

The Source Address is set to the MAC address programmed in the Core configuration registers PADDR1 and PADDR2 and the destination address is set to the fixed Multicast address 01-80-C2-00-00-01 (0x010000c28001).

When a XON Pause Frame is generated, the Pause Quanta (Payload Byte P1 and P2) is filled with 0x0000 (Zero Quanta). The Source Address is set to the MAC address programmed in the Core configuration registers PADDR1 and PADDR2 and the destination address is set to the fixed Multicast address 01-80-C2-00-00-01 (0x010000c28001).

Pause Frames generated are compliant to the IEEE 802.3 annex 31A&B.

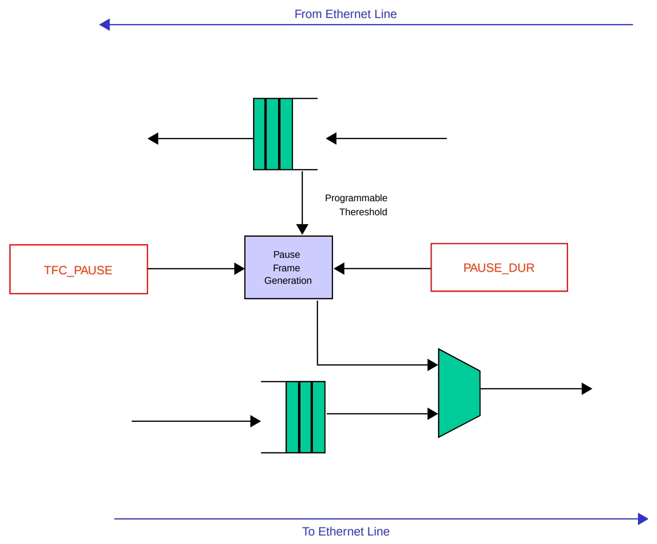


Figure 26-15. Pause Frame Generation Overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC Core receive path, the Core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

26.3.10 Magic Packet Detection

26.3.10.1 Overview

Magic Packet detection is used to wake up a node that is put in the power down mode by the node management agent. Magic Packet detection is supported only if the MAC is configured in sleep mode.

26.3.10.2 Sleep Mode

To put the MAC in sleep mode, the configuration ECR(SLEEP) should be set to °Æ1°Ø. At the same time, ECR(MAGIC_ENA) should be set to 1, to enable magic packet detection.

In addition, when the external input pin ipg_stop is asserted, sleep mode is entered also, without affecting the ECR register bits.

When the Core is in sleep mode:

- The MAC transmit logic is disabled.
- The Core FIFO receive / transmit functions are disabled.

The MAC receive logic is kept in normal mode but it ignores all traffic from the line except Magic Packets, which are detected so that a remote agent can wake up the node.

26.3.10.3 Magic Packet Detection

The Core is designed to detect Magic Packets (see 5.3 page 30) with the destination address set to:

- Any Multicast address.
- The Broadcast address.
- The Unicast address programmed in the Core registers PADDR1 / PADDR2.
- If enabled, to any of the Unicast addresses programmed in the Core supplemental MAC address registers SMAC_0.. to SMAC_3.. .

When a Magic Packets is detected, the interrupt bit EIR(WAKEUP) is asserted and none of the statistic registers is incremented.

In addition, the external pin magic_det is asserted and held asserted until the magic packet detection is disabled or the sleep mode is cancelled.

26.3.10.4 Wakeup

When a Magic Packet was detected, indicated by an asserted EIR(WAKEUP), the configuration bit ECR(SLEEP) should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit will automatically mask the MAGIC_ENA bit, disabling magic packet detection.

This will also deassert the external pin magic_det.

26.3.11 IP Accelerator Functions

26.3.11.1 Checksum Calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

For ICMP, the checksum is calculated over the complete ICMP datagram (that is, without IP header).

For TCP and UDP, the checksums contain the header and data sections and in addition the values from the IP header, which can be seen as a pseudo header that is not actually present as such in the datastream.

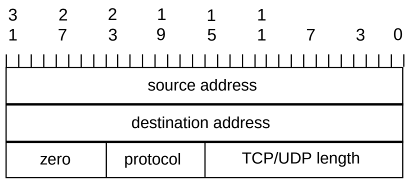


Figure 26-16. IPv4 Pseudo Header for Checksum calculation

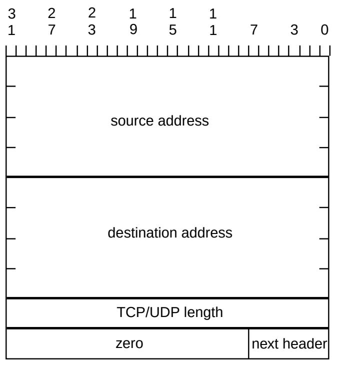


Figure 26-17. IPv6 Pseudo Header for Checksum Calculation

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4) or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header and 'zero' is filled with zeroes.

For IPv6, the complete 128 bit addresses are considered. The next header value identifies the upper layer protocol (TCP or UDP) and may differ from the IPv6 header's actual next header value. if the extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the most significant byte and the second byte sent/received is the least significant byte of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only (not actually transmitted).

26.3.11.2 Additional Padding Processing

Any Ethernet frame according to IEEE 802.3 must have a minimum length of 64 octets. The MAC usually removes padding on receive, when a frame with length information is received. As IP frames have a type value instead of the length, the MAC will not remove padding for short IP frames, as it is not aware of the frame contents.

The IP Accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically ensures to add padding as necessary to fill any frame to a length of 64.

26.3.11.3 32-bit Ethernet Payload Alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to the alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ipaccTxConf register and ipaccRxConf register respectively.

When enabled, the valid frame data is arranged as shown in the table below.

Table 26-22. 64-Bit Interface Data Structure with SHIFT16 option enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value

Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

Receive Processing

When the SHIFT16 bit in ipaccRxConf register is enabled, each frame is received with two additional bytes in front of the frame. The user application has to ignore these first two bytes and has to find the first byte of the frame in bits 23:16 of the first word from the RX FIFO.

Note that the SHIFT16 bit must be set during initialization and kept asserted during the complete operation as it influences the FIFO write behavior.

Transmit Processing

When the SHIFT16 bit in ipaccTxConf register is enabled, the first two bytes of the first word written (that is, bits 15:0) will be discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only in-between frames.

26.3.11.4 Received Frame Discard

As the Receive FIFO must be operated in store and forward mode (Register RX_SECTION_FULL must be set 0), received Frames can be discarded based on the following errors:

1. The MAC function receives the frame with an error:
    - The Frame has an invalid payload length.
    - Frame length is greater than MAX_FL.
    - Frame received with a CRC-32 error.
    - Frame truncated due to receive FIFO overflow.
    - Frame is corrupted as PHY signaled an error (mii_rx_err asserted during reception).
2. An IP frame is detected and the IP header checksum is wrong
3. An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol specific checksum is wrong

If one of the errors occurs and the IP Accelerator function is configured to discard frames (Register ipaccRxConf), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

26.3.11.5 IPv4 Fragments

When an IP (IPv4) fragment frame is received, only the IP header is inspected and its checksum is verified. The 32-bit alignment operates on fragments as on normal IP frames, as specified above.

The IP fragment frame payload is not inspected for any protocol headers, as such a protocol header would only exist in the very first fragment. To assist in the protocol specific checksum verification, the one's-complement sum is calculated on the IP payload (that is, all bytes following the IP header) and provided with the frame status word.

Note

The application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol specific checksum of the datagram after all the fragments have been received and reassembled.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo header checksum value from the first fragment and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

26.3.11.6.1 Receive Processing

An Ethernet frame of type 0x86dd identifies an IP Version 6 frame (IPv6) Frame. If an IPv6 frame is received, the first IP header is inspected (first 10 words) which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing (see 12.3 page 54).

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol specific checksum verification, the one's-complement sum is always calculated on the IP payload (that is, all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sum(s) can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum if required.

26.3.11.6.2 Transmit Processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum therefore, the IP checksum insertion configuration is ignored.

The protocol checksum will be inserted only if the next header of the IP header is a known protocol (that is, TCP, UDP or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

26.3.12 Resets and Stop Controls

26.3.12.1 Hardware Reset

Setting the ECR(RESET) bit, resets the Ethernet controller and initializes all registers and logic. This bit is self-clare.

26.3.12.2 Soft Reset

When ECR(ETHER_EN) is cleared, during operation, the following happens:

- DMA, buffer descriptor and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- Transmission is terminated by asserting mii_tx_err to the PHY.
- The current FIFO write is terminated and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- The current FIFO read is terminated.

26.3.12.3 Graceful Stop

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths. A graceful stop means that any currently ongoing transactions are completed normally and no further frames after that will be accepted. The MAC can resume from a graceful stop without the need for a reset (for example, ECR(ETHER_EN) bit deassertion is not required).

26.3.12.4 Graceful Transmit Stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission. In any of the following conditions, the transmit datapath will stop after an ongoing frame transmission has been completed normally.

- Register bit TCR(GTS) is set by software.
- Register bit TCR(TFC_PAUSE) is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expired or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by set SLEEP or SLEEP bit (see Sleep Mode).

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted (once, when transitioned into stopped).

26.3.12.5 Graceful Receive Stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO. If any of the following conditions occur, the receive datapath will stop after any ongoing frame reception has been completed normally.

- MAC is placed in sleep mode (by set SLEEP or SLEEP). The MAC will continue to receive frames and hunt for magic packets if enabled (see Sleep Mode). However, no frames will be written into the receive FIFO and therefore will not be forwarded to the application.

When the receive datapath is stopped, the following events will occur:

- The RCR(GRS) bit is set as long as the RX is in the stopped state.
- The GRA interrupt is asserted when the transmitter is also stopped (both TX and RX are stopped).
- Any ongoing receive transaction to the application (RX FIFO read) will continue normally until the frame is completed (eop). After this, the following happens:
  - When sleep mode is active, all further frames will be discarded, flushing the rx FIFO.

Note

The assertion of GRS will not wait for an ongoing transaction on the application side of the FIFO (FIFO read). If necessary, the signal can be ANDed with the ff_rx_dval (considering proper clock domains).

26.3.12.6 Graceful Stop Interrupt (GRA)

The graceful stopped interrupt (GRA) is asserted for the following conditions:

- If sleep mode is active, the interrupt asserts only after both TX and RX datapaths are stopped.
-
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt will be triggered.

26.3.13 IEEE 1588 Functions

26.3.13.1 Overview

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time stamping of incoming and outgoing frames. 1588 Support is enabled when the register bit ENA_1588 is set to '1'.

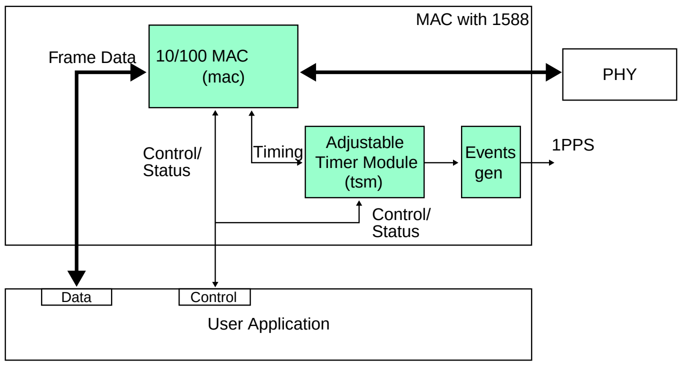


Figure 26-18. IEEE 1588 Functions Overview

26.3.13.2.1 Overview

The Adjustable Timer Module (TSM) implements the Free Running Counter (FRC), the timer, which is used to generate the timestamps. The FRC operates with the clock CLK_ENET_TIME, which can be set to any value depending on the system requirements. However, it is recommended to choose a period, which is an integer value (for example, 5 ns, 6 ns, 8 ns, but not 7.5 ns) to implement a precise timer.

The current value of the FRC is available on the Core toplevel pins frc().

Through dedicated correction logic (see below), the timer can be adjusted allowing synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed period of time to allow synchronization of software timers or perform other synchronized system functions.

The timer is usually used to implement a period of 1 second, hence, its value would range from 0 to $(1*10^9)$-1. The period event can trigger an interrupt and software then can maintain the seconds and hours time values and so on as necessary.

26.3.13.2.2 Adjustable Timer Implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and its increment rate are freely configurable allowing very fine tuning of the timer.

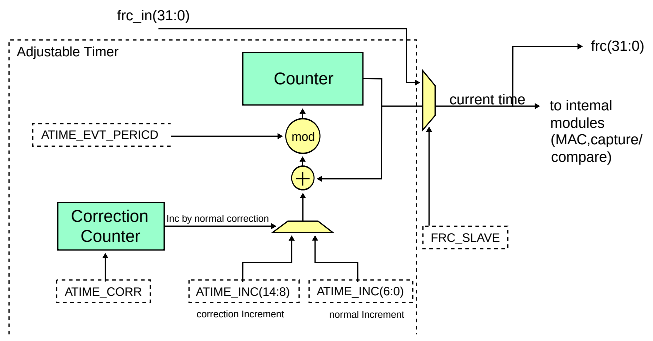


Figure 26-19. Adjustable Timer Implementation Detail

The counter is producing the current time. Each clock cycle (CLK_ENET_TIME), a constant value is added to the current time as programmed in the ATIME_INC(4:0) register. The value depends on the chosen clock frequency of CLK_ENET_TIME. For example, if the CLK_ENET_TIME operates at 125 MHz, the increment would be set to 8 representing 8 ns.

The period, configured in register ATIME_EVT_PERIOD, defines the modulo when the counter has to wrap around. In a typical implementation, the period is set to $1*10^{9,}$ so the counter wraps every 1 second and hence, all timestamps represent the absolute nanoseconds within the 1 second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it will not necessarily start from 0 but instead the counter will be loaded with the value (current + inc - ($1*10^9$)) assuming the period was set to $1*10^9$.

The correction counter operates fully independently and increments by 1 with each clock cycle (CLK_ENET_TIME). When it reaches the value configured in ATIME_CORR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value. The normal and correction increments are configured in register ATIME_INC. To speed up the timer, the correction increment would be larger than the normal increment value. To slow down the timer, the correction increment would be smaller than the normal increment value. Note that the correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections (and hence low jitter) happening in the range of 1ns independently from the chosen clock frequency.

By enabling the slave mode (ATIME_CTRL(FRC_SLAVE)), the timer is ignored and the current time is the externally provided time from pins frc_in(). This is useful if multiple modules within the system need to operate from a single timer (see below).

When slave mode is enabled, it is still required to set the ATIME_INC(6:0) increment value to the value of the master as it is used for internal comparisons.

26.3.13.3 Timer Synchronization for Multi-Port Implementations

An additional timer input is available on the Core toplevel pins frc_in() to allow to use an externally provided timer value for all time stamping functions. This is necessary if multiple instances (ports) of the MAC should be synchronized to a single reference timer in the system. In i.MX28, ENET-MAC0 is used as master, so the timer input (frc_in) of ENET_MAC0 is unused and wired to all 0. The FRC value of ENET_MAC0 is wired to frc_in of ENET_MAC1 (acting as slave). To operate the MAC in Slave mode, the timer module configuration ATIME_CTRL(FRC_SLAVE) is used to disable the internal adjustable timer and use only the externally provided time value.

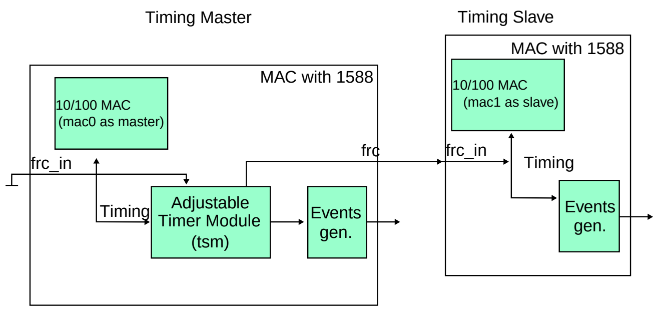


Figure 26-20. 1588 Multiple MAC implementation

26.3.13.4 Transmit Timestamping

On transmit, only 1588 event frames need to be time-stamped. The Client application (for example, the MAC driver) should detect 1588 event frames and set 'TS' bit in Enhanced uDMA Transmit Buffer Descriptor.

For every transmitted frame, the MAC returns the captured timestamp in '1588 Timestamp' and the transmit status TSE in Enhanced uDMA Transmit Buffer Descriptor. The transmit status bit indicates that the application had the TS bit asserted for the frame.

If TS is set to '1', the MAC additionally memorizes the timestamp for the frame in the register TS_TIMESTAMP. The interrupt bit EIR(TS_AVAIL) is set to indicate that a new timestamp is available.

Software would implement a handshaking procedure by setting the TS bit when it transmits the frame it needs a timestamp for and then waits on the EIR(TS_AVAIL) interrupt bit to know when the timestamp is available. It then can read the timestamp from the TS_TIMESTAMP register. This is done for all event frames, other frames will not use the TS indicator and hence will not interfere with the timestamp capture.

26.3.13.5 Receive Timestamping

When a frame is received, the MAC latches the value of the timer when the frame SFD field is detected and provides the captured timestamp '1588 Timestamp' in Enhanced uDMA Receive Buffer Descriptor. This is done for all received frames.

The DMA controller has to ensure that it transfers the timestamp provided for the frame into the corresponding field within the receive descriptor for software access.

26.3.13.6 Time Synchronization

To synchronize the local clock of a node to a remote master, the adjustable timer module is available. It implements a free running 32-bit counter, which has an additional correction counter connected to it. The correction counter is used to increase or decrease the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application (software) implements, in a slave scenario, the required control algorithm setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp related information should be configured to show the true nanoseconds value of a second (that is, the timer is configured to have a period of 1 second). Hence, the values range from 0 to $(1*10^9)-1$. In this application, the seconds counter is implemented in software using an interrupt function that is executed whenever the nanoseconds counter wraps at $1*10^9$.

Note

Implementing the seconds counter in software is not a strict requirement.

26.3.13.7 Capture/Compare Block

The Capture / Compare block can provide precise hardware timing for four input events and four output events.

The capture registers (CAPT_REG..) latch the time value when the corresponding external event occurs (evt_in()). If the corresponding interrupt bit is enabled in the CCB_INT register, an interrupt can be generated.

The compare registers (COMP_REG_..) are loaded with the time at which the corresponding event should occur. The timer reached the value, the corresponding output pin (evt_out()) is asserted. If the corresponding interrupt bit is enabled in the EIMR register, an interrupt can be generated.

26.3.14 FIFO Thresholds

26.3.14.1 Overview

The Core FIFO thresholds are all fully programmable to provide the possibility to dynamically change the FIFO operation. For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers. The thresholds are defined in 64-bit words.

26.3.14.2 Receive FIFO

Four programmable thresholds are available which are programmed with the registers RX_ALMOST_EMPTY, RX_ALMOST_FULL, RX_SECTION_EMPTY and RX_SECTION_FULL which can be set to any value to control operation as follows table.

Table 26-23. Receive FIFO Thresholds Definition

Register	Description
RX_ALMOST_EMPTY	When the FIFO level reaches the value programmed in the register RX_ALMOST_EMPTY, and the end-of-frame has not been received for the frame yet, the Core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 6 should be set.
RX_ALMOST_FULL	When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC control logic stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.
RX_SECTION_EMPTY	When the FIFO level reaches the value programmed in the register RX_SECTION_EMPTY, an indication is provided to the MAC transmit logic, which generates a XOFF Pause frame to indicate FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in the register RX_SECTION_EMPTY, an indication is provided to the MAC transmit logic, which generates a XON Pause frame to indicate the FIFO congestion is cleared to the remote Ethernet client. A value of 0 disables any pause frame generation.
RX_SECTION_FULL	When the FIFO level reaches the value programmed in the register RX_SECTION_FULL, the MAC will indicate uDMA that data is available in the Receive FIFO (cut-through operation) by asserted a internal signal. MAC will deassert the siganl again after asserted, if the fifo empties below the threshold set with RX_ALMOST_EMPTY, and if the end-of-frame is not yet stored in the fifo (Hysteresis). If a Frame has a size smaller than the threshold (i.e. an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, set the register RX_SECTION_FULL to 0. In this case, The signal is asserted only when a complete frame is stored in the receive FIFO. When programming a value>0 (cut-through operation) it should be greater than RX_ALMOST_EMPTY.

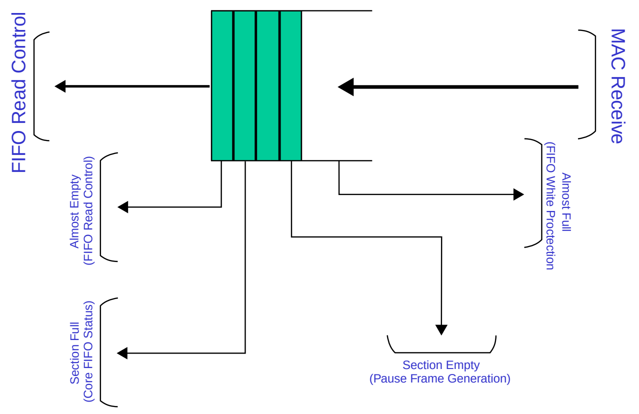

FIFO Read Control
MAC Receive
Almost Empty (FIFO Read Control)
Almost Full (FIFO White Proctection)
Section Full (Core FIFO Status)
Section Empty (Pause Frame Generation)

Figure 26-21. Receive FIFO Overview

26.3.14.3 Transmit FIFO

Four programmable thresholds are available which are programmed with registers TX_ALMOST_EMPTY, TX_ALMOST_FULL, TX_SECTION_EMPTY and TFWR / STR_FWD.

Table 26-24. Transmit FIFO Thresholds Definition

Register	Description
TX_ALMOST_EMPTY	When the FIFO level reaches the value programmed in the register TX_ALMOST_EMPTY, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits the Ethernet frame with a MII error indication. A minimum value of 4 should be set.
TX_ALMOST_FULL	When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result the frame will be transmitted with an MII error indication. A minimum value of 4 should be set. Larger values allow more latency for the application to react on ff_tx_rdy deassertion, before the frame is being truncated. A typical setting is 8, which offers 3-4 clock cycles of latency to the application to react on ff_tx_rdy deassertion.
TX_SECTION_EMPTY	When the FIFO level reaches the value programmed in the register TX_SECTION_EMPTY, the Core status ff_tx_septy is deasserted to indicate that the Transmit FIFO is getting full. This gives the application an indication to slow down or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When a value of 0 is set, the signal is never deasserted.
TFWR / STR_FWD	When the FIFO level reaches the value coded in the register TFWR and when the register bit STR_FWD is set to '0', the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation). If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line. To enable store and forward on the Transmit path, set the register bit STR_FWD to '1'. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

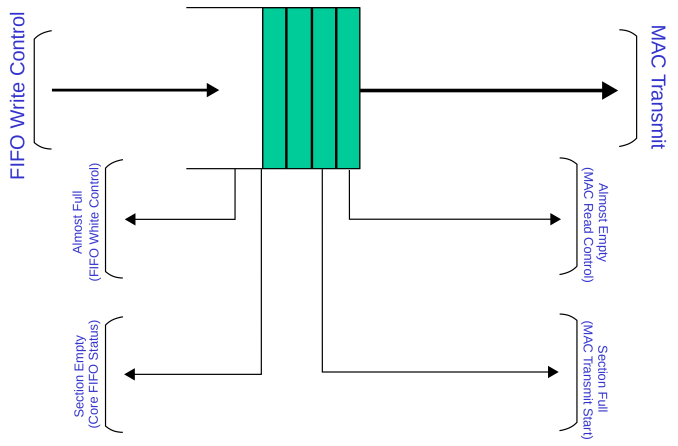

FIFO Write Control
MAC Transmit
Almost Full (FIFO Write Control)
Almost Empty (MAC Read Control)
Section Empty (Core FIFO Status)
Section Full (MAC Transmit Start)

Figure 26-22. Transmit FIFO Overview

26.3.15 Loopback Options

26.3.15.1 Overview

The Core implements external and internal loopback options with are controlled by the RCR register bits LOOP.

Table 26-25. Loopback Options

Register Bit	Description
LOOP	Internal MII Loopback The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces.
RMII_LOOP	External RMII Near End Loopback. Not surpport now.
RMII_ECHO	External RMII Far End Loopback. Not surpport now.

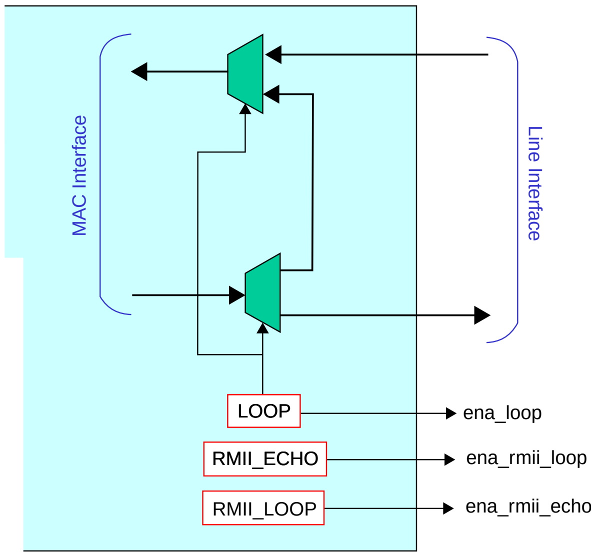


Figure 26-23. Loopback Options

26.3.16 FIFO Protection

26.3.16.1 Transmit FIFO Underflow

When the Transmit FIFO, during a frame transfer, reaches the almost empty threshold with no End of Frame indication stored in the FIFO, the MAC logic stops reading data from the FIFO, sets the MII error signal (mii_tx_err) to '1' to indicate that the fragment already transferred is not valid and deasserts the MII transmit enable signal (mii_tx_en) to terminate the frame transfer .

When after an underflow, the application completes the frame transfer , the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached and sets bit UE of Enhanced uDMA Transmit Buffer Descriptor.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication .

26.3.16.2 Transmit FIFO Overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, if the application keeps sending new data, the transmit FIFO will overflow, corrupting previously stored contents. ENET-MAC will set bit OE of Enhanced uDMA Transmit Buffer Descriptor for the next frame transmitted to indicate this overflow occurrence.

Note that overflow is a fatal error and must be addressed by resetting the core or deasserting the ECR(ETHER_EN) bit to clear the FIFOs and prepare for normal operation again.

26.3.16.3 Receive FIFO Overflow

If, during a frame reception the client application is not able to receive data , the MAC receive control, when the FIFO reaches the programmable almost full threshold truncates the incoming frame (To avoid a overflow). The frame is subsequently received on the FIFO interface with an error indication (ME of Enhanced uDMA Receive Buffer Descriptor) with the truncation error status bit (TR of Legacy FEC Receive Buffer Descriptor) set to '1'.

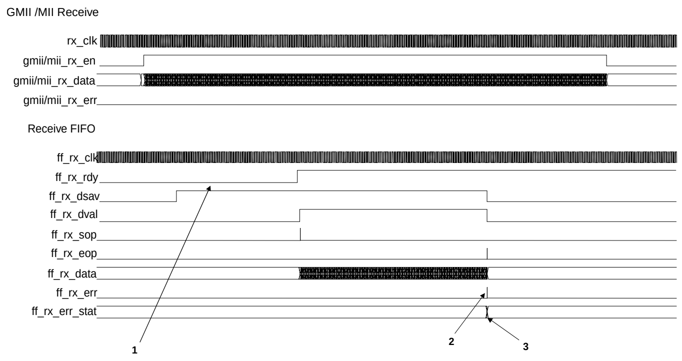


Figure 26-24. Receive FIFO Overflow Protection

26.3.17 PHY Management Interface

26.3.17.1 Overview

The MDIO interface is a two-wire Management Interface. The MDIO management interface implements a standardized method to access the PHY device management registers. This is a Master MDIO interface, which can be connected to up to 32 PHY devices.

26.3.17.2 MDIO Frame Format

The MDIO Master controller communicates with the Slave (PHY device) using Frames, which are defined in Table 26-26. A complete frame has a length of 64 bits (Optional 32 bit preamble, 14 bit command, 2 bit bus direction change, 16 bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC signal). The PHY Management interface supports the standard MDIO specification (IEEE803.2 Clause 22).

Table 26-26. MDIO Frame Formats (Read / Write)

Type		Command			

	PRE	ST MSB LSB	OP MSB LSB	Addr1 MSB LSB	Addr2 MSB LSB	TA	Data MSB LSB	Idle
Read	1...1	01	10	xxxxx	xxxxx	Z0	xxxxxxxxxxxxxxxx	Z
Write	1...1	01	01	xxxxx	xxxxx	10	xxxxxxxxxxxxxxxx	Z

Table 26-27. MDIO Frame Fields Description

Name	Description
PRE	Preamble: 32 Bits of logical '1' sent prior to every transaction when the register bit DIS_PRE is set to '0. If DIS_PRE is set to '1', the preamble is not generated.
ST	Start indication, programmed with the register bits ST: Standard MDIO (Clause 22): '01'
OP	The opcode, programmed with the register bits OP, defines whether a read or write operation is performed: If '10' a read operation is performed. If '01' a write operation is performed.
Addr1	The PHY device address, programmed with the register bits PA. Up to 32 devices can be addressed.
Addr2	Register Address, programmed with the register bits RA. Each PHY can implement up to 32 registers.
TA	Turnaround time, programmed with register bits TA. Two bit-times are reserved for read operations to switch the data bus from write to read for read operations. The PHY device will present its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data	16 bits of data written, set to register bits DATA, to the PHY or read from the PHY on register bits DATA.
Idle	Between frames, the MDIO data signal is tri-stated.

26.3.17.3 MDIO Clock Generation

The MDC clock is generated from the register interface clock pclk divided by the value programmed in the Core register MII_SPEED.

26.3.17.4 MDIO Operation

To perform a MDIO access, the MDIO Command register (Register MMFR) should be set according to the description provided in MAC_MMFR. To check when the programmed access is completed, the EIR status register should be read and the register bit MII checked.

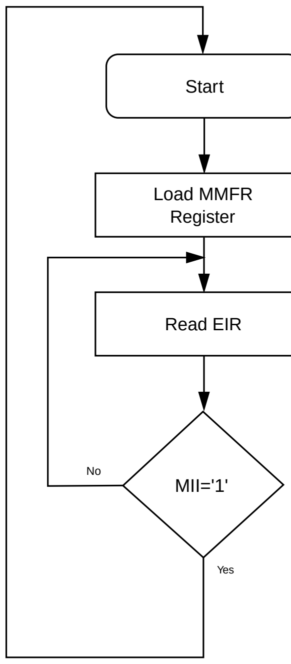


Figure 26-25. MDIO Access Overview

26.3.17.5 MDIO Buffer Connection

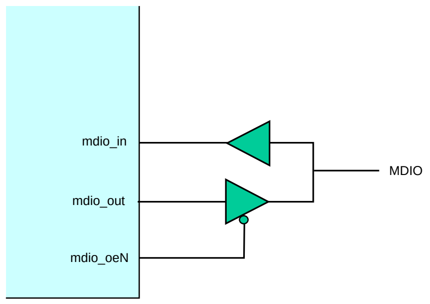


Figure 26-26. MDIO Buffer Connection

26.3.18 MII Interface

26.3.18.1 Transmit

On Transmit, all data transfers are synchronous to tx_clk rising edge. The MII data enable signal mii_tx_en is asserted to indicate the start of a new frame and remains asserted until the last byte of the frame is present on mii_tx_d(3:0) bus. Between frames, mii_tx_en remains de-asserted.

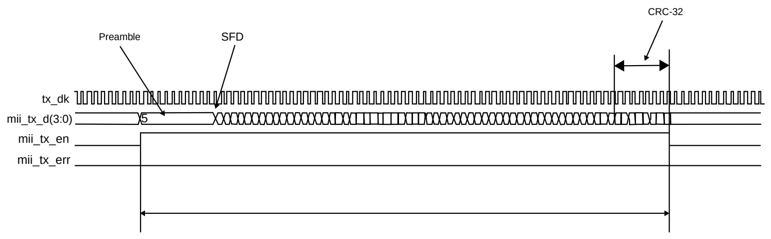


Figure 26-27. MII Transmit Operation

If a frame is received on the FIFO interface with an error (for example, Signal ff_rx_err asserted) the frame is subsequently transmitted with the MII mii_tx_err error signal for one clock cycle at any time during the packet transfer.

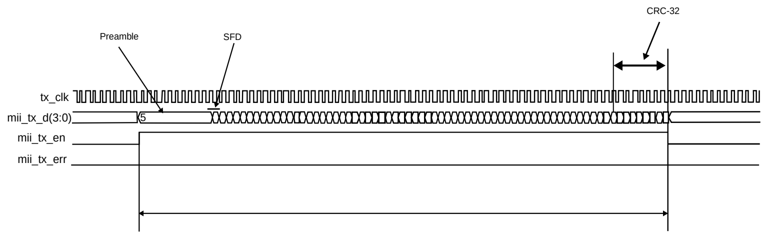


Figure 26-28. MII Transmit Operation - Errored Frame

26.3.18.2 Transmit with Collision - Half Duplex

When a collision is detected during a frame transmission (MII signal mii_col asserted), the MAC stops the current transmission, sends a 32-Bit jam pattern and re-transmits the current frame (See section Collision Detection in Half Duplex Mode for detail).

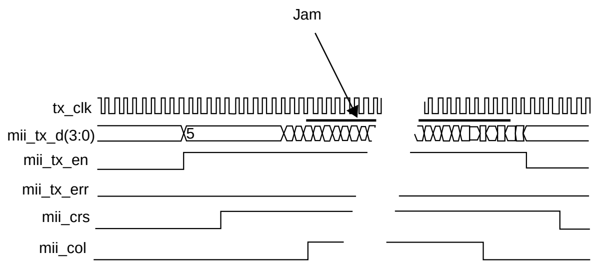


Figure 26-29. MII Transmit Operation - Transmission with Collision

26.3.18.3 Receive

On Receive, all signals are sampled on the rx_clk rising edge. The MII data enable signal mii_rx_dv is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on mii_rx_d(3:0) bus. Between frames, mii_rx_dv remains de-asserted.

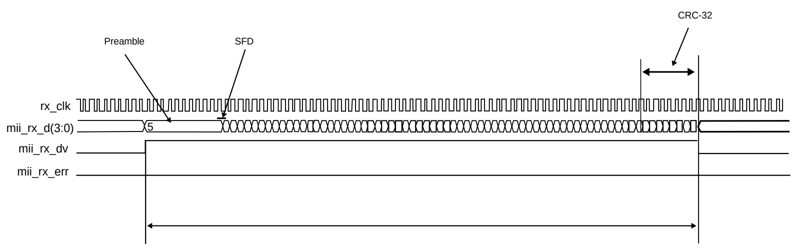


Figure 26-30. MII Receive Operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, mii_rx_err, for at least one clock cycle at any time during the packet transfer.

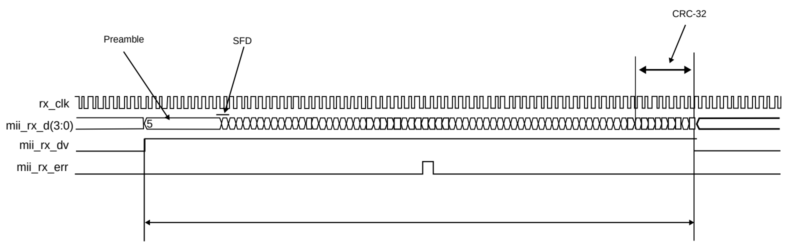
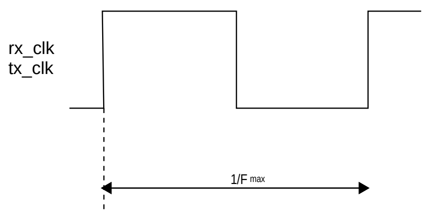
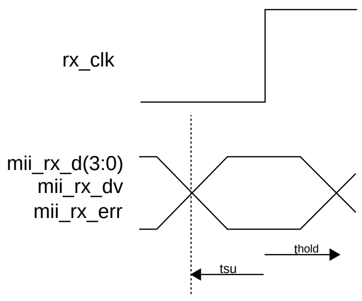


Figure 26-31. MII Receive Operation - Errored Frame

26.3.19 MII Interface Timing Characteristics





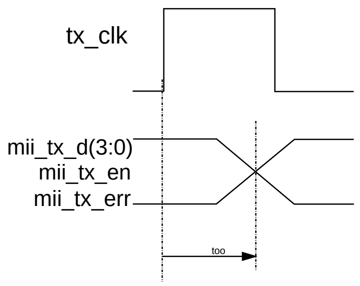


Table 26-28. MII Timing Characteristics

Symbol	Description	Value	
		Min	Max
Fmax	MII Maximum Frequency	2.5 MHz	25 MHz
tsu	MII Inputs Setup Time	2 ns	-
thold	MII Inputs Hold Time	0 ns	-
tco	MII Outputs Clock to Out	-	5 ns

26.4 Programmable Registers

Each ENET-MAC with its corresponding management counters implements a register space of 512 32-bit registers. All reads and writes must be 32 bits wide. Reserved bits should be written with 0 and ignored on read to allow future extension. Unused registers read back 0 and a write has no effect.

ENET-MAC0 base address is 0x800F0000; ENET-MAC1 base address is 0x800F4000.

HW_ENET memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_0004	ENET MAC Interrupt Event Register (HW_ENET_MAC_EIR)	32	R/W	0000_0000h	26.4.1/1661
800F_0008	ENET MAC Interrupt Mask Register (HW_ENET_MAC_EIMR)	32	R/W	0000_0000h	26.4.2/1663
800F_0010	ENET MAC Receive Descriptor Active Register (HW_ENET_MAC_RDAR)	32	R/W	0000_0000h	26.4.3/1663
800F_0014	ENET MAC Transmit Descriptor Active Register (HW_ENET_MAC_TDAR)	32	R/W	0000_0000h	26.4.4/1664
800F_0024	ENET MAC Control Register (HW_ENET_MAC_ECR)	32	R/W	F000_0000h	26.4.5/1665
800F_0040	ENET MAC MII Management Frame Register (HW_ENET_MAC_MMFR)	32	R/W	0000_0000h	26.4.6/1666
800F_0044	ENET MAC MII Speed Control Register (HW_ENET_MAC_MSCR)	32	R/W	0000_0000h	26.4.7/1667

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_0064	ENET MAC MIB Control/Status Register (HW_ENET_MAC_MIBC)	32	R/W	C000_0000h	26.4.8/1668
800F_0084	ENET MAC Receive Control Register (HW_ENET_MAC_RCR)	32	R/W	05EE_0001h	26.4.9/1669
800F_00C4	ENET MAC Transmit Control Register (HW_ENET_MAC_TCR)	32	R/W	0000_0000h	26.4.10/1671
800F_00E4	ENET MAC Physical Address Lower Register (HW_ENET_MAC_PALR)	32	R/W	0000_0000h	26.4.11/1672
800F_00E8	ENET MAC Physical Address Upper Register (HW_ENET_MAC_PAUR)	32	R/W	0000_8808h	26.4.12/1673
800F_00EC	ENET MAC Opcode/Pause Duration Register (HW_ENET_MAC_OPD)	32	R/W	0001_0000h	26.4.13/1673
800F_0118	ENET MAC Descriptor Individual Upper Address Register (HW_ENET_MAC_IAUR)	32	R/W	0000_0000h	26.4.14/1674
800F_011C	ENET MAC Descriptor Individual Lower Address Register (HW_ENET_MAC_IALR)	32	R/W	0000_0000h	26.4.15/1674
800F_0120	ENET MAC Descriptor Group Upper Address Register (HW_ENET_MAC_GAUR)	32	R/W	0000_0000h	26.4.16/1675
800F_0124	ENET MAC Descriptor Group Lower Address Register (HW_ENET_MAC_GALR)	32	R/W	0000_0000h	26.4.17/1675
800F_0144	ENET MAC Transmit FIFO Watermark and Store and Forward Control Register (HW_ENET_MAC_TFW_SFCR)	32	R/W	0000_0000h	26.4.18/1676
800F_014C	ENET MAC FIFO Receive Bound Register (HW_ENET_MAC_FRBR)	32	R/W	0000_0600h	26.4.19/1677
800F_0150	ENET MAC FIFO Receive FIFO Start Register (HW_ENET_MAC_FRSR)	32	R/W	0000_0500h	26.4.20/1677
800F_0180	ENET MAC Pointer to Receive Descriptor Ring Register (HW_ENET_MAC_ERDSR)	32	R/W	0000_0000h	26.4.21/1678
800F_0184	ENET MAC Pointer to Transmit Descriptor Ring Register (HW_ENET_MAC_ETDSR)	32	R/W	0000_0000h	26.4.22/1679
800F_0188	ENET MAC Maximum Receive Buffer Size Register (HW_ENET_MAC_EMRBR)	32	R/W	0000_0000h	26.4.23/1680
800F_0190	ENET MAC Receive FIFO Section Full Threshold Register (HW_ENET_MAC_RX_SECTION_FULL)	32	R/W	0000_0000h	26.4.24/1681
800F_0194	ENET MAC Receive FIFO Section Empty Threshold Register (HW_ENET_MAC_RX_SECTION_EMPTY)	32	R/W	0000_0000h	26.4.25/1681
800F_0198	ENET MAC Receive FIFO Almost Empty Threshold Register (HW_ENET_MAC_RX_ALMOST_EMPTY)	32	R/W	0000_0004h	26.4.26/1682
800F_019C	ENET MAC Receive FIFO Almost Full Thresholdt Register (HW_ENET_MAC_RX_ALMOST_FULL)	32	R/W	0000_0004h	26.4.27/1682
800F_01A0	ENET MAC Transmit FIFO Section Empty Threshold Register (HW_ENET_MAC_TX_SECTION_EMPTY)	32	R/W	0000_0000h	26.4.28/1683

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_01A4	ENET MAC Transmit FIFO Almost Empty Threshold Register (HW_ENET_MAC_TX_ALMOST_EMPTY)	32	R/W	0000_0004h	26.4.29/1683
800F_01A8	ENET MAC Transmit FIFO Almost Full Threshold Register (HW_ENET_MAC_TX_ALMOST_FULL)	32	R/W	0000_0008h	26.4.30/1684
800F_01AC	ENET MAC Transmit Inter-Packet Gap Register (HW_ENET_MAC_TX_IPG_LENGTH)	32	R/W	0000_000Ch	26.4.31/1684
800F_01B0	ENET MAC Frame Truncation Length Register (HW_ENET_MAC_TRUNC_FL)	32	R/W	0000_07FFh	26.4.32/1685
800F_01C0	ENET MAC Accelerator Transmit Function Configuration Register (HW_ENET_MAC_IPACCTXCONF)	32	R/W	0000_0000h	26.4.33/1685
800F_01C4	ENET MAC Accelerator Receive Function Configuration Register (HW_ENET_MAC_IPACCRXCONF)	32	R/W	0000_0000h	26.4.34/1686
800F_0200	ENET MAC RMON Tx packet drop (HW_ENET_MAC_RMON_T_DROP)	32	R	0000_0000h	26.4.35/1687
800F_0204	ENET MAC RMON Tx packet count (HW_ENET_MAC_RMON_T_PACKETS)	32	R	0000_0000h	26.4.36/1688
800F_0208	ENET MAC RMON Tx Broadcast Packets (HW_ENET_MAC_RMON_T_BC_PKT)	32	R	0000_0000h	26.4.37/1688
800F_020C	ENET MAC RMON Tx Multicast Packets (HW_ENET_MAC_RMON_T_MC_PKT)	32	R	0000_0000h	26.4.38/1689
800F_0210	ENET MAC RMON Tx Packets w CRC/Align error (HW_ENET_MAC_RMON_T_CRC_ALIGN)	32	R	0000_0000h	26.4.39/1689
800F_0214	ENET MAC RMON Tx Packets < 64 bytes, good CRC (HW_ENET_MAC_RMON_T_UNDERSIZE)	32	R	0000_0000h	26.4.40/1690
800F_0218	ENET MAC RMON Tx Packets > MAX_FL bytes, good CRC (HW_ENET_MAC_RMON_T_OVERSIZE)	32	R	0000_0000h	26.4.41/1690
800F_021C	ENET MAC RMON Tx Packets < 64 bytes, bad CRC (HW_ENET_MAC_RMON_T_FRAG)	32	R	0000_0000h	26.4.42/1691
800F_0220	ENET MAC RMON Tx Packets > MAX_FL bytes, bad CRC (HW_ENET_MAC_RMON_T_JAB)	32	R	0000_0000h	26.4.43/1691
800F_0224	ENET MAC RMON Tx collision count (HW_ENET_MAC_RMON_T_COL)	32	R	0000_0000h	26.4.44/1692
800F_0228	ENET MAC RMON Tx 64 byte packets (HW_ENET_MAC_RMON_T_P64)	32	R	0000_0000h	26.4.45/1692
800F_022C	ENET MAC RMON Tx 65 to 127 byte packets (HW_ENET_MAC_RMON_T_P65TO127N)	32	R	0000_0000h	26.4.46/1693
800F_0230	ENET MAC RMON Tx 128 to 255 byte packets (HW_ENET_MAC_RMON_T_P128TO255N)	32	R	0000_0000h	26.4.47/1693
800F_0234	ENET MAC RMON Tx 256 to 511 byte packets (HW_ENET_MAC_RMON_T_P256TO511)	32	R	0000_0000h	26.4.48/1694
800F_0238	ENET MAC RMON Tx 512 to 1023 byte packets (HW_ENET_MAC_RMON_T_P512TO1023)	32	R	0000_0000h	26.4.49/1694

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_023C	ENET MAC RMON Tx 1024 to 2047 byte packets (HW_ENET_MAC_RMON_T_P1024TO2047)	32	R	0000_0000h	26.4.50/1695
800F_0240	ENET MAC RMON Tx packets w > 2048 bytes (HW_ENET_MAC_RMON_T_P_GTE2048)	32	R	0000_0000h	26.4.51/1695
800F_0244	ENET MAC RMON Tx Octets (HW_ENET_MAC_RMON_T_OCTETS)	32	R	0000_0000h	26.4.52/1696
800F_0248	ENET MAC Frames Transmitted count drop (HW_ENET_MAC_IEEE_T_DROP)	32	R	0000_0000h	26.4.53/1696
800F_024C	ENET MAC Frames Transmitted OK (HW_ENET_MAC_IEEE_T_FRAME_OK)	32	R	0000_0000h	26.4.54/1697
800F_0250	ENET MAC Frames Transmitted with Single Collision (HW_ENET_MAC_IEEE_T_1COL)	32	R	0000_0000h	26.4.55/1697
800F_0254	ENET MAC Frames Transmitted with Multiple Collisions (HW_ENET_MAC_IEEE_T_MCOL)	32	R	0000_0000h	26.4.56/1698
800F_0258	ENET MAC Frames Transmitted after Deferral Delay (HW_ENET_MAC_IEEE_T_DEF)	32	R	0000_0000h	26.4.57/1698
800F_025C	ENET MAC Frames Transmitted with Late Collision (HW_ENET_MAC_IEEE_T_LCOL)	32	R	0000_0000h	26.4.58/1699
800F_0260	ENET MAC Frames Transmitted with Excessive Collisions (HW_ENET_MAC_IEEE_T_EXCOL)	32	R	0000_0000h	26.4.59/1699
800F_0264	ENET MAC Frames Transmitted with Tx FIFO Underrun (HW_ENET_MAC_IEEE_T_MACERR)	32	R	0000_0000h	26.4.60/1700
800F_0268	ENET MAC Frames Transmitted with Carrier Sense Error (HW_ENET_MAC_IEEE_T_CSERR)	32	R	0000_0000h	26.4.61/1700
800F_026C	ENET MAC Frames Transmitted with SQE Error (HW_ENET_MAC_IEEE_T_SQE)	32	R	0000_0000h	26.4.62/1700
800F_0270	ENET MAC Frames Transmitted flow control (HW_ENET_MAC_IEEE_T_FDXFC)	32	R	0000_0000h	26.4.63/1701
800F_0274	ENET MAC Frames Transmitted error (HW_ENET_MAC_IEEE_T_OCTETS_OK)	32	R	0000_0000h	26.4.64/1701
800F_0284	ENET MAC RMON Rx packet count (HW_ENET_MAC_RMON_R_PACKETS)	32	R	0000_0000h	26.4.65/1702
800F_0288	ENET MAC RMON Rx Broadcast Packets (HW_ENET_MAC_RMON_R_BC_PKT)	32	R	0000_0000h	26.4.66/1702
800F_028C	ENET MAC RMON Rx Multicast Packets (HW_ENET_MAC_RMON_R_MC_PKT)	32	R	0000_0000h	26.4.67/1703
800F_0290	ENET MAC RMON Rx Packets w CRC/Align error (HW_ENET_MAC_RMON_R_CRC_ALIGN)	32	R	0000_0000h	26.4.68/1703
800F_0294	ENET MAC RMON Rx Packets < 64 bytes, good CRC (HW_ENET_MAC_RMON_R_UNDERSIZE)	32	R	0000_0000h	26.4.69/1704
800F_0298	ENET MAC RMON Rx Packets > MAX_FL, good CRC (HW_ENET_MAC_RMON_R_OVERSIZE)	32	R	0000_0000h	26.4.70/1704

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_029C	ENET MAC RMON Rx Packets < 64 bytes, bad CRC (HW_ENET_MAC_RMON_R_FRAG)	32	R	0000_0000h	26.4.71/1705
800F_02A0	ENET MAC RMON Rx Packets > MAX_FL bytes, bad CRC (HW_ENET_MAC_RMON_R_JAB)	32	R	0000_0000h	26.4.72/1705
800F_02A8	ENET MAC RMON Rx 64 byte packets (HW_ENET_MAC_RMON_R_P64)	32	R	0000_0000h	26.4.73/1706
800F_02AC	ENET MAC RMON Rx 65 to 127 byte packets (HW_ENET_MAC_RMON_R_P65TO127)	32	R	0000_0000h	26.4.74/1706
800F_02B0	ENET MAC RMON Rx 128 to 255 byte packets (HW_ENET_MAC_RMON_R_P128TO255)	32	R	0000_0000h	26.4.75/1707
800F_02B4	ENET MAC RMON Rx 256 to 511 byte packets (HW_ENET_MAC_RMON_R_P256TO511)	32	R	0000_0000h	26.4.76/1707
800F_02B8	ENET MAC RMON Rx 512 to 1023 byte packets (HW_ENET_MAC_RMON_R_P512TO1023)	32	R	0000_0000h	26.4.77/1708
800F_02BC	ENET MAC RMON Rx 1024 to 2047 byte packets (HW_ENET_MAC_RMON_R_P1024TO2047)	32	R	0000_0000h	26.4.78/1708
800F_02C0	ENET MAC RMON Rx packets w > 2048 bytes (HW_ENET_MAC_RMON_R_P_GTE2048)	32	R	0000_0000h	26.4.79/1709
800F_02C4	ENET MAC RMON Rx Octets (HW_ENET_MAC_RMON_R_OCTETS)	32	R	0000_0000h	26.4.80/1709
800F_02C8	ENET MAC Frames Received count drop (HW_ENET_MAC_IEEE_R_DROP)	32	R	0000_0000h	26.4.81/1709
800F_02CC	ENET MAC Frames Received OK (HW_ENET_MAC_IEEE_R_FRAME_OK)	32	R	0000_0000h	26.4.82/1710
800F_02D0	ENET MAC Frames Received with CRC Error (HW_ENET_MAC_IEEE_R_CRC)	32	R	0000_0000h	26.4.83/1710
800F_02D4	ENET MAC Frames Received with Alignment Error (HW_ENET_MAC_IEEE_R_ALIGN)	32	R	0000_0000h	26.4.84/1711
800F_02D8	ENET MAC Frames Received overflow (HW_ENET_MAC_IEEE_R_MACERR)	32	R	0000_0000h	26.4.85/1711
800F_02DC	ENET MAC Frames Received flow control (HW_ENET_MAC_IEEE_R_FDXFC)	32	R	0000_0000h	26.4.86/1712
800F_02E0	ENET MAC Frames Received error (HW_ENET_MAC_IEEE_R_OCTETS_OK)	32	R	0000_0000h	26.4.87/1712
800F_0400	ENET MAC IEEE1588 Timer Control Register (HW_ENET_MAC_ATIME_CTRL)	32	R/W	0000_0000h	26.4.88/1713
800F_0404	ENET MAC IEEE1588 Timer value Register (HW_ENET_MAC_ATIME)	32	R/W	0000_0000h	26.4.89/1715
800F_0408	ENET MAC IEEE1588 Offsetvalue for one-shot event generation Register (HW_ENET_MAC_ATIME_EVT_OFFSET)	32	R/W	0000_0000h	26.4.90/1715
800F_040C	ENET MAC IEEE1588 Timer Period Register (HW_ENET_MAC_ATIME_EVT_PERIOD)	32	R/W	3B9A_CA00h	26.4.91/1716

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_0410	ENET MAC IEEE1588 Correction counter wrap around value Register (HW_ENET_MAC_ATIME_CORR)	32	R/W	0000_0000h	26.4.92/1716
800F_0414	ENET MAC IEEE1588 Clock period of the timestamping clock (ts_clk) in nanoseconds and correction increment Register (HW_ENET_MAC_ATIME_INC)	32	R/W	0000_0000h	26.4.93/1717
800F_0418	ENET MAC IEEE1588 Timestamp of the last Frame Register (HW_ENET_MAC_TS_TIMESTAMP)	32	R	0000_0000h	26.4.94/1718
800F_0500	ENET MAC Supplemental MAC Address 0 (HW_ENET_MAC_SMAC_0_0)	32	R/W	0000_0000h	26.4.95/1719
800F_0504	ENET MAC Supplemental MAC Address 0 (HW_ENET_MAC_SMAC_0_1)	32	R/W	0000_0000h	26.4.96/1719
800F_0508	ENET MAC Supplemental MAC Address 1 (HW_ENET_MAC_SMAC_1_0)	32	R/W	0000_0000h	26.4.97/1719
800F_050C	ENET MAC Supplemental MAC Address 1 (HW_ENET_MAC_SMAC_1_1)	32	R/W	0000_0000h	26.4.98/1720
800F_0510	ENET MAC Supplemental MAC Address 2 (HW_ENET_MAC_SMAC_2_0)	32	R/W	0000_0000h	26.4.99/1720
800F_0514	ENET MAC Supplemental MAC Address 2 (HW_ENET_MAC_SMAC_2_1)	32	R/W	0000_0000h	26.4.100/1721
800F_0518	ENET MAC Supplemental MAC Address 3 (HW_ENET_MAC_SMAC_3_0)	32	R/W	0000_0000h	26.4.101/1721
800F_051C	ENET MAC Supplemental MAC Address 3 (HW_ENET_MAC_SMAC_3_1)	32	R/W	0000_0000h	26.4.102/1722
800F_0600	ENET MAC Compare register 0 (HW_ENET_MAC_COMP_REG_0)	32	R/W	0000_0000h	26.4.103/1722
800F_0604	ENET MAC Compare register 1 (HW_ENET_MAC_COMP_REG_1)	32	R/W	0000_0000h	26.4.104/1723
800F_0608	ENET MAC Compare register 2 (HW_ENET_MAC_COMP_REG_2)	32	R/W	0000_0000h	26.4.105/1723
800F_060C	ENET MAC Compare register 3 (HW_ENET_MAC_COMP_REG_3)	32	R/W	0000_0000h	26.4.106/1724
800F_0640	ENET MAC Capture register 0 (HW_ENET_MAC_CAPT_REG_0)	32	R	0000_0000h	26.4.107/1724
800F_0644	ENET MAC Capture register 1 (HW_ENET_MAC_CAPT_REG_1)	32	R	0000_0000h	26.4.108/1725
800F_0648	ENET MAC Capture register 2 (HW_ENET_MAC_CAPT_REG_2)	32	R	0000_0000h	26.4.109/1725
800F_064C	ENET MAC Capture register 3 (HW_ENET_MAC_CAPT_REG_3)	32	R	0000_0000h	26.4.110/1726
800F_0680	ENET MAC IEEE1588 Interrupt register. (HW_ENET_MAC_CCB_INT)	32	R/W	0000_0000h	26.4.111/1726

HW_ENET memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_0684	ENET MAC IEEE1588 Interrupt enable mask register (HW_ENET_MAC_CCB_INT_MASK)	32	R/W	0000_0000h	26.4.112/1727

26.4.1 ENET MAC Interrupt Event Register (HW_ENET_MAC_EIR)

When an event occurs that sets a bit in HW_ENET_MAC_EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (HW_ENET_MAC_EIMR) is also set. Writing a 1 to an HW_ENET_MAC_EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

Address: HW_ENET_MAC_EIR – 800F_0000h base + 4h offset = 800F_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0	BABR	BABT	GRA	TXF	TXB	RXF	RXB	MII	EBERR	LC	RL	UN	PLR	WAKEUP	TS_AVAIL
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS_TIMER	RSRVD1														
W		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_EIR field descriptions

Field	Description
31 RSRVD0	Reserved bits. Write as 0.
30 BABR	Babbling receive error. This bit indicates a frame was received with length in excess of MAX_FL bytes.
29 BABT	Babbling transmit error. This bit indicates the transmitted frame length exceeds MAX_FL bytes. This condition usually caused by a frame that is too long placed into the transmit data buffer(s). Truncation does not occur.

HW_ENET_MAC_EIR field descriptions (continued)

Field	Description
28 GRA	Graceful stop complete. This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. Note: The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared (by writing 1) and the TX is still stopped, the bit will not become set again.
27 TXF	Transmit frame interrupt. This bit indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated (Signal dma_txf_int asserted).
26 TXB	Transmit buffer interrupt. This bit indicates a transmit buffer descriptor has been updated (Signal dma_txb_int asserted).
25 RXF	Receive frame interrupt. This bit indicates a frame has been received and the last corresponding buffer descriptor has been updated (Signal dma_rxf_int asserted).
24 RXB	Receive buffer interrupt. This bit indicates a receive buffer descriptor not the last in the frame has been updated (Signal dma_rxb_int asserted).
23 MII	MII interrupt. This bit indicates the MII has completed the data transfer requested.
22 EBERR	Ethernet bus error. This bit indicates a system bus error occurs when a DMA transaction is underway (Signal dma_eberr_int asserted). When the EBERR bit is set, ETHER_EN is cleared, halting frame processing by the MAC. When this occurs, software needs to insure proper actions (possibly resetting the system) to resume normal operation.
21 LC	Late collision. This bit indicates a collision occurs beyond the collision window (slot time) in half duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision retry limit. This bit indicates a collision occurs on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half duplex mode.
19 UN	Transmit FIFO underrun. This bit indicates the transmit FIFO became empty before the complete frame was transmitted. A bad CRC is appended to the frame fragment and the remainder of the frame is discarded.
18 PLR	Payload receive error. This bit indicates a frame was received with a payload length error.
17 WAKEUP	Node Wake Up Request Indication. Read only status bit to indicate that a Magic Packet has been detected. Will act only if ECR(MAGIC_ENA) is 1.
16 TS_AVAIL	Transmit Timestamp Available. Indicates that the timestamp of the last transmitted Timing Frame is available in the register TS_TIMESTAMP.
15 TS_TIMER	The adjustable timer reached the period or offset events.
14–0 RSRVD1	Reserved bits. Write as 0.

26.4.2 ENET MAC Interrupt Mask Register (HW_ENET_MAC_EIMR)

Address: HW_ENET_MAC_EIMR – 800F_0000h base + 8h offset = 800F_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EIMR (bit define is same with EIR register)																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_EIMR field descriptions

Field	Description
31–0 EIMR (bit define is same with EIR register)	Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is not set. • 0 The corresponding interrupt source is masked. • 1 The corresponding interrupt source is not masked (i.e. interrupt is enabled).

26.4.3 ENET MAC Receive Descriptor Active Register (HW_ENET_MAC_RDAR)

The Receive Descriptor Active Register (RDAR) is a command register, written by the user, indicating the receive descriptor ring is updated (empty receive buffers have been produced by the driver with the empty bit set)

Whenever the register is written, the RDAR bit is set. This is independent of the data actually written by the user. When set, the uDMA polls the receive descriptor ring and processes receive frames (provided ether_en is also set). Once the uDMA polls a receive descriptor whose empty bit is not set, the uDMA clears the RDAR bit and ceases receive descriptor ring polling until the user sets the bit again, signifying that additional descriptors are placed into the receive descriptor ring. The RDAR registers are cleared at reset and when ether_en transitions from asserted to de-asserted or when the ecr_reset is set.

Address: HW_ENET_MAC_RDAR – 800F_0000h base + 10h offset = 800F_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0							RDAR	RSRVD1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RDAR field descriptions

Field	Description
31–25 RSRVD0	Reserved bits. Write as 0.
24 RDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the uDMA when no additional empty descriptors remain in the receive ring. Also cleared when ether_en transitions from asserted to de-asserted.
23–0 RSRVD1	Reserved bits. Write as 0.

26.4.4 ENET MAC Transmit Descriptor Active Register (HW_ENET_MAC_TDAR)

The Transmit Descriptor Active Register (TDAR) is a command register which the user writes to indicate the transmit descriptor ring has been updated (transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor)

When the register is written, the TDAR bit is set. This value is independent of the data actually written by the user. When set, the uDMA polls the transmit descriptor ring and processes transmit frames (provided ether_en is also set). Once the uDMA polls a transmit descriptor that has a ready bit not set, the uDMA clears the TDAR bit and ceases transmit descriptor ring polling until the user sets the bit again to signify additional descriptors have been placed into the transmit descriptor ring. The TDAR register is cleared at reset, when ether_en transitions from asserted to de-asserted, or when the ecr_reset is set.

Address: HW_ENET_MAC_TDAR – 800F_0000h base + 14h offset = 800F_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0							TDAR	RSRVD1																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_TDAR field descriptions

Field	Description
31–25 RSRVD0	Reserved bits. Write as 0.
24 TDAR	Set to 1 when this register is written, regardless of the value written. Cleared by the uDMA when no additional ready descriptors remain in the transmit ring. Also cleared when ether_en transitions from asserted to de-asserted.
23–0 RSRVD1	Reserved bits. Write as 0.

26.4.5 ENET MAC Control Register (HW_ENET_MAC_ECR)

To clear any of the event bits within the Interrupt Event Register (EIR), the register must be written with a '1' in the corresponding bit position.

Address: HW_ENET_MAC_ECR – 800F_0000h base + 24h offset = 800F_0024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:7]									DBG_EN	RSRVD0	ENA_1588	SLEEP	MAGIC_ENA	ETHER_EN	RESET
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ECR field descriptions

Field	Description
31–7 RSRVD0	Reserved bits. Write as 0.
6 DBG_EN	Enables the debug input pin mac_freeze. When set, the input has an effect. When cleared (0, reset value) the input pin has no effect.
5 RSRVD0	Reserved bits. Write as 0.

HW_ENET_MAC_ECR field descriptions (continued)

Field	Description
4 ENA_1588	IEEE1588 Enable. Should be set to '1' to enable the Frame Time Stamping functions. Also drives the DMA control bit ena_1588.
3 SLEEP	Put controller in Sleep Mode. When asserted (Set to 1) the controller is configured in sleep mode. When set to 0 (Reset value) the controller is in normal operating mode.
2 MAGIC_ENA	Enable Magic Packet Detection. When set to 1, the controller detects Magic Packets and will assert the EIR (WAKEUP) bit when a frame is detected. When set to 0 (Reset value) the Magic Detection logic is disabled. Note: MAGIC_ENA is relevant only if the SLEEP bit is 1. If set to 1, changing the SLEEP bit will enable or disable both sleep mode and magic packet detection.
1 ETHER_EN	When this bit is set, MAC is enabled, and reception and transmission are possible. When this bit is cleared, reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. The buffer descriptor(s) for an aborted transmit frame are not updated after clearing this bit. When ETHER_EN is cleared, the DMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware alters the ETHER_EN bit under the following conditions: RESET is set by software, in which case ETHER_EN is cleared An error condition causes the EBERR bit to set, in which case ETHER_EN is cleared
0 RESET	The behavior of ENET-MAC when set this bit is depend on Switch mode. If disable switch, assert this bit will reset MAC and UDMA. If enable switch, assert this bit will only reset MAC. UDMA will be reset by software reset of switch.

26.4.6 ENET MAC MII Management Frame Register (HW_ENET_MAC_MMFR)

MDIO Management Register

Performing a write to the MMFR register triggers a management frame transaction to the PHY device unless the MSCR is programmed to 0. If the MSCR register is written to a non-zero value in the case of writing to MMFR when MSCR equals 0, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero. If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR(MII) interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

Address: HW_ENET_MAC_MMFR – 800F_0000h base + 40h offset = 800F_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ST		OP		PA					RA					TA		DATA															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_MMFR field descriptions

Field	Description
31–30 ST	Start of frame delimiter. These bits must be programmed to 0b01 for a valid MII management frame.
29–28 OP	Operation code: 00 Write frame operation, but not MII compliant. 01 Write frame operation for a valid MII management frame. 10 Read frame operation for a valid MII management frame. 11 Read frame operation, but not MII compliant.
27–23 PA	PHY address. This field specifies one of up to 32 attached PHY devices.
22–18 RA	Register address. This field specifies one of up to 32 registers within the specified PHY device.
17–16 TA	Turn around. This field must be programmed to '10' to generate a valid MII management frame.
15–0 DATA	Management frame data. This is the field for data to be written to or read from the PHY register.

26.4.7 ENET MAC MII Speed Control Register (HW_ENET_MAC_MSCR)

Address: HW_ENET_MAC_MSCR – 800F_0000h base + 44h offset = 800F_0044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:11]					HOLDTIME			DIS_PRE	MII_SPEED						RSRVD1
W	▨	▨	▨	▨	▨											▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_MSCR field descriptions

Field	Description
31–11 RSRVD0	Reserved bits. Write as 0.
10–8 HOLDTIME	The IEEE802.3 Clause 22 defines a minimum of 10ns for the holdtime on the MDIO output. Depending on the host bus frequency the setting may need to be increased. The following Bit 10:8: MDIO hold time setting: 000 : 1 pclk cycle (default) 001 : 2 pclk cycles 010 : 3 pclk cycles 011 : 4 pclk cycles 100 : 5 pclk cycles 101 : 6 pclk cycles 110 : 7 pclk cycles 111 : 8 pclk cycles
7 DIS_PRE	Asserting this bit causes preamble (32 1s) not to be prepended to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY device(s) does not require it.
6–1 MII_SPEED	MII_SPEED controls the frequency of the MII management interface clock (Signal mdc) relative to the internal bus clock (pclk). A value of 0 in this field turns off the mdc and leaves it in low voltage state. Any non-zero value results in the mdc frequency of 1/(MII_SPEED × 2) of the internal bus frequency (Clock pclk).
0 RSRVD1	Reserved bits. Write as 0.

26.4.8 ENET MAC MIB Control/Status Register (HW_ENET_MAC_MIBC)

Address: HW_ENET_MAC_MIBC – 800F_0000h base + 64h offset = 800F_0064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MIB_DIS	MIB_IDLE	MIB_CLEAR	RSRVD0[28:16]												
W				▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:0]															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_MIBC field descriptions

Field	Description
31 MIB_DIS	A read/write control bit. If set, the MIB logic halts and does not update any MIB counters.
30 MIB_IDLE	A read-only status bit. If set the MIB block is not currently updating any MIB counters.
29 MIB_CLEAR	A read/write control bit. If set all statistics counters are reset to 0.
28–0 RSRVD0	Reserved bits. Write as 0.

26.4.9 ENET MAC Receive Control Register (HW_ENET_MAC_RCR)

Address: HW_ENET_MAC_RCR – 800F_0000h base + 84h offset = 800F_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GRS	NO_LGTH_CHECK	MAX_FL													
W																
Reset	0	0	0	0	0	1	0	1	1	1	1	0	1	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CNTL_FRM_ENA	CRC_FWD	PAUSE_FWD	PAD_EN	RSRVD0	RSRVD0	RMII_10T	RMII_MODE	RSRVD0	RSRVD0	FCE	BC_REJ	PROM	MII_MODE	DRT	LOOP
W					▨	▨			▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_ENET_MAC_RCR field descriptions

Field	Description
31 GRS	Graceful receive stopped. Read-only status indicating that the MAC receive datapath is stopped
30 NO_LGTH_ CHECK	Payload Length Check Disable. When set to 1, the controller checks the frame's payload length with the Frame Length/Type field. When set to 0 (Reset value) the payload length check is disabled
29–16 MAX_FL	Maximum frame length. Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL causes the BABT interrupt to occur. Receive frames longer than MAX_FL causes the BABR interrupt to occur and sets the LG bit in the end of frame receive buffer descriptor. Set to 1518 (Decimal) after reset.

HW_ENET_MAC_RCR field descriptions (continued)

Field	Description
15 CNTL_FRM_ENA	MAC Control Frame Enable. When set to 1, MAC Control frames with any Opcode other than 0x01 (Pause Frame) are silently discarded. When set to 0 (Reset value), MAC Control frames with any Opcode other than 0x01 are accepted and forwarded to the Client interface.
14 CRC_FWD	Terminate / Forward Received CRC. If cleared (Reset value 0) the CRC field of received frames is transmitted to the user application. If set to 1 the CRC field is stripped from the frame. Note: If padding function is enabled (Bit PAD_EN set to 1), CRC_FWD is ignored and the CRC field is checked and always terminated and removed.
13 PAUSE_FWD	Terminate / Forward Pause Frames. If enabled (Set to 1) pause frames are forwarded to the user application. In normal mode (Set to reset value 0) pause frames are terminated and discarded in the MAC.
12 PAD_EN	Enable / Disable Frame Padding Remove on receive. If enabled (Set to 1) padding is removed from received frames. If disabled (set to reset value 0) no padding is removed on receive by the MAC.
11 RSRVD0	Reserved bits. Write as 0.
10 RSRVD0	Reserved bits. Write as 0.
9 RMII_10T	RMII 10-Base T. Enables 10Mbps mode of the RMII. When set to 1, the controller signal set_10 is set to 1, when set to 0 set_10 is set to 0.
8 RMII_MODE	RMII Mode Enable. Indicates, when ECR(ETH_SPEED) is set to 0, if the MAC is in RMII or MII. 0 MAC configured for MII mode. 1 MAC configured for RMII operation. When set to 1, the controller signal ena_rmii is set to 1, when set to 0, ena_rmii is set to 0.
7 RSRVD0	Reserved bits. Write as 0.
6 RSRVD0	Reserved bits. Write as 0.
5 FCE	Flow control enable. If asserted, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter will stop transmitting data frames for a given duration.
4 BC_REJ	Broadcast frame reject. If asserted, frames with DA (destination address) equals 0xFFFFFFFFFFFF are rejected unless the PROM bit is set. If both BC_REJ and PROM equals 1, frames with broadcast DA are accepted and the M (MISS) is set in the receive buffer descriptor.
3 PROM	Promiscuous mode. All frames are accepted regardless of address matching.
2 MII_MODE	Media independent interface mode. Should always be set to 1, setting MII_MODE to 0 has no effect.
1 DRT	Disable receive on transmit. 0 Receive path operates independently of transmit (use for full duplex or to monitor transmit activity in half duplex mode). 1 Disable reception of frames while transmitting (normally used for half duplex mode).

HW_ENET_MAC_RCR field descriptions (continued)

Field	Description
0 LOOP	Internal loopback. If set, transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. When asserted the controller signal ena_loop is set to 1.

26.4.10 ENET MAC Transmit Control Register (HW_ENET_MAC_TCR)

Address: HW_ENET_MAC_TCR – 800F_0000h base + C4h offset = 800F_00C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:10]						TX_CRC_FWD	TX_ADDR_INS	TX_ADDR_SEL			RFC_PAUSE	TFC_PAUSE	FEDN	RSRVD0	GTS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_TCR field descriptions

Field	Description
31–10 RSRVD0	Reserved bits. Write as 0.
9 TX_CRC_FWD	Forward frame from application with CRC. When set (1) the transmitter will not append any CRC to transmitted frames as it is expecting a frame with crc from the application. When cleared (0, default) the toplevel input pin ff_tx_crc_fwd controls if the frame has a crc from the application (1) or not (0) (i.e. the register bit is OR'ed with ff_tx_crc_fwd input).
8 TX_ADDR_INS	Set MAC address on transmit. If enabled (Set to 1) the MAC overwrites the source MAC address with the programmed MAC address according to TX_ADDR_SEL. If disabled (Set to reset value 0), the source MAC address is not modified by the MAC.
7–5 TX_ADDR_SEL	Source MAC address select on transmit. If register TX_ADDR_INS is set to 1 the MAC address that is used to overwrite the source MAC address: 000: Node MAC Address programmed on PADDR1/2 registers is used as the source address. 100: Supplemental MAC Address 0 programmed on SMAC_0_0 and SMAC_0_1 registers is used as the source address.

HW_ENET_MAC_TCR field descriptions (continued)

Field	Description
	101: Supplemental MAC Address 1 programmed on SMAC_1_0 and SMAC_1_1 registers is used as the source address. 110: Supplemental MAC Address 2 programmed on SMAC_2_0 and SMAC_2_1 registers is used as the source address. other (any value other than above): Supplemental MAC Address 3 programmed on SMAC_3_0 and SMAC_3_1 registers is used as the source address.
4 RFC_PAUSE	Receive frame control pause. This read-only status bit is asserted when a full duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This bit automatically clears when the pause duration is complete.
3 TFC_PAUSE	Transmit frame control pause. Transmits a PAUSE frame when asserted. When this bit is set, the MAC stops transmission of data frames after the current transmission is complete. At this time, GRA interrupt in the EIR register is asserted. With transmission of data frames stopped, MAC transmits a MAC Control PAUSE frame. Next, the MAC clears the TFC_PAUSE bit and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC Control PAUSE frame.
2 FEDN	Full duplex enable. If set, frames transmit independent of carrier sense and collision inputs. This bit should only be modified when ETHER_EN is cleared.
1 RSRVD0	Reserved bits. Write as 0.
0 GTS	Graceful transmit stop. When this bit is set, MAC stops transmission after any frame currently transmitted is complete and GRA interrupt in the EIR register is asserted. If frame transmission is not currently underway, the GRA interrupt will be asserted immediately. Once transmission has completed, a restart can accomplish by clearing the GTS bit. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS equals 1, transmission stops after the collision. The frame is transmitted again once GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear the ECR register bit ETHER_EN following the GRA interrupt.

26.4.11 ENET MAC Physical Address Lower Register (HW_ENET_MAC_PALR)

Address: HW_ENET_MAC_PALR – 800F_0000h base + E4h offset = 800F_00E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_PALR field descriptions

Field	Description
31–0 PADDR1	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

26.4.12 ENET MAC Physical Address Upper Register (HW_ENET_MAC_PAUR)

Address: HW_ENET_MAC_PAUR – 800F_0000h base + E8h offset = 800F_00E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PADDR2																TYPE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

HW_ENET_MAC_PAUR field descriptions

Field	Description
31–16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
15–0 TYPE	Type field in PAUSE frames. These 16-bits are a constant value of 0x8808 (not writeable).

26.4.13 ENET MAC Opcode/Pause Duration Register (HW_ENET_MAC_OPD)

Address: HW_ENET_MAC_OPD – 800F_0000h base + ECh offset = 800F_00ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OPCODE																PAUSE_DUR															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_OPD field descriptions

Field	Description
31–16 OPCODE	Opcode field used in PAUSE frames. These bits are a constant, 0x01 (not writeable).
15–0 PAUSE_DUR	Pause Duration field used in PAUSE frames.

26.4.14 ENET MAC Descriptor Individual Upper Address Register (HW_ENET_MAC_IAUR)

Address: HW_ENET_MAC_IAUR – 800F_0000h base + 118h offset = 800F_0118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR1																															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IAUR field descriptions

Field	Description
31–0 IADDR1	The upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

26.4.15 ENET MAC Descriptor Individual Lower Address Register (HW_ENET_MAC_IALR)

Address: HW_ENET_MAC_IALR – 800F_0000h base + 11Ch offset = 800F_011Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR2																															
Re- set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IALR field descriptions

Field	Description
31–0 IADDR2	The lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

26.4.16 ENET MAC Descriptor Group Upper Address Register (HW_ENET_MAC_GAUR)

Address: HW_ENET_MAC_GAUR – 800F_0000h base + 120h offset = 800F_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_GAUR field descriptions

Field	Description
31–0 GADDR1	The GADDR1 register contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

26.4.17 ENET MAC Descriptor Group Lower Address Register (HW_ENET_MAC_GALR)

Address: HW_ENET_MAC_GALR – 800F_0000h base + 124h offset = 800F_0124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_GALR field descriptions

Field	Description
31–0 GADDR2	The GADDR2 register contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

26.4.18 ENET MAC Transmit FIFO Watermark and Store and Forward Control Register (HW_ENET_MAC_TFW_SFCR)

Address: HW_ENET_MAC_TFW_SFCR – 800F_0000h base + 144h offset = 800F_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:9]							STR_ FWD	RSRVD1		TFWR					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_TFW_SFCR field descriptions

Field	Description
31–9 RSRVD0	Reserved bits. Write as 0.
8 STR_FWD	Store and Forward Enable. When set to 1, Store and Forward is enabled, when set to 0 (Reset value), the transmission start threshold is programmed with the TFWR bits.
7–6 RSRVD1	Reserved bits. Write as 0.
5–0 TFWR	When STR_FWD is set to 0, number of bytes, in steps of 64 Bytes, written to transmit FIFO before transmission of a frame begins: 000000 64 bytes written 000001 64 bytes written 000010 128 bytes written 000011 192 bytes written 000100 256 bytes written 000101 320 bytes written 000110 384 bytes written ...... 111110 3968 bytes written 111111 4032 bytes written Note: if a frame with less than the threshold is written it will still be sent, independently of this threshold setting. The threshold is only relevant if the frame is larger than the threshold given.

26.4.19 ENET MAC FIFO Receive Bound Register (HW_ENET_MAC_FRBR)

Address: HW_ENET_MAC_FRBR – 800F_0000h base + 14Ch offset = 800F_014Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:10]						R_BOUND								RSRVD1	
W	▨														▨	
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_FRBR field descriptions

Field	Description
31–10 RSRVD0	Reserved bits. Write as 0.
9–2 R_BOUND	Read-only. Highest valid FIFO RAM address. Set to 01100000 to provide software compatibility with existing devices.
1–0 RSRVD1	Reserved bits. Write as 0.

26.4.20 ENET MAC FIFO Receive FIFO Start Register (HW_ENET_MAC_FRSR)

Address: HW_ENET_MAC_FRSR – 800F_0000h base + 150h offset = 800F_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:11]					RSRVD1	R_FSTART								RSRVD2	
W	▨					▨									▨	
Reset	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0

HW_ENET_MAC_FRSR field descriptions

Field	Description
31–11 RSRVD0	Reserved bits. Write as 0.
10 RSRVD1	Must be set. Not used, implemented for software compatibility.
9–2 R_FSTART	Address of first receive FIFO location. Acts as delimiter between receive and transmit FIFOs. For proper operation, ensure that R_FSTART is set to 0x48 or greater. Read/write: Not used, implemented for software compatibility.
1–0 RSRVD2	Reserved bits. Write as 0.

26.4.21 ENET MAC Pointer to Receive Descriptor Ring Register (HW_ENET_MAC_ERDSR)

The user writes the ERDSR. It provides a pointer to the start of the circular receive buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however it is recommended it be made 128-bit aligned (evenly divisible by 16).The user should write bits 1 and 0 to 0. Hardware ignores non-zero values in these two bit positions.

Address: HW_ENET_MAC_ERDSR – 800F_0000h base + 180h offset = 800F_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	R_DES_START[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R_DES_START[15:2]														RSRVD0	
W															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ERDSR field descriptions

Field	Description
31–2 R_DES_START	Pointer to start of receive buffer descriptor queue.
1–0 RSRVD0	Reserved bits. Write as 0.

26.4.22 ENET MAC Pointer to Transmit Descriptor Ring Register (HW_ENET_MAC_ETDSR)

The user writes the ETDSR. It provides a pointer to the start of the circular transmit buffer descriptor queue in external memory. This pointer must be 32-bit aligned; however, it is recommended it be made 128-bit aligned (evenly divisible by 16). The user should write bits 1 and 0 to 0. Hardware ignores non-zero values in these two bit positions.

Address: HW_ENET_MAC_ETDSR – 800F_0000h base + 184h offset = 800F_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	X_DES_START[31:16]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	X_DES_START[15:2]														RSRVD0	
W															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ETDSR field descriptions

Field	Description
31–2 X_DES_START	Pointer to start of transmit buffer descriptor queue.
1–0 RSRVD0	Reserved bits. Write as 0.

26.4.23 ENET MAC Maximum Receive Buffer Size Register (HW_ENET_MAC_EMRBR)

The EMRBR is a user-programmable register that dictates the maximum size of all receive buffers. Note that because receive frames are truncated at 2k-15bytes, only bits 10-4 are used. This value should take into consideration that the receive CRC is always written into the last receive buffer. To allow on maximum size frame per buffer, EMRBR must be set to RCR[MAX_FL] or larger. The EMRBR must be evenly divisible by 16. To ensure this, bits 3-0 are forced low. To minimize bus utilization (descriptor fetches), it is recommended that EMRBR be greater than or equal to 256 bytes.

Address: HW_ENET_MAC_EMRBR – 800F_0000h base + 188h offset = 800F_0188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																					R_BUF_SIZE							RSRVD1			
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_EMRBR field descriptions

Field	Description
31–11 RSRVD0	Reserved bits. Write as 0.
10–4 R_BUF_SIZE	Receive buffer size in bytes. 0x00 0 bytes 0x01 16 bytes 0x02 32 bytes ... 0x7F 2032 bytes
3–0 RSRVD1	Reserved bits. Write as 0.

26.4.24 ENET MAC Receive FIFO Section Full Threshold Register (HW_ENET_MAC_RX_SECTION_FULL)

Address: HW_ENET_MAC_RX_SECTION_FULL – 800F_0000h base + 190h offset = 800F_0190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								RX_SECTION_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RX_SECTION_FULL field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 RX_SECTION_ FULL	Value, in 64-Bit words, of the Receive FIFO section full threshold.

26.4.25 ENET MAC Receive FIFO Section Empty Threshold Register (HW_ENET_MAC_RX_SECTION_EMPTY)

Address: HW_ENET_MAC_RX_SECTION_EMPTY – 800F_0000h base + 194h offset = 800F_0194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								RX_SECTION_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RX_SECTION_EMPTY field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 RX_SECTION_ EMPTY	Value, in 64-Bit words, of the Receive FIFO section empty threshold.

26.4.26 ENET MAC Receive FIFO Almost Empty Threshold Register (HW_ENET_MAC_RX_ALMOST_EMPTY)

Address: HW_ENET_MAC_RX_ALMOST_EMPTY – 800F_0000h base + 198h offset = 800F_0198h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								RX_ALMOST_EMPTY							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_ENET_MAC_RX_ALMOST_EMPTY field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 RX_ALMOST_ EMPTY	Value, in 64-Bit words, of the Receive FIFO almost empty threshold.

26.4.27 ENET MAC Receive FIFO Almost Full Thresholdt Register (HW_ENET_MAC_RX_ALMOST_FULL)

Address: HW_ENET_MAC_RX_ALMOST_FULL – 800F_0000h base + 19Ch offset = 800F_019Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								RX_ALMOST_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_ENET_MAC_RX_ALMOST_FULL field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 RX_ALMOST_ FULL	Value, in 64-Bit words, of the Receive FIFO almost full threshold.

26.4.28 ENET MAC Transmit FIFO Section Empty Threshold Register (HW_ENET_MAC_TX_SECTION_EMPTY)

Address: HW_ENET_MAC_TX_SECTION_EMPTY – 800F_0000h base + 1A0h offset = 800F_01A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								TX_SECTION_EMPTY							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_TX_SECTION_EMPTY field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 TX_SECTION_ EMPTY	Value, in 64-Bit words, of the Transmit FIFO section empty threshold.

26.4.29 ENET MAC Transmit FIFO Almost Empty Threshold Register (HW_ENET_MAC_TX_ALMOST_EMPTY)

Address: HW_ENET_MAC_TX_ALMOST_EMPTY – 800F_0000h base + 1A4h offset = 800F_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								TX_ALMOST_EMPTY							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

HW_ENET_MAC_TX_ALMOST_EMPTY field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 TX_ALMOST_ EMPTY	Value, in 64-Bit words, of the Transmit FIFO almost empty threshold.

26.4.30 ENET MAC Transmit FIFO Almost Full Threshold Register (HW_ENET_MAC_TX_ALMOST_FULL)

Address: HW_ENET_MAC_TX_ALMOST_FULL – 800F_0000h base + 1A8h offset = 800F_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								TX_ALMOST_FULL							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

HW_ENET_MAC_TX_ALMOST_FULL field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 TX_ALMOST_FULL	Value, in 64-Bit words, of the Transmit FIFO almost full threshold. The number of bits, K, is Log2 of the Transmit FIFO depth. A minimum value of 6 is required to have correct ff_tx_rdy deassertion before the fifo overflows. If a lower value is set, the FIFO will overflow when ff_tx_rdy is deasserted, hence there would be no time for the application to react properly. A recommended value of at least 8 should be set allowing a latency of 2 clock cycles to the application. If more latency is required the value can be increased as necessary (latency=TX_ALMOST_FULL-5). Note that a FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHER_EN.

26.4.31 ENET MAC Transmit Inter-Packet Gap Register (HW_ENET_MAC_TX_IPG_LENGTH)

Address: HW_ENET_MAC_TX_IPG_LENGTH – 800F_0000h base + 1ACh offset = 800F_01ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																											TX_IPG_LENGTH				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0

HW_ENET_MAC_TX_IPG_LENGTH field descriptions

Field	Description
31–5 RSRVD0	Reserved bits. Write as 0.

HW_ENET_MAC_TX_IPG_LENGTH field descriptions (continued)

Field	Description
4–0 TX_IPG_ LENGTH	Transmit Inter Packet Gap. Set, in Bytes, the IPG between transmitted Frames. Can be set to any value between 8 and 27. If set to a Value below 8, the IPG is 8, if set to any value above 27, the IPG is 27.

26.4.32 ENET MAC Frame Truncation Length Register (HW_ENET_MAC_TRUNC_FL)

Address: HW_ENET_MAC_TRUNC_FL – 800F_0000h base + 1B0h offset = 800F_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																		TRUNC_FL													
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

HW_ENET_MAC_TRUNC_FL field descriptions

Field	Description
31–14 RSRVD0	Reserved bits. Write as 0.
13–0 TRUNC_FL	Frame Truncation Length. Set the value from which a receive Frame is truncated (if greater than this value). Should be a value greater or equal to RCR(MAX_FL). Note: Truncation happens at TRUNC_FL, however when truncation occured, the application (FIFO) may receive less data, guaranteeing that it will never receive more than the set limit.

26.4.33 ENET MAC Accelerator Transmit Function Configuration Register (HW_ENET_MAC_IPACCTXCONF)

Transmit Accelerator function configuration. Control accelerator actions to be performed when sending frames. The register can be changed before or after each frame, but must stay unmodified during frame write into the transmit FIFO.

Address: HW_ENET_MAC_IPACCTXCONF – 800F_0000h base + 1C0h offset = 800F_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:5]											TX_PROTCHK_INS	TX_IPCHK_INS	RSRVD1		SHIFT16
W	▨													▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IPACCTXCONF field descriptions

Field	Description
31–5 RSRVD0	Reserved bits. Write as 0.
4 TX_PROTCHK_INS	Enable insertion of protocol checksum. If enabled (1) and an IP frame with a known protocol is transmitted, the checksum will be inserted automatically into the frame. The checksum field should be all zero. Other frames are not modified. The setting is OR'ed with the pin ff_tx_protchk_ins.
3 TX_IPCHK_INS	Enable insertion of IP header checksum. If enabled (1) and an IP frame is transmitted, its checksum will be inserted automatically. The IP header checksum field should be all zero. If a non-IP frame is transmitted the frame will not be modified. The setting is OR'ed with the pin ff_tx_ipchk_ins.
2–1 RSRVD1	Reserved bits. Write as 0.
0 SHIFT16	Enable TX FIFO Shift16 function. Indicates to the transmit data FIFO, that the frame will be written with 2 additional octets before the frame data. This means the actual Frame starts at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory as the 14-byte Ethernet header is extended to a 16 byte header.

26.4.34 ENET MAC Accelerator Receive Function Configuration Register (HW_ENET_MAC_IPACCRXCONF)

Address: HW_ENET_MAC_IPACCRXCONF – 800F_0000h base + 1C4h offset = 800F_01C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:8]								SHIFT16	RX_LINEERR_DISC	RSRVD1			RX_PROTERR_DISCARD	RX_IPERR_DISCARD	RX_IP_PAD_REMOVE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IPACCRXCONF field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7 SHIFT16	If set 1, instructs the MAC to write 2 additional bytes in front of each frame received into the RX FIFO. The actual frame data then starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. Note: If this function only affects the FIFO storage and has no influence on the statistics, which still use the actual length of the frame received.
6 RX_LINEERR_ DISC	Enable discard of frames with MAC layer errors. If set any frame received with a CRC, length or PHY error is automatically discarded and not forwarded to the user application interface. See 12.4 page 55.
5–3 RSRVD1	Reserved bits. Write as 0.
2 RX_PROTERR_ DISCARD	Enable discard of frames with wrong protocol checksum. If set and TCP/IP, UDP/IP or ICMP/IP frame is received that has a wrong TCP or UDP or ICMP checksum the frame is discarded. Discarding is only available when the RX FIFO operates in Store and Forward mode.
1 RX_IPERR_ DISCARD	Enable discard of frames with wrong IPv4 header checksum. If set and an IPv4 frame is received with a mismatching header checksum, the frame will be discarded. IPv6 has no header checksum and will therefore not be affected by this setting. Discarding is only available when the RX FIFO operates in Store and Forward mode.
0 RX_IP_PAD_ REMOVE	Enable padding removal for short IP frames. If set any bytes following the IP payload section of the frame are removed from the frame.

26.4.35 ENET MAC RMON Tx packet drop (HW_ENET_MAC_RMON_T_DROP)

Count of frames not counted correctly

Note: Counter not implemented (read 0 always) as not applicable

Address: HW_ENET_MAC_RMON_T_DROP – 800F_0000h base + 200h offset = 800F_0200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_DROP																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_DROP field descriptions

Field	Description
31–0 RMON_T_DROP	Reserved (this counter not implemented)

26.4.36 ENET MAC RMON Tx packet count (HW_ENET_MAC_RMON_T_PACKETS)

Address: HW_ENET_MAC_RMON_T_PACKETS – 800F_0000h base + 204h offset = 800F_0204h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_PACKETS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_PACKETS field descriptions

Field	Description
31–0 RMON_T_PACKETS	ENET MAC RMON Tx packet count

26.4.37 ENET MAC RMON Tx Broadcast Packets (HW_ENET_MAC_RMON_T_BC_PKT)

Address: HW_ENET_MAC_RMON_T_BC_PKT – 800F_0000h base + 208h offset = 800F_0208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_BC_PKT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_BC_PKT field descriptions

Field	Description
31–0 RMON_T_BC_ PKT	ENET MAC RMON Tx Broadcast Packets count

26.4.38 ENET MAC RMON Tx Multicast Packets (HW_ENET_MAC_RMON_T_MC_PKT)

Address: HW_ENET_MAC_RMON_T_MC_PKT – 800F_0000h base + 20Ch offset = 800F_020Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_MC_PKT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_MC_PKT field descriptions

Field	Description
31–0 RMON_T_MC_ PKT	Count of ENET MAC RMON Tx Multicast Packets

26.4.39 ENET MAC RMON Tx Packets w CRC/Align error (HW_ENET_MAC_RMON_T_CRC_ALIGN)

Address: HW_ENET_MAC_RMON_T_CRC_ALIGN – 800F_0000h base + 210h offset = 800F_0210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_CRC_ALIGN																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_CRC_ALIGN field descriptions

Field	Description
31–0 RMON_T_CRC_ ALIGN	Count of ENET MAC RMON Tx Packets w CRC/Align error

26.4.40 ENET MAC RMON Tx Packets < 64 bytes, good CRC (HW_ENET_MAC_RMON_T_UNDERSIZE)

Address: HW_ENET_MAC_RMON_T_UNDERSIZE – 800F_0000h base + 214h offset = 800F_0214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_UNDERSIZE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_UNDERSIZE field descriptions

Field	Description
31–0 RMON_T_ UNDERSIZE	Count of ENET MAC RMON Tx Packets < 64 bytes, good CRC

26.4.41 ENET MAC RMON Tx Packets > MAX_FL bytes, good CRC (HW_ENET_MAC_RMON_T_OVERSIZE)

Address: HW_ENET_MAC_RMON_T_OVERSIZE – 800F_0000h base + 218h offset = 800F_0218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_OVERSIZE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_OVERSIZE field descriptions

Field	Description
31–0 RMON_T_ OVERSIZE	Count of ENET MAC RMON Tx Packets > MAX_FL bytes, good CRC

26.4.42 ENET MAC RMON Tx Packets < 64 bytes, bad CRC (HW_ENET_MAC_RMON_T_FRAG)

Address: HW_ENET_MAC_RMON_T_FRAG – 800F_0000h base + 21Ch offset = 800F_021Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_FRAG																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_FRAG field descriptions

Field	Description
31–0 RMON_T_FRAG	Count of ENET MAC RMON Tx Packets < 64 bytes, bad CRC

26.4.43 ENET MAC RMON Tx Packets > MAX_FL bytes, bad CRC (HW_ENET_MAC_RMON_T_JAB)

Address: HW_ENET_MAC_RMON_T_JAB – 800F_0000h base + 220h offset = 800F_0220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_JAB																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_JAB field descriptions

Field	Description
31–0 RMON_T_JAB	Count of ENET MAC RMON Tx Packets > MAX_FL bytes, bad CRC

26.4.44 ENET MAC RMON Tx collision count (HW_ENET_MAC_RMON_T_COL)

Address: HW_ENET_MAC_RMON_T_COL – 800F_0000h base + 224h offset = 800F_0224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_COL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_COL field descriptions

Field	Description
31–0 RMON_T_COL	ENET MAC RMON Tx collision count

26.4.45 ENET MAC RMON Tx 64 byte packets (HW_ENET_MAC_RMON_T_P64)

Address: HW_ENET_MAC_RMON_T_P64 – 800F_0000h base + 228h offset = 800F_0228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P64																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P64 field descriptions

Field	Description
31–0 RMON_T_P64	Count of ENET MAC RMON Tx 64 byte packets

26.4.46 ENET MAC RMON Tx 65 to 127 byte packets (HW_ENET_MAC_RMON_T_P65TO127N)

Address: HW_ENET_MAC_RMON_T_P65TO127N – 800F_0000h base + 22Ch offset = 800F_022Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P65TO127N																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P65TO127N field descriptions

Field	Description
31–0 RMON_T_ P65TO127N	Count of ENET MAC RMON Tx 65 to 127 byte packets

26.4.47 ENET MAC RMON Tx 128 to 255 byte packets (HW_ENET_MAC_RMON_T_P128TO255N)

ENET MAC RMON Tx 128 to 255 byte packets

Address: HW_ENET_MAC_RMON_T_P128TO255N – 800F_0000h base + 230h offset = 800F_0230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P128TO255N																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P128TO255N field descriptions

Field	Description
31–0 RMON_T_ P128TO255N	Count of ENET MAC RMON Tx 128 to 255 byte packets

26.4.48 ENET MAC RMON Tx 256 to 511 byte packets (HW_ENET_MAC_RMON_T_P256TO511)

Address: HW_ENET_MAC_RMON_T_P256TO511 – 800F_0000h base + 234h offset = 800F_0234h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P256TO511																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P256TO511 field descriptions

Field	Description
31–0 RMON_T_ P256TO511	Count of ENET MAC RMON Tx 256 to 511 byte packets

26.4.49 ENET MAC RMON Tx 512 to 1023 byte packets (HW_ENET_MAC_RMON_T_P512TO1023)

Address: HW_ENET_MAC_RMON_T_P512TO1023 – 800F_0000h base + 238h offset = 800F_0238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P512TO1023																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P512TO1023 field descriptions

Field	Description
31–0 RMON_T_ P512TO1023	Count of ENET MAC RMON Tx 512 to 1023 byte packets

26.4.50 ENET MAC RMON Tx 1024 to 2047 byte packets (HW_ENET_MAC_RMON_T_P1024TO2047)

Address: HW_ENET_MAC_RMON_T_P1024TO2047 – 800F_0000h base + 23Ch offset = 800F_023Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P1024TO2047																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P1024TO2047 field descriptions

Field	Description
31–0 RMON_T_ P1024TO2047	Count of ENET MAC RMON Tx 1024 to 2047 byte packets

26.4.51 ENET MAC RMON Tx packets w > 2048 bytes (HW_ENET_MAC_RMON_T_P_GTE2048)

Address: HW_ENET_MAC_RMON_T_P_GTE2048 – 800F_0000h base + 240h offset = 800F_0240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_P_GTE2048																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_P_GTE2048 field descriptions

Field	Description
31–0 RMON_T_P_ GTE2048	Count of ENET MAC RMON Tx packets w > 2048 bytes

26.4.52 ENET MAC RMON Tx Octets (HW_ENET_MAC_RMON_T_OCTETS)

Address: HW_ENET_MAC_RMON_T_OCTETS – 800F_0000h base + 244h offset = 800F_0244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_T_OCTETS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_T_OCTETS field descriptions

Field	Description
31–0 RMON_T_OCTETS	Count of ENET MAC RMON Tx Octets

26.4.53 ENET MAC Frames Transmitted count drop (HW_ENET_MAC_IEEE_T_DROP)

Count of frames not counted correctly

Note: Counter not implemented (read 0 always) as not appl

Address: HW_ENET_MAC_IEEE_T_DROP – 800F_0000h base + 248h offset = 800F_0248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_DROP																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_DROP field descriptions

Field	Description
31–0 IEEE_T_DROP	Reserved (this counter not implemented)

26.4.54 ENET MAC Frames Transmitted OK (HW_ENET_MAC_IEEE_T_FRAME_OK)

Address: HW_ENET_MAC_IEEE_T_FRAME_OK – 800F_0000h base + 24Ch offset = 800F_024Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_FRAME_OK																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_FRAME_OK field descriptions

Field	Description
31–0 IEEE_T_FRAME_OK	Count of ENET MAC Frames Transmitted OK

26.4.55 ENET MAC Frames Transmitted with Single Collision (HW_ENET_MAC_IEEE_T_1COL)

Address: HW_ENET_MAC_IEEE_T_1COL – 800F_0000h base + 250h offset = 800F_0250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_1COL																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_1COL field descriptions

Field	Description
31–0 IEEE_T_1COL	Count of ENET MAC Frames Transmitted with Single Collision

26.4.56 ENET MAC Frames Transmitted with Multiple Collisions (HW_ENET_MAC_IEEE_T_MCOL)

Address: HW_ENET_MAC_IEEE_T_MCOL – 800F_0000h base + 254h offset = 800F_0254h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_MCOL																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_MCOL field descriptions

Field	Description
31–0 IEEE_T_MCOL	Count of ENET MAC Frames Transmitted with Multiple Collisions

26.4.57 ENET MAC Frames Transmitted after Deferral Delay (HW_ENET_MAC_IEEE_T_DEF)

Address: HW_ENET_MAC_IEEE_T_DEF – 800F_0000h base + 258h offset = 800F_0258h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_DEF																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_DEF field descriptions

Field	Description
31–0 IEEE_T_DEF	ENET MAC Frame count Transmitted after Deferral Delay

26.4.58 ENET MAC Frames Transmitted with Late Collision (HW_ENET_MAC_IEEE_T_LCOL)

Address: HW_ENET_MAC_IEEE_T_LCOL – 800F_0000h base + 25Ch offset = 800F_025Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_LCOL																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_LCOL field descriptions

Field	Description
31–0 IEEE_T_LCOL	Count of ENET MAC Frames Transmitted with Late Collision

26.4.59 ENET MAC Frames Transmitted with Excessive Collisions (HW_ENET_MAC_IEEE_T_EXCOL)

Address: HW_ENET_MAC_IEEE_T_EXCOL – 800F_0000h base + 260h offset = 800F_0260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_EXCOL																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_EXCOL field descriptions

Field	Description
31–0 IEEE_T_EXCOL	Count of ENET MAC Frames Transmitted with Excessive Collisions

26.4.60 ENET MAC Frames Transmitted with Tx FIFO Underrun (HW_ENET_MAC_IEEE_T_MACERR)

Address: HW_ENET_MAC_IEEE_T_MACERR – 800F_0000h base + 264h offset = 800F_0264h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_MACERR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_MACERR field descriptions

Field	Description
31–0 IEEE_T_MACERR	Count of ENET MAC Frames Transmitted with Tx FIFO Underrun

26.4.61 ENET MAC Frames Transmitted with Carrier Sense Error (HW_ENET_MAC_IEEE_T_CSERR)

Address: HW_ENET_MAC_IEEE_T_CSERR – 800F_0000h base + 268h offset = 800F_0268h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_CSERR																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_CSERR field descriptions

Field	Description
31–0 IEEE_T_CSERR	Count of ENET MAC Frames Transmitted with Carrier Sense Error

26.4.62 ENET MAC Frames Transmitted with SQE Error (HW_ENET_MAC_IEEE_T_SQE)

Note: Counter not implemented (read 0 always) as no SQE information is available

Address: HW_ENET_MAC_IEEE_T_SQE – 800F_0000h base + 26Ch offset = 800F_026Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_SQE																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_SQE field descriptions

Field	Description
31–0 IEEE_T_SQE	Count of ENET MAC Frames Transmitted with SQE Error

26.4.63 ENET MAC Frames Transmitted flow control (HW_ENET_MAC_IEEE_T_FDXFC)

Flow Control Pause frames transmitted

Address: HW_ENET_MAC_IEEE_T_FDXFC – 800F_0000h base + 270h offset = 800F_0270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_FDXFC																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_FDXFC field descriptions

Field	Description
31–0 IEEE_T_FDXFC	Count of ENET MAC Frames Transmitted flow control

26.4.64 ENET MAC Frames Transmitted error (HW_ENET_MAC_IEEE_T_OCTETS_OK)

Octet count for Frames Transmitted w/o Error

Note: counts total octets (includes header and FCS fiel

Address: HW_ENET_MAC_IEEE_T_OCTETS_OK – 800F_0000h base + 274h offset = 800F_0274h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_T_OCTETS_OK																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_T_OCTETS_OK field descriptions

Field	Description
31–0 IEEE_T_ OCTETS_OK	Octet count for Frames Transmitted w/o Error

26.4.65 ENET MAC RMON Rx packet count (HW_ENET_MAC_RMON_R_PACKETS)

Address: HW_ENET_MAC_RMON_R_PACKETS – 800F_0000h base + 284h offset = 800F_0284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_PACKETS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_PACKETS field descriptions

Field	Description
31–0 RMON_R_ PACKETS	ENET MAC RMON Rx packet count

26.4.66 ENET MAC RMON Rx Broadcast Packets (HW_ENET_MAC_RMON_R_BC_PKT)

Address: HW_ENET_MAC_RMON_R_BC_PKT – 800F_0000h base + 288h offset = 800F_0288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_BC_PKT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_BC_PKT field descriptions

Field	Description
31–0 RMON_R_BC_ PKT	Count of ENET MAC RMON Rx Broadcast Packets

26.4.67 ENET MAC RMON Rx Multicast Packets (HW_ENET_MAC_RMON_R_MC_PKT)

Address: HW_ENET_MAC_RMON_R_MC_PKT – 800F_0000h base + 28Ch offset = 800F_028Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_MC_PKT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_MC_PKT field descriptions

Field	Description
31–0 RMON_R_MC_ PKT	Count of ENET MAC RMON Rx Multicast Packets

26.4.68 ENET MAC RMON Rx Packets w CRC/Align error (HW_ENET_MAC_RMON_R_CRC_ALIGN)

Address: HW_ENET_MAC_RMON_R_CRC_ALIGN – 800F_0000h base + 290h offset = 800F_0290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_CRC_ALIGN																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_CRC_ALIGN field descriptions

Field	Description
31–0 RMON_R_CRC_ ALIGN	Count of ENET MAC RMON Rx Packets w CRC/Align error

26.4.69 ENET MAC RMON Rx Packets < 64 bytes, good CRC (HW_ENET_MAC_RMON_R_UNDERSIZE)

Address: HW_ENET_MAC_RMON_R_UNDERSIZE – 800F_0000h base + 294h offset = 800F_0294h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_UNDERSIZE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_UNDERSIZE field descriptions

Field	Description
31–0 RMON_R_ UNDERSIZE	Count of ENET MAC RMON Rx Packets < 64 bytes, good CRC

26.4.70 ENET MAC RMON Rx Packets > MAX_FL, good CRC (HW_ENET_MAC_RMON_R_OVERSIZE)

Address: HW_ENET_MAC_RMON_R_OVERSIZE – 800F_0000h base + 298h offset = 800F_0298h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_OVERSIZE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_OVERSIZE field descriptions

Field	Description
31–0 RMON_R_ OVERSIZE	ENET MAC RMON Rx Packet count > MAX_FL, good CRC

26.4.71 ENET MAC RMON Rx Packets < 64 bytes, bad CRC (HW_ENET_MAC_RMON_R_FRAG)

Address: HW_ENET_MAC_RMON_R_FRAG – 800F_0000h base + 29Ch offset = 800F_029Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_FRAG																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_FRAG field descriptions

Field	Description
31–0 RMON_R_FRAG	Count of ENET MAC RMON Rx Packets < 64 bytes, bad CRC

26.4.72 ENET MAC RMON Rx Packets > MAX_FL bytes, bad CRC (HW_ENET_MAC_RMON_R_JAB)

Address: HW_ENET_MAC_RMON_R_JAB – 800F_0000h base + 2A0h offset = 800F_02A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_JAB																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_JAB field descriptions

Field	Description
31–0 RMON_R_JAB	Count of ENET MAC RMON Rx Packets > MAX_FL bytes, bad CRC

26.4.73 ENET MAC RMON Rx 64 byte packets (HW_ENET_MAC_RMON_R_P64)

Address: HW_ENET_MAC_RMON_R_P64 – 800F_0000h base + 2A8h offset = 800F_02A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P64																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P64 field descriptions

Field	Description
31–0 RMON_R_P64	Count of ENET MAC RMON Rx 64 byte packets

26.4.74 ENET MAC RMON Rx 65 to 127 byte packets (HW_ENET_MAC_RMON_R_P65TO127)

Address: HW_ENET_MAC_RMON_R_P65TO127 – 800F_0000h base + 2ACh offset = 800F_02ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P65TO127																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P65TO127 field descriptions

Field	Description
31–0 RMON_R_ P65TO127	Count of ENET MAC RMON Rx 65 to 127 byte packets

26.4.75 ENET MAC RMON Rx 128 to 255 byte packets (HW_ENET_MAC_RMON_R_P128TO255)

Address: HW_ENET_MAC_RMON_R_P128TO255 – 800F_0000h base + 2B0h offset = 800F_02B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P128TO255																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P128TO255 field descriptions

Field	Description
31–0 RMON_R_ P128TO255	Count of ENET MAC RMON Rx 128 to 255 byte packets

26.4.76 ENET MAC RMON Rx 256 to 511 byte packets (HW_ENET_MAC_RMON_R_P256TO511)

Address: HW_ENET_MAC_RMON_R_P256TO511 – 800F_0000h base + 2B4h offset = 800F_02B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P256TO511																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P256TO511 field descriptions

Field	Description
31–0 RMON_R_ P256TO511	ENET MAC RMON Rx 256 to 511 byte packet count

26.4.77 ENET MAC RMON Rx 512 to 1023 byte packets (HW_ENET_MAC_RMON_R_P512TO1023)

Address: HW_ENET_MAC_RMON_R_P512TO1023 – 800F_0000h base + 2B8h offset = 800F_02B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P512TO1023																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P512TO1023 field descriptions

Field	Description
31–0 RMON_R_ P512TO1023	Count of ENET MAC RMON Rx 512 to 1023 byte packets

26.4.78 ENET MAC RMON Rx 1024 to 2047 byte packets (HW_ENET_MAC_RMON_R_P1024TO2047)

Address: HW_ENET_MAC_RMON_R_P1024TO2047 – 800F_0000h base + 2BCh offset = 800F_02BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P1024TO2047																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P1024TO2047 field descriptions

Field	Description
31–0 RMON_R_ P1024TO2047	Count of ENET MAC RMON Rx 1024 to 2047 byte packets

26.4.79 ENET MAC RMON Rx packets w > 2048 bytes (HW_ENET_MAC_RMON_R_P_GTE2048)

Address: HW_ENET_MAC_RMON_R_P_GTE2048 – 800F_0000h base + 2C0h offset = 800F_02C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_P_GTE2048																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_P_GTE2048 field descriptions

Field	Description
31–0 RMON_R_P_ GTE2048	Count of ENET MAC RMON Rx packetsw > 2048 bytes

26.4.80 ENET MAC RMON Rx Octets (HW_ENET_MAC_RMON_R_OCTETS)

Address: HW_ENET_MAC_RMON_R_OCTETS – 800F_0000h base + 2C4h offset = 800F_02C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RMON_R_OCTETS																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_RMON_R_OCTETS field descriptions

Field	Description
31–0 RMON_R_ OCTETS	Count of ENET MAC RMON Rx Octets

26.4.81 ENET MAC Frames Received count drop (HW_ENET_MAC_IEEE_R_DROP)

Count of frames not counted correctly

Note: Counter increments if a frame with invalid/missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

Address: HW_ENET_MAC_IEEE_R_DROP – 800F_0000h base + 2C8h offset = 800F_02C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_DROP																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_DROP field descriptions

Field	Description
31–0 IEEE_R_DROP	Count of frames not counted correctly

26.4.82 ENET MAC Frames Received OK (HW_ENET_MAC_IEEE_R_FRAME_OK)

Address: HW_ENET_MAC_IEEE_R_FRAME_OK – 800F_0000h base + 2CCh offset = 800F_02CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_FRAME_OK																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_FRAME_OK field descriptions

Field	Description
31–0 IEEE_R_FRAME_OK	Count of frames Received OK

26.4.83 ENET MAC Frames Received with CRC Error (HW_ENET_MAC_IEEE_R_CRC)

Address: HW_ENET_MAC_IEEE_R_CRC – 800F_0000h base + 2D0h offset = 800F_02D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_CRC																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_CRC field descriptions

Field	Description
31–0 IEEE_R_CRC	Count of frames Received with CRC Error

26.4.84 ENET MAC Frames Received with Alignment Error (HW_ENET_MAC_IEEE_R_ALIGN)

Address: HW_ENET_MAC_IEEE_R_ALIGN – 800F_0000h base + 2D4h offset = 800F_02D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_ALIGN																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_ALIGN field descriptions

Field	Description
31–0 IEEE_R_ALIGN	Count of frames Received with Alignment Error

26.4.85 ENET MAC Frames Received overflow (HW_ENET_MAC_IEEE_R_MACERR)

Receive Fifo Overflow count

Address: HW_ENET_MAC_IEEE_R_MACERR – 800F_0000h base + 2D8h offset = 800F_02D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_MACERR																															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_MACERR field descriptions

Field	Description
31–0 IEEE_R_ MACERR	Value of Frames Received overflow

26.4.86 ENET MAC Frames Received flow control (HW_ENET_MAC_IEEE_R_FDXFC)

Flow Control Pause frames received

Address: HW_ENET_MAC_IEEE_R_FDXFC – 800F_0000h base + 2DCh offset = 800F_02DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_FDXFC																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_FDXFC field descriptions

Field	Description
31–0 IEEE_R_FDXFC	Count of flow Control Pause frames received

26.4.87 ENET MAC Frames Received error (HW_ENET_MAC_IEEE_R_OCTETS_OK)

Octet count for Frames Rcvd w/o Error

Note: counts total octets (includes header and FCS fields)

Address: HW_ENET_MAC_IEEE_R_OCTETS_OK – 800F_0000h base + 2E0h offset = 800F_02E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IEEE_R_OCTETS_OK																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_IEEE_R_OCTETS_OK field descriptions

Field	Description
31–0 IEEE_R_OCTETS_OK	Octet count for Frames Rcvd w/o Error

26.4.88 ENET MAC IEEE1588 Timer Control Register (HW_ENET_MAC_ATIME_CTRL)

Note: The command bits can be used to trigger the corresponding events directly. It is not necessary to preserve any of the configuration bits when a command bit is set in the register (i.e. no read-modify-write is required). The bits read out 0 after the command has completed.

Address: HW_ENET_MAC_ATIME_CTRL – 800F_0000h base + 400h offset = 800F_0400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0 [15:14]		FRC_SLAVE	RSRVD1	CAPTURE	RSRVD2	RESTART	RSRVD3	PIN_PERIOD_ENA	RSRVD4	EVT_PERIOD_RST	EVT_PERIOD_ENA	EVT_OFFSET_RST	EVT_OFFSET_ENA	ONE_SHOT	ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME_CTRL field descriptions

Field	Description
31–14 RSRVD0	Reserved bits. Write as 0.
13 FRC_SLAVE	Enable timer slave mode. If set '1' the internal timer is disabled and instead the externally provided timer value from pins frc_in() is used. When operating in slave mode all other configuration bits in this register have no effect. Only the capture command bit can still be used to capture the current timer value. When disabled (0, default) the timer is active and all configuration bits in this register are relevant.
12 RSRVD1	Reserved bits. Write as 0.
11 CAPTURE	Capture timer value. When set the current time is captured and can be read from the ATIMER register. Command bit: When set, all other bits are ignored during a write.
10 RSRVD2	Reserved bits. Write as 0.

HW_ENET_MAC_ATIME_CTRL field descriptions (continued)

Field	Description
9 RESTART	Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled while the command is triggered, the timer is reset to zero and starts counting from there. Command bit: When set, all other bits are ignored during a write.
8 RSRVD3	Reserved bits. Write as 0.
7 PIN_PERIOD_ ENA	Enable external pin frc_evt_period assertion on period event when set.
6 RSRVD4	Reserved bits. Write as 0.
5 EVT_PERIOD_ RST	Reset timer on periodical event. When set (1) the timer is reset to zero (wraps around) when the period setting is reached (causing an periodical event). If cleared the counter will increment continuously until it wraps around. Should be set 1 for normal operation.
4 EVT_PERIOD_ ENA	Enable periodical event. When set (1) a period event interrupt can be generated (EIR(TS_TIMER)) and the external pin frc_evt_period is asserted when the timer wraps around according to the periodic setting ATIME_EVT_PERIOD. The timer period value should be set before.
3 EVT_OFFSET_ RST	Reset timer on offset event. When set (1) together with the EVT_OFFSET_ENA the timer is reset to zero when the offset setting is reached (causing an offset event).
2 EVT_OFFSET_ ENA	Enable one-shot offset event. When set (1) an offset-event interrupt can be generated (EIR(TS_TIMER)). The bit is cleared when the offset event has been reached so no further event is created until the bit is set again. The timer offset value should be set before.
1 ONE_SHOT	Avoid timer wrap around. If set, the timer stops at maximum. An overflow interrupt is caused (if enabled) when the maximum is reached. When cleared (default) the timer operates continuously.
0 ENABLE	When set (1) the timer starts incrementing. When (0) the timer stops at the current value

26.4.89 ENET MAC IEEE1588 Timer value Register (HW_ENET_MAC_ATIME)

Address: HW_ENET_MAC_ATIME – 800F_0000h base + 404h offset = 800F_0404h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATIME																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME field descriptions

Field	Description
31–0 ATIME	A write sets the timer. A read returns the last captured value. To read the current, value a capture command must be issued in the ATIME_CTRL control register prior to reading this register.

26.4.90 ENET MAC IEEE1588 Offsetvalue for one-shot event generation Register (HW_ENET_MAC_ATIME_EVT_OFFSET)

Address: HW_ENET_MAC_ATIME_EVT_OFFSET – 800F_0000h base + 408h offset = 800F_0408h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATIME_EVT_OFFSET																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME_EVT_OFFSET field descriptions

Field	Description
31–0 ATIME_EVT_ OFFSET	Offsetvalue for one-shot event generation. When the timer reaches the value an event can be generated to reset the counter. If the increment value in ATIME_INC is given in true nanoseconds, this value is also given in true nanoseconds.

26.4.91 ENET MAC IEEE1588 Timer Period Register (HW_ENET_MAC_ATIME_EVT_PERIOD)

Address: HW_ENET_MAC_ATIME_EVT_PERIOD – 800F_0000h base + 40Ch offset = 800F_040Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATIME_EVT_PERIOD																															
Re-set	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME_EVT_PERIOD field descriptions

Field	Description
31–0 ATIME_EVT_ PERIOD	Value for generating periodic events. Each time the timer has reached this time, the period event occurs and the timer restarts. If the increment value in ATIME_INC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1000000000 (=1*109) to represent a timer wrap around of 1 second. The increment value set in ATIME_INC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter.

26.4.92 ENET MAC IEEE1588 Correction counter wrap around value Register (HW_ENET_MAC_ATIME_CORR)

Address: HW_ENET_MAC_ATIME_CORR – 800F_0000h base + 410h offset = 800F_0410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0	ATIME_CORR[30:16]														
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ATIME_CORR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME_CORR field descriptions

Field	Description
31 RSRVD0	Reserved bits. Write as 0.
30–0 ATIME_CORR	Correction counter wrap around value. Correction period. Defines after how many clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATIME_INC(12:8) (see below). E.g. setting the increment amount to zero will stop the timer for one clock cycle, slowing it down. Larger values can be used to speed up the timer. A value of 0 disables the counter and no corrections will occur. Note: This value is given in clock cycles, not in nanoseconds as all other values.

26.4.93 ENET MAC IEEE1588 Clock period of the timestamping clock (ts_clk) in nanoseconds and correction increment Register (HW_ENET_MAC_ATIME_INC)

Address: HW_ENET_MAC_ATIME_INC – 800F_0000h base + 414h offset = 800F_0414h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0 [15:15]	ATIME_INC_CORR							RSRVD1	ATIME_INC						
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_ATIME_INC field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–8 ATIME_INC_ CORR	Correction increment value. This value will be added every time the correction timer expires (every clock cycles given in ATIME_CORR). A value smaller than in bits 6:0 will slow down the timer. A value larger than in bits 6:0 will speed up the timer.
7 RSRVD1	Reserved bits. Write as 0.
6–0 ATIME_INC	Clock period of the timestamping clock (ts_clk) in nanoseconds. The timer will increment by this amount with every clock cycle. For example, 10 for 100MHz, 8 for 125MHz, 5 for 200MHz.

26.4.94 ENET MAC IEEE1588 Timestamp of the last Frame Register (HW_ENET_MAC_TS_TIMESTAMP)

Timestamp of the last Frame transmitted by the controller that had the ff_tx_ts_frm signal asserted from the user application. Valid when the Interrupt status bit TS_AVAIL is set to '1'.

Address: HW_ENET_MAC_TS_TIMESTAMP – 800F_0000h base + 418h offset = 800F_0418h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TS_TIMESTAMP																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_TS_TIMESTAMP field descriptions

Field	Description
31–0 TS_TIMESTAMP	IEEE1588 Timestamp of the last Frame Register

26.4.95 ENET MAC Supplemental MAC Address 0 (HW_ENET_MAC_SMAC_0_0)

Address: HW_ENET_MAC_SMAC_0_0 – 800F_0000h base + 500h offset = 800F_0500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_0_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_0_0 field descriptions

Field	Description
31–0 SMAC_0_0	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match. Note: if the supplemental MAC addresses are not used, they should all be set to the same value as the PALR/PAUR registers.

26.4.96 ENET MAC Supplemental MAC Address 0 (HW_ENET_MAC_SMAC_0_1)

Address: HW_ENET_MAC_SMAC_0_1 – 800F_0000h base + 504h offset = 800F_0504h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_0_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_0_1 field descriptions

Field	Description
31–0 SMAC_0_1	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match. Register Bits 15 to 0 are reserved.

26.4.97 ENET MAC Supplemental MAC Address 1 (HW_ENET_MAC_SMAC_1_0)

Same as define of MAC_SMAC_0_0.

Address: HW_ENET_MAC_SMAC_1_0 – 800F_0000h base + 508h offset = 800F_0508h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_1_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_1_0 field descriptions

Field	Description
31–0 SMAC_1_0	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match. Note: if the supplemental MAC addresses are not used, they should all be set to the same value as the PALR/PAUR registers.

26.4.98 ENET MAC Supplemental MAC Address 1 (HW_ENET_MAC_SMAC_1_1)

Same as define of MAC_SMAC_0_1.

Address: HW_ENET_MAC_SMAC_1_1 – 800F_0000h base + 50Ch offset = 800F_050Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_1_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_1_1 field descriptions

Field	Description
31–0 SMAC_1_1	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match. Register Bits 15 to 0 are reserved.

26.4.99 ENET MAC Supplemental MAC Address 2 (HW_ENET_MAC_SMAC_2_0)

Same as define of MAC_SMAC_0_0.

Address: HW_ENET_MAC_SMAC_2_0 – 800F_0000h base + 510h offset = 800F_0510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_2_0																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_2_0 field descriptions

Field	Description
31–0 SMAC_2_0	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match. Note: if the supplemental MAC addresses are not used, they should all be set to the same value as the PALR/PAUR registers.

26.4.100 ENET MAC Supplemental MAC Address 2 (HW_ENET_MAC_SMAC_2_1)

Same as define of MAC_SMAC_0_1.

Address: HW_ENET_MAC_SMAC_2_1 – 800F_0000h base + 514h offset = 800F_0514h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_2_1																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_2_1 field descriptions

Field	Description
31–0 SMAC_2_1	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match. Register Bits 15 to 0 are reserved.

26.4.101 ENET MAC Supplemental MAC Address 3 (HW_ENET_MAC_SMAC_3_0)

Same as define of MAC_SMAC_0_0.

Address: HW_ENET_MAC_SMAC_3_0 – 800F_0000h base + 518h offset = 800F_0518h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_3_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_3_0 field descriptions

Field	Description
31–0 SMAC_3_0	Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match. Note: if the supplemental MAC addresses are not used, they should all be set to the same value as the PALR/PAUR registers.

26.4.102 ENET MAC Supplemental MAC Address 3 (HW_ENET_MAC_SMAC_3_1)

Same as define of MAC_SMAC_0_1.

Address: HW_ENET_MAC_SMAC_3_1 – 800F_0000h base + 51Ch offset = 800F_051Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMAC_3_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_SMAC_3_1 field descriptions

Field	Description
31–0 SMAC_3_1	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match. Register Bits 15 to 0 are reserved.

26.4.103 ENET MAC Compare register 0 (HW_ENET_MAC_COMP_REG_0)

When the timer reaches this value, the event pin evt_out(0) is asserted for one ts_clk cycle. This repeats when the timer wraps around. A value of 0 disables the function.

Address: HW_ENET_MAC_COMP_REG_0 – 800F_0000h base + 600h offset = 800F_0600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP_REG_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_COMP_REG_0 field descriptions

Field	Description
31–0 COMP_REG_0	Value of compare register 0

26.4.104 ENET MAC Compare register 1 (HW_ENET_MAC_COMP_REG_1)

When the timer reaches this value, the event pin evt_out(1) is asserted for one ts_clk cycle. This repeats when the timer wraps around. A value of 0 disables the function.

Address: HW_ENET_MAC_COMP_REG_1 – 800F_0000h base + 604h offset = 800F_0604h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP_REG_1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_COMP_REG_1 field descriptions

Field	Description
31–0 COMP_REG_1	Value of compare register 1

26.4.105 ENET MAC Compare register 2 (HW_ENET_MAC_COMP_REG_2)

When the timer reaches this value, the event pin evt_out(2) is asserted for one ts_clk cycle. This repeats when the timer wraps around. A value of 0 disables the function.

Address: HW_ENET_MAC_COMP_REG_2 – 800F_0000h base + 608h offset = 800F_0608h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP_REG_2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_COMP_REG_2 field descriptions

Field	Description
31–0 COMP_REG_2	Value of compare register 2

26.4.106 ENET MAC Compare register 3 (HW_ENET_MAC_COMP_REG_3)

When the timer reaches this value, the event pin evt_out(3) is asserted for one ts_clk cycle. This repeats when the timer wraps around. A value of 0 disables the function.

Address: HW_ENET_MAC_COMP_REG_3 – 800F_0000h base + 60Ch offset = 800F_060Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP_REG_3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_COMP_REG_3 field descriptions

Field	Description
31–0 COMP_REG_3	Value of compare register 3

26.4.107 ENET MAC Capture register 0 (HW_ENET_MAC_CAPT_REG_0)

Timer value latched when a rising edge occured on input pin evt_in(0).

Address: HW_ENET_MAC_CAPT_REG_0 – 800F_0000h base + 640h offset = 800F_0640h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT_REG_0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_CAPT_REG_0 field descriptions

Field	Description
31–0 CAPT_REG_0	Value of capture register 0

26.4.108 ENET MAC Capture register 1 (HW_ENET_MAC_CAPT_REG_1)

Timer value latched when a rising edge occured on input pin evt_in(1).

Address: HW_ENET_MAC_CAPT_REG_1 – 800F_0000h base + 644h offset = 800F_0644h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT_REG_1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_CAPT_REG_1 field descriptions

Field	Description
31–0 CAPT_REG_1	Value of capture register 1

26.4.109 ENET MAC Capture register 2 (HW_ENET_MAC_CAPT_REG_2)

Timer value latched when a rising edge occured on input pin evt_in(2).

Address: HW_ENET_MAC_CAPT_REG_2 – 800F_0000h base + 648h offset = 800F_0648h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT_REG_2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_CAPT_REG_2 field descriptions

Field	Description
31–0 CAPT_REG_2	Value of capture register 2

26.4.110 ENET MAC Capture register 3 (HW_ENET_MAC_CAPT_REG_3)

Timer value latched when a rising edge occured on input pin evt_in(3).

Address: HW_ENET_MAC_CAPT_REG_3 – 800F_0000h base + 64Ch offset = 800F_064Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT_REG_3																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_MAC_CAPT_REG_3 field descriptions

Field	Description
31–0 CAPT_REG_3	Value of capture register 3

26.4.111 ENET MAC IEEE1588 Interrupt register. (HW_ENET_MAC_CCB_INT)

For each capture/compare event an interrupt can be generated. The interrupt is cleared by writing a 1 to the corresponding bit

Note: The interrupt bits are set on event occurence. To clear an interrupt the corresponding bit must be written with 1.

Address: HW_ENET_MAC_CCB_INT – 800F_0000h base + 680h offset = 800F_0680h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0												COMPARE3	COMPARE2	COMPARE1	COMPARE0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1												CAPTURE3	CAPTURE2	CAPTURE1	CAPTURE0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

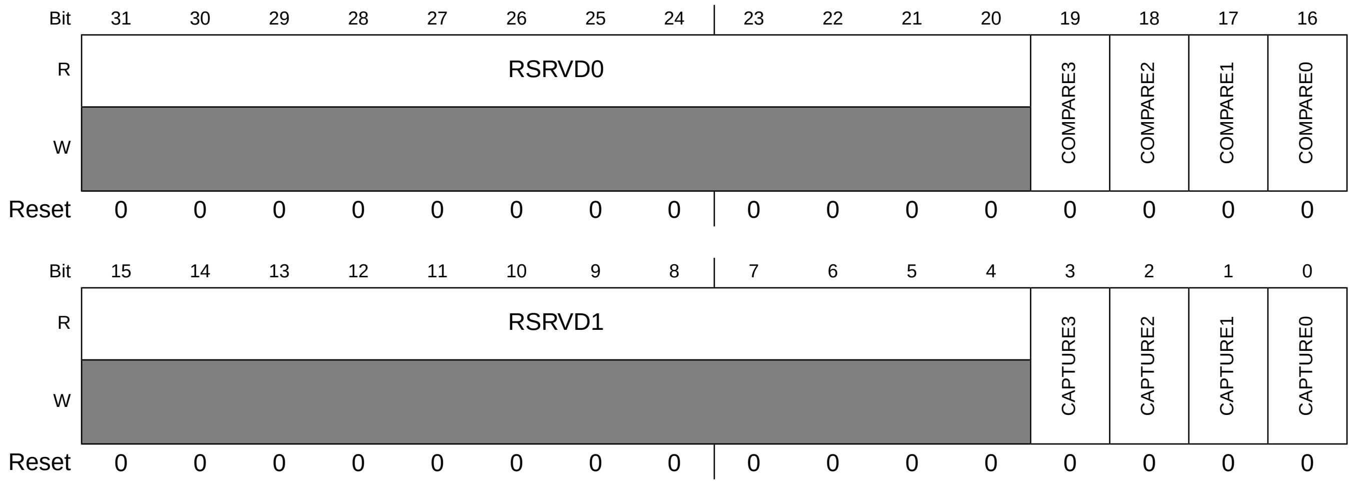
HW_ENET_MAC_CCB_INT field descriptions

Field	Description
31–20 RSRVD0	Reserved bits. Write as 0.
19 COMPARE3	compare event 3.
18 COMPARE2	compare event 2.
17 COMPARE1	compare event 1.
16 COMPARE0	compare event 0.
15–4 RSRVD1	Reserved bits. Write as 0.
3 CAPTURE3	capture event 3.
2 CAPTURE2	capture event 2.
1 CAPTURE1	capture event 1.
0 CAPTURE0	capture event 0.

26.4.112 ENET MAC IEEE1588 Interrupt enable mask register (HW_ENET_MAC_CCB_INT_MASK)

An interrupt is created when an interrupt is indicated (CCB_INT) and the corresponding mask bit is set to 1. When an interrupt occurs the ccb_int pin is asserted.

Address: HW_ENET_MAC_CCB_INT_MASK – 800F_0000h base + 684h offset = 800F_0684h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R RSRVD0 COMPARE3 COMPARE2 COMPARE1 COMPARE0
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R RSRVD1 CAPTURE3 CAPTURE2 CAPTURE1 CAPTURE0
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ENET_MAC_CCB_INT_MASK field descriptions

Field	Description
31–20 RSRVD0	Reserved bits. Write as 0.
19 COMPARE3	compare event 3 interrupt mask.
18 COMPARE2	compare event 2 interrupt mask.
17 COMPARE1	compare event 1 interrupt mask.
16 COMPARE0	compare event 0 interrupt mask.
15–4 RSRVD1	Reserved bits. Write as 0.
3 CAPTURE3	capture event 3 interrupt mask.
2 CAPTURE2	capture event 2 interrupt mask.
1 CAPTURE1	capture event 1 interrupt mask.
0 CAPTURE0	capture event 0 interrupt mask.

Chapter 27
Inter IC (I2C)

27.1 I^{2}C Overview

The I^{2}C is a standard two-wire serial interface used to connect the chip with peripherals or host controllers. This interface provides a standard speed (up to 100 kbps), and a fast speed (up to 400 kbps) I^{2}C connection to multiple devices with the chip acting in either I^{2}C master or I^{2}C slave mode. Typical applications for the I^{2}C bus include: EEPROM, LED/LCD, FM tuner, cell phone baseband chip connection, and so on.

The I^{2}C port supports multi-master configurations.

As implemented on the i.MX28, the I^{2}C block includes the following functions:

- The I^{2}C block can be configured as either a master or slave device. In master mode, it generates the clock (I2C_SCL) and initiates transactions on the data line (I2C_SDA).
- The I^{2}C block packs/unpacks data into 8-, 16-, 24-, or 32-bit words for DMA transactions. Data on the I^{2}C bus is always byte-oriented. Short transmission (up to three bytes plus address) can be easily triggered using only PIO operations, that is, no DMA setup required.
- PIO mode, that is, soft DMA mode is supported.
- PIO Queue mode is supported. This mode allows software to queue up multiple commands and supporting data for later automatic execution.
- The I^{2}C block support programmable 7-bit and 10-bit device addresses for master transactions. It also has a programmable 7-bit address that defaults to 0x43 = 7'b1000011 for slave transactions. As seen in the 8-bit device address byte, this address corresponds to 0x86 where the least significant bit (LSB) is the R/W bit. 10-bit address is not supported in slave mode.

- Master transactions are composed of one or more DMA commands chained together. The first byte conveys the slave address and read/write bit for the first command. If the entire transaction is an I^{2}C write command, then it can be sent by a single DMA command. If the command is an I^{2}C read transaction, then at least two DMA commands are required to handle it.
- When the slave interface is enabled, it immediately goes into address search mode and searches for a start event. It then looks for a match on its programmable device address. As soon as the address byte is matched, it is acknowledged on the I^{2}C bus and then the SCL clock is held low until released by software. The address phase initiates a CPU interrupt, if a slave address match is detected. Software then reads the address LSB to determine whether to use a read or write DMA command to complete the slave transaction.
- The I^{2}C block does not support CBUS (start byte is 00000001) device in master mode.

Figure 27-1 shows a block diagram of the I^{2}C interface implemented on the i.MX28.

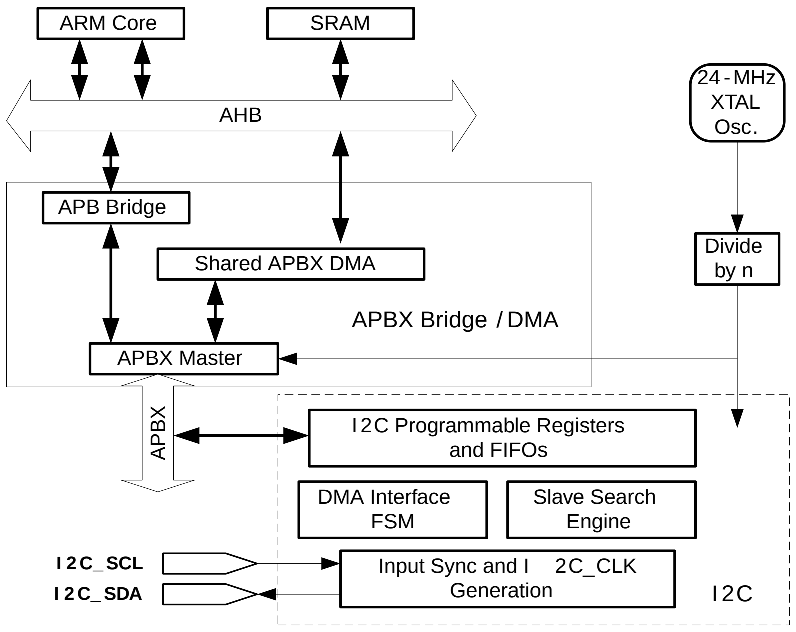


Figure 27-1. I^{2}C Interface Block Diagram

27.2 Operation

The I^{2}C Interface on the i.MX28 includes the following external pins:

- **I2C_SDA: I^{2}C Serial Data**—This pin carries all address and data bits.
- **I2C_SCL: I^{2}C Serial Clock**—This pin carries the clock used to time the address and data.

Pullup resistors are required on both of the I^{2}C lines as all of the I^{2}C drivers are open drain (pulldown only). Typically, external 2kΩ resistors are used to pull the signals up to VddIO for normal and fast speeds.

27.2.1 I^{2}C Interrupt Sources

The I^{2}C port can be used in either interrupt-driven or polled modes. An interrupt can be generated by the completion of a DMA command in the APBX DMA. DMA interrupts are the reporting mechanism for I^{2}C transactions that terminate normally. Abnormal terminations or partial completions are signaled by interrupts generated within the I^{2}C controller.

If I^{2}C interrupts are enabled, a level-sensitive interrupt are signaled to the processor upon one of the events listed in Table 27-1.

Table 27-1. I^{2}C Slave and Master Interrupt Condition in HW_I2C_CTRL1

SOURCE	Bit Name	Description
Slave Address	SLAVE_IRQ	This interrupt is generated when an address match occurs. It indicates that the CPU should read the captured RW bit from the I^{2}C address byte to determine the type of DMA to use for the data transfer phase.
Slave Stop	SLAVE_STOP_IRQ	This interrupt is generated when a stop condition is detected after a slave address has been matched.
Oversize Xfer	OVERSIZE_XFER_TERM_IRQ	The DMA and I^{2}C controller are initialized for an expected transfer size. If the data phase is not terminated within this transfer size then oversize transfer processing goes into effect and the CPU is alerted through this interrupt. This interrupt is only used in slave mode.
Early Termination	EARLY_TERM_IRQ	The DMA and I^{2}C controller are initialized for an expected transfer size. If the data phase is terminated before this transfer size then early termination processing goes into effect and the CPU is alerted through this interrupt.
Master Loss	MASTER_LOSS_IRQ	A master begins transmission on an idle I^{2}C bus and monitors the data line. If it ever attempts to send a one on the line and notes that a zero has been sent instead, then it notes that it has lost mastership of the I^{2}C bus. It terminates its transfer and reports the condition to the CPU through this interrupt. This detection only happens on master transmit operations.
No Slave Ack	NO_SLAVE_ACK_IRQ	When a start condition is transmitted in master mode, the next byte contains an address for a targeted slave. If the targeted slave does not acknowledge the address byte, then this interrupt is set, no further I^{2}C protocol is processed, and the I^{2}C bus returns to the idle state.

SOURCE	Bit Name	Description
Data Engine Complete	DATA_ENGINE_CMPLT_IRQ	This bit is set whenever the DMA interface state machine completes a transaction and resets its run bit. This is useful for PIO mode transmit transactions that are not mediated by the DMA and therefore cannot use the DMA command completion interrupt. This bit is still set for master completions when the DMA is used, but can be ignored in that case.
Bus Free	BUS_FREE_IRQ	When bus mastership is lost during the I^{2}C arbitration phase, the bus becomes busy running services for another master. This interrupt is set whenever a stop command is detected so the master transaction can attempt a retry.
Read Queue Threshold	RD_QUEUE_IRQ	This interrupt is set whenever the read FIFO has filled up equal to or greater than the programmed threshold level.
Write Queue Threshold	WR_QUEUE_IRQ	This interrupt is set whenever the write FIFO has drained down equal to or less than the programmed threshold level.

The interrupt lines are tied directly to the bits of Control Register 1. Clearing these bits through software removes the interrupt request.

27.2.2 I^{2}C Bus Protocol

The I^{2}C block can be programmed and driven by the SoC using one of three internal interface modes: DMA mode, PIO mode and PIO Queue mode. The discussion of the I^{2}C protocol in the following subsections is written from the perspective of DMA mode. The descriptions of how to use the other two modes and their similarities and difference from DMA mode will be included in a later section.

In DMA mode, one DMA command chain can only contain one data-transfer command besides address-transfer command(s). If it’s desired to transfer data to/from two different system memory buffers, two dma transfer should be used.

The I^{2}C interface operates as shown in Figure 27-2 and Figure 27-3.

- A START condition is defined as a high-to-low transition on the data line while the I2C_SCL line is held high.
- After this has been transmitted by the master, the bus is considered busy.
- The next byte of data transmitted after the start condition contains the address of the slave in the first seven bits, and the eighth bit tells whether the Master is receiving data from the slave or transmitting data to the slave.

- When an address is sent, each device in the system compares the first seven bits after a start condition with its address.
- If they match, the device considers itself addressed by the master.

In slave mode, the default I^{2}C write address is 86h, and its default read address is 87h. The slave address is programmable.

Data transfer with acknowledge is obligatory.

- The transmitter must release the I2C_SDA line during the acknowledge pulse.
- The receiver must then pull the data line low, so that it remains stable low during the high period of the acknowledge clock pulse.
- A receiver that has been addressed is obliged to generate an acknowledge after each byte of data has been received.
- A slave device can terminate a transfer by withholding its acknowledgement.

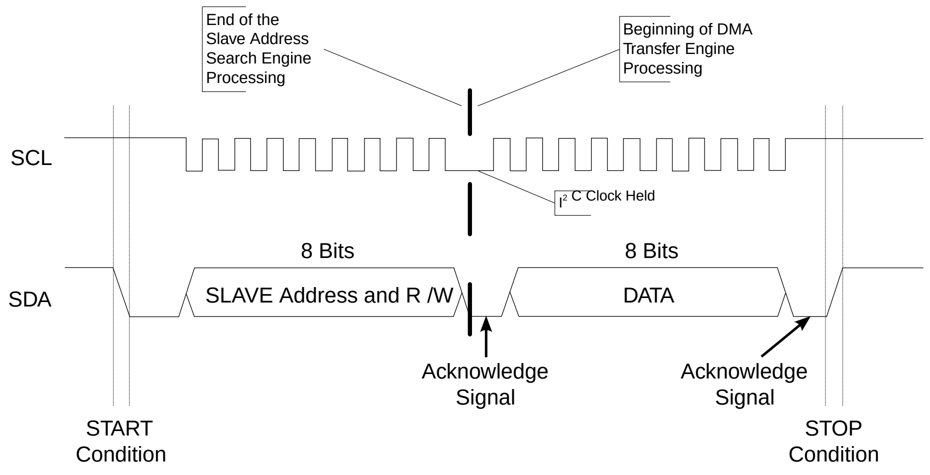


Figure 27-2. I^{2}C Data and Clock Timing

The clock is generated by the master, according to parameters set in the HW_I2C_TIMINGn register. This register also provides programmable timing for capturing received data, as well as for changing the transmitted data bit.

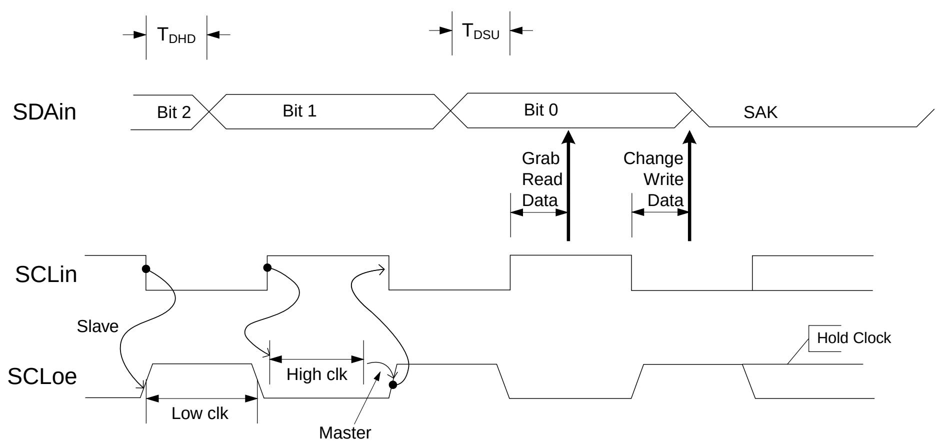


Figure 27-3. I^{2}C Data and Clock Timing Generation

27.2.2.1 Simple Device Transactions

The simplest transfer of interest on an I^{2}C bus is writing a single data byte from a master to a slave, for example, writing a single byte to an FM tuner. In this transaction, a start condition is transmitted, followed by the device address byte, followed by a single byte of write data. This sequence always ends with a stop condition.

Table 27-2. I^{2}C Transfer When the Interface is Transmitting as a Master

ST	SAD+W	SAK	DATA	SAK	SP

Table 27-3 defines the symbols used in describing I^{2}C transactions. For example, in the single byte write operation, ST is a start condition, and SP is a stop condition. The data transfer occurs between these two bus events. It starts with a slave address plus write byte (SAD+W) addressing the targeted slave. A slave-generated acknowledge bit (SAK) tells the master that a slave has recognized the address and will accept the transfer. The master sends the data byte (DATA), and the slave acknowledges it with an SAK.

Table 27-3. I^{2}C Slave and Master Mode Address Definitions

BIT	Description
ST	Start Condition
SR	Repeated Start Condition
SAD	Slave Address
SAK	Slave Acknowledge
SUB	Sub-Address, e.g., for EEPROMs
DATA	Data

BIT	Description
SP	Stop Condition
MAK	Master Acknowledge
NMAK	No Master Acknowledge

To receive one data byte from a slave device such as an FM tuner, the following bus transaction takes place.

Table 27-4. I^{2}C Transfer FM Tuner Read of One Byte

ST	SAD+R	SAK	DATA	NMAK	SP

In this transaction:

- The master first generates a start condition, ST.
- It then sends the seven-bit slave address for the FM tuner plus a read bit (SAD+R).
- The slave in the FM tuner responds with a slave acknowledge bit (SAK).
- The master then generates I^{2}C clocks for a data byte to be transferred (DATA).
- The slave provides data to the I^{2}C data bus during the DATA byte transfer.
- Next, the master generates a master non-acknowledge to the slave (NMAK), indicating the end of the data transfer to the slave. The slave will then release the data line.
- Finally, the master generates a stop condition (SP), terminating the transaction and freeing the I^{2}C bus for other masters to use.

The following example shows a multiple byte read from an FM tuner or other slave device:

Table 27-5. I^{2}C Transfer FM Tuner Read of Three Bytes

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

27.2.2.2 Typical EEPROM Transactions

I^{2}C EEPROMs typically have a specific transaction sequence for reading and writing data bytes to and from the EEPROM array. Table 27-6 through Table 27-9 show the first two bytes of data as a sub-address for purposes of illustration. The sub-address is used to address the memory space inside the device. Table 27-3 defines each element of the transactions shown. When writing a single byte of data to the EEPROM, one must first transfer two bytes of sub-address as follows:

Table 27-6. I^{2}C Transfer When Master is Writing One Byte of Data to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	SP

The sub-address only needs to be specified once for a multibyte transfer, as shown here. Note that the sub-address must be sent for each start condition that initiates a transaction.

Table 27-7. I^{2}C Transfer When Master is Writing Multiple Bytes to a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	DATA	SAK	SP

One must also provide the sub-address before reading bytes from the EEPROM. The sub-address is transmitted from the master to the slave before it can receive data bytes. The two transfers are joined into a single bus transaction through the use of a repeated start condition (SR). Normally, a stop condition precedes a start condition. However, when a start condition is preceded by another start condition, it is known as a repeated start (SR). Note that the two-byte sub address is transferred using an SAD+W address, while the data is received using a SAD+R address.

Table 27-8. I^{2}C Transfer When Master is Receiving One Byte of Data from a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	NMAK	SP

Table 27-9. I^{2}C Transfer When Master is Receiving Multiple Bytes of Data from a Slave

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

27.2.2.3 Master Mode Protocol

In master mode, the I^{2}C interface generates the clock and initiates all transfers.

27.2.2.4 Clock Generation

The I^{2}C clock is generated from the APBX clock, as described in the register description.

- If another device pulls the clock low before the I^{2}C block has counted the high period, then the I^{2}C block immediately pulls the clock low as well and starts counting its low period.
- Once the low period has been counted, the I^{2}C block releases the clock line high, but must then check to see if another device stills holds the line low, in which case it enters a high wait state.

In this way, the I2C_SCL clock is generated, with its low period determined by the device with the longest clock low period and its high period determined by the one with the shortest clock high period.

27.2.2.5 Master Mode Operation

The finite state machine for master mode operation is shown in Figure 27-4 through Figure 27-7. Figure 27-4 shows the generation of the optional start condition. Figure 27-5 shows the receive states, Figure 27-6 shows the transmit states. Figure 27-7 shows the generation of the optional stop state.

Table 27-10 through Table 27-13 show examples of Master Mode I^{2}C transactions. Table 27-3 defines each sub-address shown. The following read-after-write transactions are performed using the restart technique.

Table 27-10. I^{2}C Transfer When Master is Transmitting 1 Byte of Data to Slave Internal Subaddress

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	DATA	SAK	SP

Table 27-11. I^{2}C Transfer When the Interface as Master is Receiving >1 Byte of Data from Slave

ST	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

Table 27-12. I^{2}C Transfer when Master is Receiving 1 Byte of Data from Slave Internal Subaddress

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	NMAK	SP

Table 27-13. I^{2}C Transfer When Master is Receiving >1 byte of Data from Slave Internal Subaddress

ST	SAD+W	SAK	SUB	SAK	SUB	SAK	SR	SAD+R	SAK	DATA	MAK	DATA	MAK	DATA	NMAK	SP

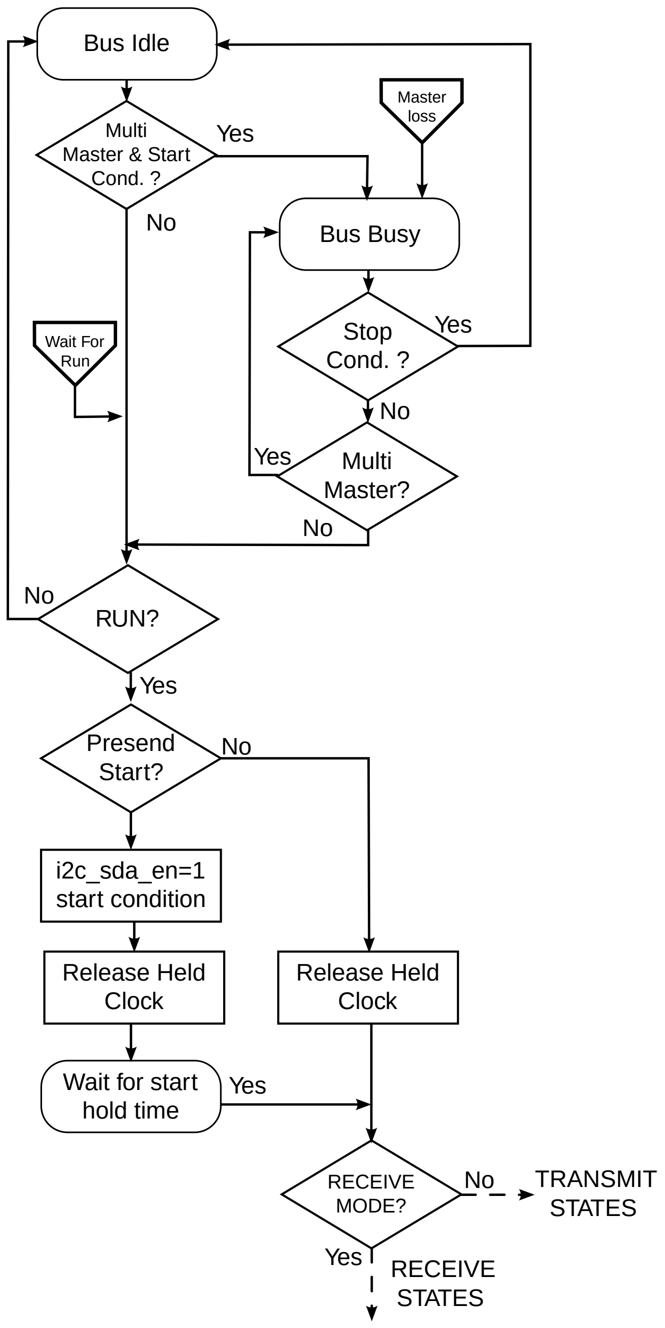


Figure 27-4. I^{2}C Master Mode Flow Chart-Initial States

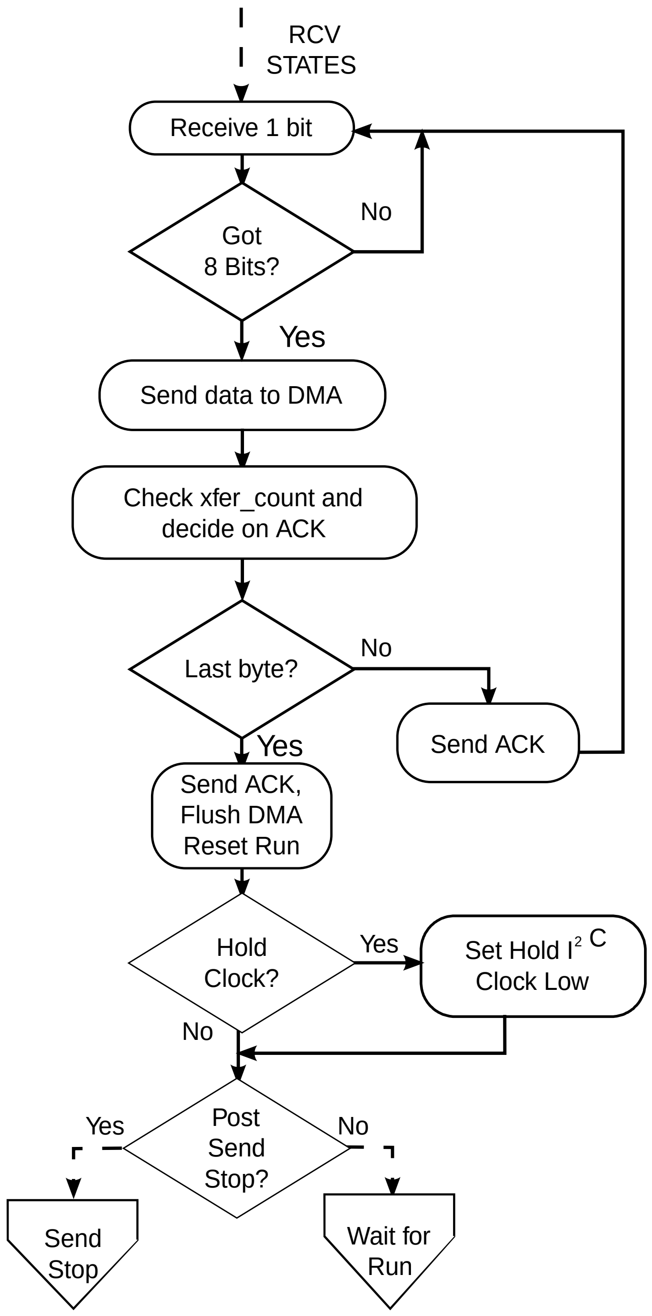


Figure 27-5. I^{2}C Master Mode Flow Chart-Receive States

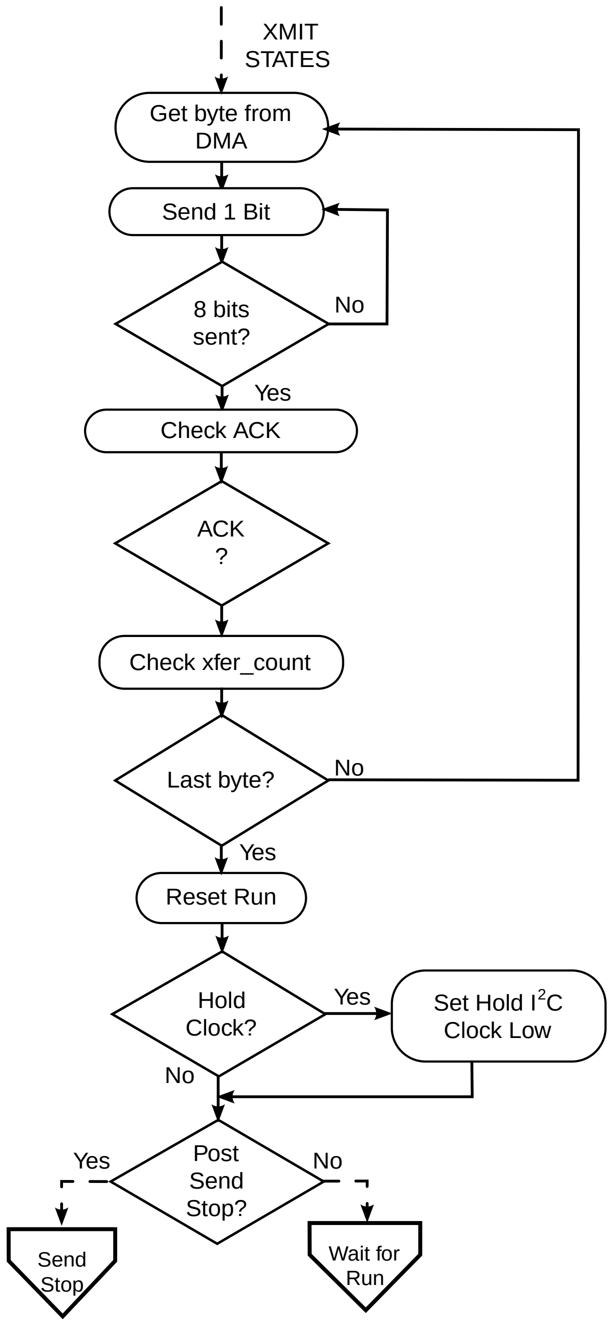


Figure 27-6. I^{2}C Master Mode Flow Chart-Transmit States

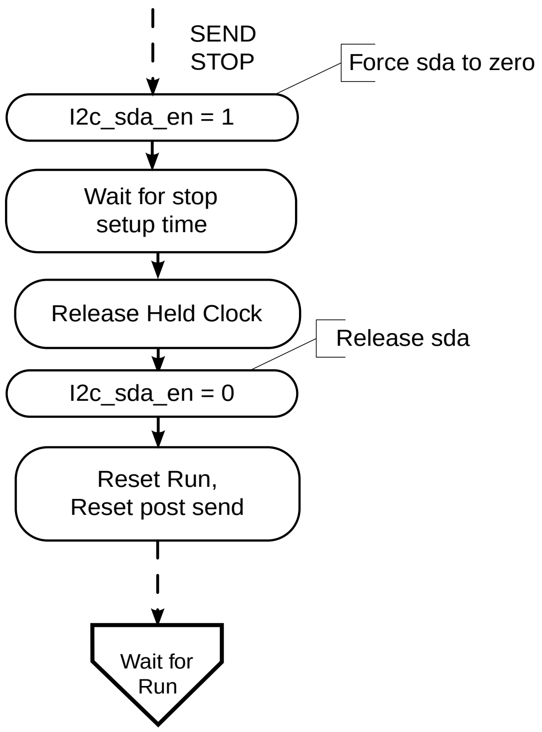


Figure 27-7. I^{2}C Master Mode Flow Chart-Send Stop States

27.2.2.6 Slave Mode Protocol

The I^{2}C slave protocol is handled by a combination of I^{2}C functional block hardware, the DMA, and some supporting software to intervene in the transaction.

The flow chart for slave mode is shown in Figure 27-8.

- At device start-up, all the registers are reset so that the state is known from that time onward.
- Once the I^{2}C slave search engine is enabled, the slave waits to detect a start condition on the I2C_SCL and I2C_SDA lines.
- Once this is detected, the slave reads in eight bits and checks against its programmed device address (which defaults to 0x86 == 7'b1000011) to see if a master device is trying to start a transfer with i.MX28 operating as a slave.
- If it is the programmed address, an acknowledgement is sent; otherwise, the slave does not acknowledge and returns to state IDLE.
- Once the slave search engine detects an address, it holds the clock line and interrupts the CPU.
- Next the software checks the RW bit.

- If it is a write operation, then the software programs the DMA channel for a DMA_WRITE (to on-chip RAM or off-chip SDRAM).
- The slave search engine leaves the programmable state set up for the DMA transfer engine to send the address acknowledge for the address byte as soon as the clock is released.
- It then accepts eight-bit bytes and pushes them into the DMA data register, acknowledging each data byte as it is received, until the transfer count reaches zero.
- The DMA engine stops with the clock held and the hardware ready to acknowledge the last byte when the clock is released. Software decides whether the last byte is acknowledged or not.
- If the master is requesting a read operation, then the i.MX28 slave must start sending data on the I2C_SDA bus immediately after acknowledging the slave address and RW bit.
- After each byte, the acknowledgement from the master must be checked. When the master has received the last byte, it does not send an acknowledgement, and the slave terminates while setting the Early Termination interrupt request. This notifies software that the DMA will not be interrupting for the termination and that software should deal with a shorter than expected packet of data.
- If the transfer count reaches zero and the master has not sent an MNAK or stop condition, then the slave DMA transfer controller terminates the transfer while setting the Oversize Transfer interrupt request. This notifies software to set up for an additional buffer of data to transmit to the master.

Data is transmitted in byte format. Each data transfer has to contain eight bits. The number of bytes transferred per transfer is unlimited. Data is transferred with the most significant bit (MSB) first. If a receiver cannot receive another complete byte of data until it has performed some other function, it can hold the I2C_SCL clock line low to force the transmitter into a wait state. Data transfer can only continue when the receiver is ready for another byte and releases the clock line.

If a slave receiver does not acknowledge the slave address (for example, it is unable to receive because it is performing some real-time function), the data line must be left high by the slave. The master can then abort the transfer.

A low-to-high transition on the I2C_SDA line while the I2C_SCL line is high is defined as a Stop condition. Each data transfer must be terminated by the generation of a Stop condition. A write transfer from a master can be terminated by the master by sending a Stop condition instead of an additional data byte. The i.MX28 slave DMA transfer engine reports this to software as an Early Termination interrupt request.

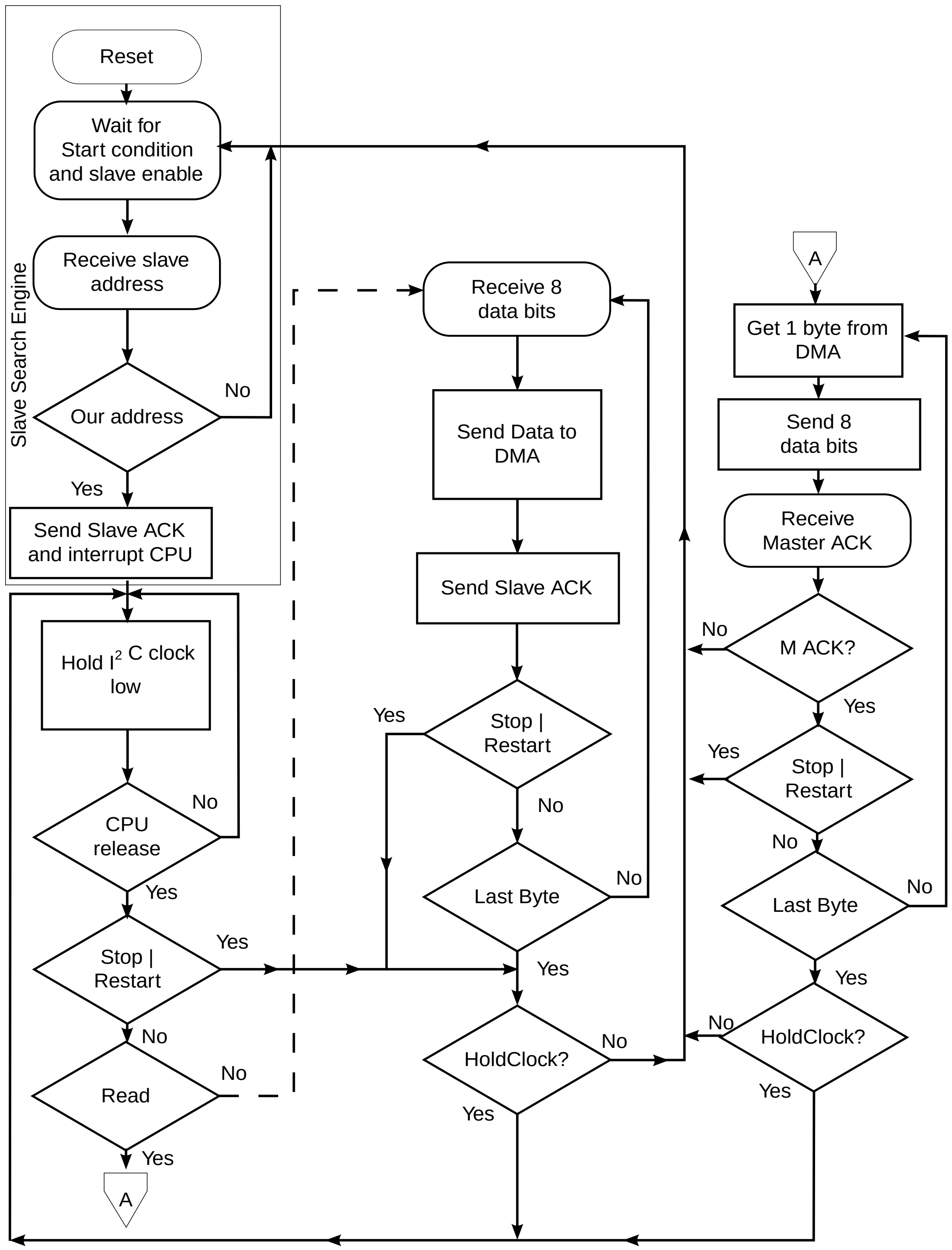


Figure 27-8. I^{2}C Slave Mode Flow Chart

27.2.3 Programming Examples

This section provides two programming examples, Five Byte Master Write Using DMA and Reading 256 bytes from an EEPROM.

27.2.3.1 Five Byte Master Write Using DMA

The example in Figure 27-9 shows sending five bytes from an i.MX28 operating as an I^{2}C master to another device acting as an I^{2}C slave.

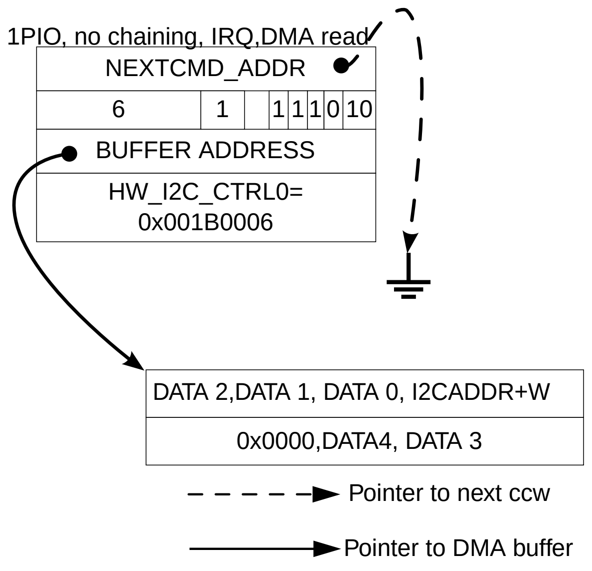


Figure 27-9. I^{2}C Writing Five Bytes

The DMA command is initialized to send six bytes to the I^{2}C controller and one word of PIO information to the HW_I2C_CTRL0 register.

Table 27-14. I^{2}C Transfer When the Master Transmits 5 Bytes of Data to the Slave

ST	SAD+W	SAK	DATA0	SAK	DATA1	SAK	DATA2	SAK	DATA3	SAK	DATA4	SAK	SP

The following C code is used to send a five-byte transmission:

```
// SEND: start, 0x56, 0x01,0x02,0x03,0x04,0x05,stop
//----------------------------------------------------
#define I2C_CHANNEL_NUM 3
// dma buffer of 6 bytes (i2c address + 5 data bytes)
static reg32_t I2C_DATA_BUFFER[2]=
{
0x03020156,  //slave address 56+W
0x00000504   // last two data bytes
};
// DMA command chain
const static reg32_t  I2C_DMA_CMD[4] =
{
(reg32_t)  0,
(BF_APBX_CHn_CMD_XFER_COUNT(6) |
BF_APBX_CHn_CMD_SEMAPHORE(1)   |
BF_APBX_CHn_CMD_CMDWORDS(1)    |
BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
BF_APBX_CHn_CMD_CHAIN(0)       |
BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
(reg32_t) &eeprom_command_buffer[0],
BF_I2C_CTRL0_POST_SEND_STOP(BV_I2C_CTRL0_POST_SEND_STOP_SEND_STOP)|
BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START_SEND_START) |
BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE_MASTER)            |
BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION_TRANSMIT)              |
BF_I2C_CTRL0_XFER_COUNT(6)
};


void SendFiveBytes(){
// Reset the APBX dma channels associated with I2C.
reset_mask = BF_APBX_CTRL0_RESET_CHANNEL((1 << I2C_CHANNEL_NUM));
HW_APBX_CTRL0_SET(reset_mask);
// Poll for reset to clear the channel.
for (retries = 0; retries < RESET_TIMEOUT; retries++)
if ((reset_mask & HW_APBX_CTRL0_RD()) == 0)
break;
if( retries == RESET_TIMEOUT) exit(1);
// Setup dma channel configuration.
BF_WRn(APBX_CHn_NXTCMDAR,I2C_CHANNEL_NUM,
CMD_ADDR,(reg32_t) I2C_DMA_CMD);
BF_WR(APBX_CTRL1, CH3_CMDCMPLT_IRQ, 0); // clear interrupt
// Start the dma channel by incrementing semaphore.
BF_WRn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, INCREMENT_SEMA, 1);

// Poll for the semaphore to decrement to zero on the DMA channel.
for (retries = 0; retries < SEMAPHORE_TIMEOUT; retries++)
if (0 == BF_RDn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, PHORE))
break;
// a frame with one byte of address and five bytes of data was just sent
}
```

27.2.3.2 Reading 256 bytes from an EEPROM

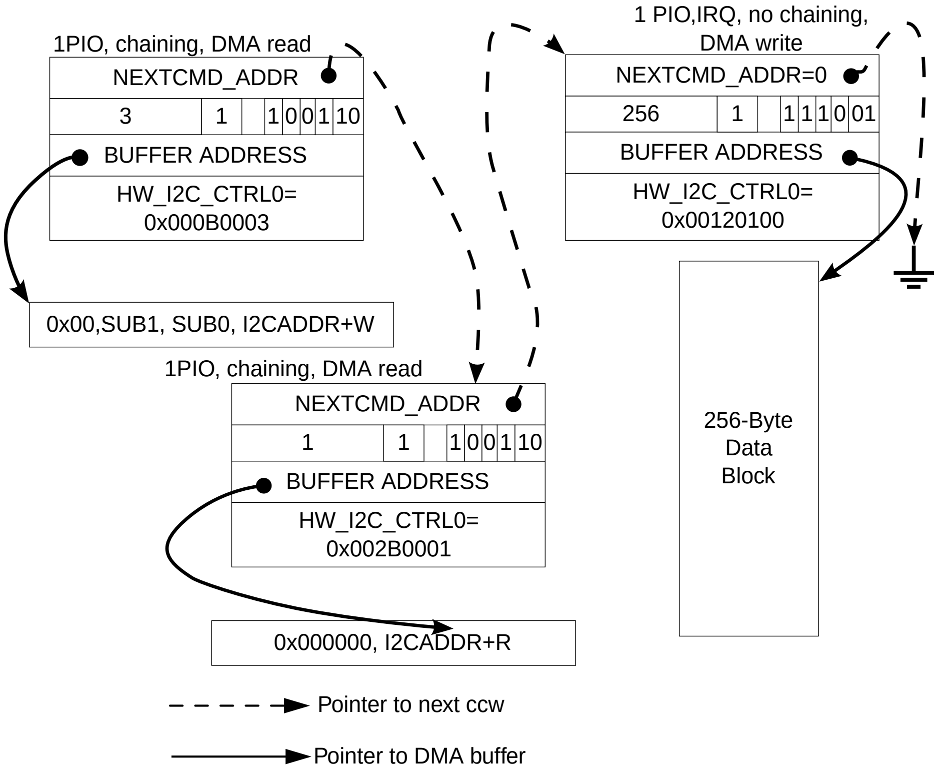


Figure 27-10. I^{2}C Reading 256 Bytes from an EEPROM

```
//-----------------------------------------------------------------------
// dma buffers to hold i2c command string for slave addres+W plus sub0,
// sub 1 and the second command, a slave address+R
// eePROM write address == 0xA0,  read address == 0xA1
//-----------------------------------------------------------------------
unsigned char eeprom_command_buffer[4] = {0xA0,0x34,0x12,0xA1};
//-----------------------------------------------------------------------
// I2C DMA chain
//-----------------------------------------------------------------------
const static reg32_t  I2C_DMA_CMD3[4] =
{
    0x0,
    (BF_APBX_CHn_CMD_XFER_COUNT(256) |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)  |
     BF_APBX_CHn_CMD_SEMAPHORE(1)    |
     BF_APBX_CHn_CMD_CMDWORDS(1)     |
     BF_APBX_CHn_CMD_CHAIN(0)        |            // last command
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_WRITE)),
    (reg32_t) &eeprom_command_buffer[3],
    BF_I2C_CTRL0_POST_SEND_STOP(BV_I2C_CTRL0_POST_SEND_STOP_SEND_STOP)|
    BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE_MASTER)         |
    BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION_RECEIVE)            |
    BF_I2C_CTRL0_XFER_COUNT(256)
};
const static reg32_t  I2C_DMA_CMD2[4] =
{
    (reg32_t)  I2C_DMA_CMD3,
    (BF_APBX_CHn_CMD_XFER_COUNT(1) |
     BF_APBX_CHn_CMD_CMDWORDS(1)    |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
     BF_APBX_CHn_CMD_CHAIN(1)       |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[3],
   BF_I2C_CTRL0_RETAIN_CLOCK(BV_I2C_CTRL0_RETAIN_CLOCK_HOLD_LOW)       |
   BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START_SEND_START)|
   BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE_MASTER)           |
   BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION_TRANSMIT)             |
   BF_I2C_CTRL0_XFER_COUNT(1)
};
const static reg32_t  I2C_DMA_CMD1[4] =
{
    (reg32_t)  I2C_DMA_CMD2,
    (BF_APBX_CHn_CMD_XFER_COUNT(3) |
     BF_APBX_CHn_CMD_CMDWORDS(1)    |
     BF_APBX_CHn_CMD_WAIT4ENDCMD(1)|
     BF_APBX_CHn_CMD_CHAIN(1)       |
     BV_FLD(APBX_CHn_CMD, COMMAND, DMA_READ)),
    (reg32_t) &eeprom_command_buffer[0],
  BF_I2C_CTRL0_PRE_SEND_START(BV_I2C_CTRL0_PRE_SEND_START_SEND_START) |
  BF_I2C_CTRL0_MASTER_MODE(BV_I2C_CTRL0_MASTER_MODE_MASTER)           |
  BF_I2C_CTRL0_DIRECTION(BV_I2C_CTRL0_DIRECTION_TRANSMIT)             |
  BF_I2C_CTRL0_XFER_COUNT(3)
};

/////////////////////////////////////////////////////
//Read256BytesFromEEPROM returns 1 for errors and 0 for OK
/////////////////////////////////////////////////////
int Read256BytesFromEEPROM(unsigned short usAddress){
 // insert eePROM addres param into dma command buffer
  I2C_CMD_BUFFER[1] = (unsigned char) (usAddress      &0x00ff);
  I2C_CMD_BUFFER[2] = (unsigned char) ((usAddress>>8) &0x00ff);
 // Reset the APBX dma channels associated with I2C.
 reset_mask = BF_APBX_CTRL0_RESET_CHANNEL((1 << I2C_CHANNEL_NUM));
 HW_APBX_CTRL0_SET(reset_mask);
 // Poll for reset to clear the channel.
 for (retries = 0; retries < RESET_TIMEOUT; retries++)
      if ((reset_mask & HW_APBX_CTRL0_RD()) == 0)
             break;
 if (retries == RESET_TIMEOUT)exit(1);
 // Setup dma channel configuration.
 BF_WRn(APBX_CHn_NXTCMDAR,I2C_CHANNEL_NUM,
        CMD_ADDR,(reg32_t) I2C_DMA_CMD1);
 BF_WR(APBX_CTRL1, CH3_CMDCMPLT_IRQ, 0);
 // Start the dma channel by incrementing semaphore.
 BF_WRn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, INCREMENT_SEMA, 1);
 // Poll for the semaphore to decrement to zero on the DMA channel.
 for (retries = 0; retries < SEMAPHORE_TIMEOUT; retries++){
      if (0 == BF_RDn(APBX_CHn_SEMA, I2C_CHANNEL_NUM, PHORE))
             break;
      if (1 == HW_I2C_CTRL1.MASTER_LOSS_IRQ)        return 1; // error
      if (1 == HW_I2C_CTRL1.NO_SLAVE_ACK_IRQ)       return 1; // error
      if (1 == HW_I2C_CTRL1.EARLY_TERM_IRQ)         return 1; // error
 }
 if(ferreters == SEMAPHORE_TIMEOUT) exit(2);
 // the 256 bytes were read from the eePROM so return with no Error
 return 0;
}
```

27.3 Internal Interface Modes

The I^{2}C functionality can be programmed and managed by the i.MX28 in three different modes. The description and examples above explain the DMA mode. The two additional modes are explained in the following subsections.

27.3.1 PIO Mode

The I^{2}C block on the i.MX28 supports a new PIO mode or soft-DMA mode. This mode works similar to the DMA mode except the processor is responsible for monitoring the HW_I2C_DEBUG0_DMAEREQ and HW_I2C_DEBUG0_DMAENDCMD pio bits. Also the DATA_ENGINE_CMPLT_IRQ interrupt can be enabled in the block to tell the processor when a DMA command is complete. (This is the same as the HW_I2C_DEBUG0_DMAENDCMD event.) For example, the DMA example Reading 256 bytes from an EEPROM can be executed using the CPU in PIO mode instead of using the DMA engine in DMA mode. (Be aware that the PIO_MODE bit used in previous generation SoCs is obsolete and has been removed in this version.) To execute the example in PIO mode the following basic steps should be followed:

1. Write to the HW_I2C_CTRL0 register with the desired field values to write the three byte address to the EEPROM just as it is embedded in the DMA descriptor. Set up other registers as needed, for example, HW_I2C_TIMING1 register.
2. Assert the HW_I2C_CTRL0_RUN bit.
3. Wait for the HW_I2C_DEBUG0_DMAREQ bit to assert.
4. Write the word containing the 3 bytes of EEPROM slave address to the HW_I2C_DATA register.
5. Clear the HW_I2C_DEBUG0_DMAREQ bit.
6. Wait for the HW_I2C_CTRL1_ DATA_ENGINE_CMPLT_IRQ bit to assert (or the interrupt to occur).
7. Clear the HW_I2C_CTRL1_ DATA_ENGINE_CMPLT_IRQ bit.
8. Write the next HW_I2C_CTRL0 word.
9. Reassert the HW_I2C_CTRL0_RUN bit.
10. Wait for the HW_I2C_DEBUG0_ DMAREQ bit to assert.

11. Write the read address to the HW_I2C_DATA register.
12. Clear the HW_I2C_DEBUG0_DMAREQ bit.
13. Wait for the HW_I2C_CTRL1_ DATA_ENGINE_CMPLT_IRQ bit to assert.
14. Clear the HW_I2C_CTRL1_ DATA_ENGINE_CMPLT_IRQ bit.
15. Write the next HW_I2C_CTRL0 word.
16. Reassert the HW_I2C_CTRL0_RUN bit.
17. Wait for the HW_I2C_DEBUG0_ DMAREQ bit to assert.
18. Get the data from the interface by reading the HW_I2C_DATA register.
19. Clear the HW_I2C_DEBUG0_DMAREQ bit.
20. Continue steps 17 through 19 until all data is read and the HW_I2C_CTRL1_ DATA_ENGINE_CMPLT_IRQ bit asserts.

This basic way of handshaking with the I^{2}C controller can be followed for reading or writing any amount of data from any slave device on the I^{2}C bus.

27.3.2 PIO Queue Mode

The PIO Queue mode is similar to the PIO mode described above except in this mode, control and data writes are queued up into internal FIFOs and executed when the HW_I2C_QUEUECTRL_QUEUE_RUN bit is set. There is one read and one write FIFO implemented in the system. Each FIFO is eight words deep. This mode is enabled by setting the HW_I2C_QUEUECTRL_PIO_QUEUE_MODE bit. This must be set before writing any control/command words or data. To write control words, the information is written to the HW_I2C_QUEUECMD register instead of the HW_I2C_CTRL register. This new register is very similar to the HW_I2C_CTRL and has almost all the same fields except for the soft reset and other block related fields. So essentially, the same control words in the PIO mode example above can be used in this mode but need to be diverted to this new control register. One difference from the other modes is that transfer data is written and read using different registers. The HW_I2C_DATA register is still used similar to the other modes but only for writing I^{2}C transfer data. The HW_I2C_QUEUEDATA register is used to read the receive transfer data is read from the block. As an example of how this mode works, consider the PIO mode execution steps above. PIO Queue mode works the same way with only slight differences. HW_I2C_QUEUECTRL_PIO_QUEUE_MODE bit must be set before any other writes to the CTRL or DATA registers are executed. And instead of kicking off a command and polling or waiting for an interrupt to sequence the writing or reading of data or starting a new command, all control and data writes to the registers are stored in the write FIFO. Multiple commands (control and supporting data) can be written back-to-back as long as the write FIFO is not full. The read FIFO is used only during an I^{2}C read. This data will also be queued up by the I^{2}C logic.

This FIFO data is read back through the HW_I2C_QUEUEDATA register. If there are multiple read data words in the queue, the next one will be immediately available after a read from the data register. So, it is possible for software to wait until multiple words are available and then do multiple or burst reads from the HW_I2C_QUEUEDATA register.

The HW_I2C_QUEUECTRL_QUEUE_RUN bit may be set after the commands are queued up or before. If the bit is set while the write FIFO is empty, then the logic will wait until the control word is written. Care must be taken that control words and data are written in the proper order as explained above; otherwise, commands will not execute properly.

The read and write FIFOs are managed with a watermark or FIFO threshold mechanism and an interrupt per FIFO. The HW_I2C_QUEUECTRL_WR_THRESH and HW_I2C_QUEUECTRL_RD_THRESH fields set the threshold values. These values are in units of words which is the basic unit of the FIFOs. For HW_I2C_QUEUECTRL_WR_THRESH, the associated interrupt bit will assert whenever the number of words in the FIFO is less than or equal to the threshold value. For HW_I2C_QUEUECTRL_RD_THRESH, the associated interrupt bit will assert whenever the number of words in the FIFO is greater than or equal to the threshold value. Note that the associated interrupt bits will assert whether or not the interrupt is enabled back to the processor. These threshold and interrupts will allow the CPU to be notified when the I^{2}C channel needs to be serviced and eliminates the need for software to poll for transaction status.

27.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

27.4.1 Pinmux Selection During Reset

For proper I^{2}C operation, the appropriate pinmux(s) must be selected before taking the block out of reset. Failure to select the I^{2}C pinmux selections before taking the block out of reset will cause the I^{2}C clock to operate incorrectly and will require another I^{2}C hardware reset.

27.4.1.1 Correct and Incorrect Reset Examples

Incorrect:

Clear I^{2}C SFTRST/CLKGATE
... Setup ...
I^{2}C PinMux Selections
*** I^{2}C will not operate.*

Correct:

I^{2}C PinMux Selections
Clear I^{2}C SFTRST/CLKGATE
... Setup ...
*** I^{2}C operates correctly.*

27.5 Programmable Registers

I2C Hardware Register Format Summary

I2C0 base address is 0x80058000; I2C2 base address is 0x8005A000

HW_I2C memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8005_8000	I2C Control Register 0 (HW_I2C_CTRL0)	32	R/W	C000_0000h	27.5.1/1753
8005_8010	I2C Timing Register 0 (HW_I2C_TIMING0)	32	R/W	0078_0030h	27.5.2/1755
8005_8020	I2C Timing Register 1 (HW_I2C_TIMING1)	32	R/W	0080_0030h	27.5.3/1756
8005_8030	I2C Timing Register 2 (HW_I2C_TIMING2)	32	R/W	0030_0030h	27.5.4/1757
8005_8040	I2C Control Register 1 (HW_I2C_CTRL1)	32	R/W	0886_0000h	27.5.5/1758
8005_8050	I2C Status Register (HW_I2C_STAT)	32	R	C000_0000h	27.5.6/1761
8005_8060	I2C Queue control reg. (HW_I2C_QUEUECTRL)	32	R/W	0000_0000h	27.5.7/1764
8005_8070	I2C Queue Status Register. (HW_I2C_QUEUESTAT)	32	R	0000_2020h	27.5.8/1766
8005_8080	I2C Queue command reg (HW_I2C_QUEUECMD)	32	R/W	0000_0000h	27.5.9/1767
8005_8090	I2C Controller Read Data Register for queue mode only. (HW_I2C_QUEUEDATA)	32	R	0000_0000h	27.5.10/1769
8005_80A0	I2C Controller DMA Read and Write Data Register (HW_I2C_DATA)	32	R/W	0000_0000h	27.5.11/1770
8005_80B0	I2C Device Debug Register 0 (HW_I2C_DEBUG0)	32	R/W	0010_0000h	27.5.12/1770
8005_80C0	I2C Device Debug Register 1 (HW_I2C_DEBUG1)	32	R/W	C000_0000h	27.5.13/1772

HW_I2C memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8005_80D0	I2C Version Register (HW_I2C_VERSION)	32	R	0104_0000h	27.5.14/1774

27.5.1 I2C Control Register 0 (HW_I2C_CTRL0)

The I2C Control Register specifies the reset state and the command and transfer size information for the I2C controller.

HW_I2C_CTRL0: 0x000

HW_I2C_CTRL0_SET: 0x004

HW_I2C_CTRL0_CLR: 0x008

HW_I2C_CTRL0_TOG: 0x00C

This register is either written by the DMA or the CPU depending on the state of an I2C transaction.

EXAMPLE

```
// turn off soft reset and clock gating
HW_I2C_CTRL0_CLR(BM_I2C_CTRL0_SFTRST | BM_I2C_CTRL0_CLKGATE);
```

Address: HW_I2C_CTRL0 – 8005_8000h base + 0h offset = 8005_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RUN	RSVD2	PRE_ACK	ACKNOWLEDGE	SEND_NAK_ON_LAST	RSVD1	MULTI_MASTER	CLOCK_HELD	RETAIN_CLOCK	POST_SEND_STOP	PRE_SEND_START	SLAVE_ADDRESS_ENABLE	MASTER_MODE	DIRECTION
W				▨				▨								
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	XFER_COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_CTRL0 field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state. 0x0 **RUN** — Allow I2C to operate normally. 0x1 **RESET** — Hold I2C in reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **RUN** — Allow I2C to operate normally. 0x1 **NO_CLKS** — Do not clock I2C gates in order to minimize power consumption.
29 RUN	Set this bit to one to enable the I2C Controller operation. This bit is automatically set by DMA commands that write to CTRL1 after the last PIO write of the DMA command. For Soft DMA operation, software can set this bit to enable the controller. 0x0 **HALT** — No I2C command in progress. 0x1 **RUN** — Process a slave or master I2C command.
28 RSVD2	Always set this bit field to zero.
27 PRE_ACK	Reserved for Freescale use.
26 ACKNOWLEDGE	Set this bit to one to cause a pending acknowledge bit (prior to DMA transfer) to be acknowledged. Set it to zero to NAK the pending acknowledge bit. This bit is set to one by the slave search engine if the criteria is met for acknowledging a slave address. Software can reset the bit to slave-not-acknowledge the address. This bit defines the state of the i2c_data line during the address acknowledge bit time. The slave search engine holds the clock at this point for a software decision. This bit has no effect when the presend start option is selected. 0x0 **SNAK** — slave not acknowledge when the held clock is released. 0x1 **ACK** — slave acknowledge when the held clock is released.
25 SEND_NAK_ON_ LAST	Set this bit to one to cause the DMA transfer engine to send a NAK on the last byte. 0x0 **ACK_IT** — Send an ACK on the last byte received. 0x1 **NAK_IT** — Send a NAK on the last byte received.
24 RSVD1	Always set this bit field to zero.
23 MULTI_MASTER	Set this bit to one to enable the master state machine to monitor the start conditions generated by other masters. The bus is assumed to be busy from the first start condition generated by another master until a stop condition is generated. 0x0 **SINGLE** — Assume we are the only master. 0x1 **MULTIPLE** — Enable multiple master bus busy monitoring from start detects.
22 CLOCK_HELD	This bit is set to one by the I2C controller state machines. It holds the I2C clock line low until cleared. It must be cleared by firmware, either by CPU instructions or DMA PIO transactions. It is set high when a slave address is matched by the slave controller. It is also set high at the end of a master or slave transaction that had the RETAIN_CLOCK bit set high. Software should not set this bit to one. 0x0 **RELEASE** — Release the clock line. 0x1 **HELD_LOW** — The clock line is currently being held low.
21 RETAIN_CLOCK	Set this bit to one to retain the clock at the end of this transaction. This has the effect of holding the clock low until the start of the next transaction. 0x0 **RELEASE** — Release the clock line after this data transfer. 0x1 **HOLD_LOW** — Hold the clock line low after this data transfer.

HW_I2C_CTRL0 field descriptions (continued)

Field	Description
20 POST_SEND_ STOP	Set this bit to one to send a stop condition after transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. 0x0 **NO_STOP** — Do not send a stop condition before this transaction. 0x1 **SEND_STOP** — Send a stop condition before this transaction.
19 PRE_SEND_ START	Set this bit to one to send a start condition before transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. 0x0 **NO_START** — Do not send a start condition before this transaction. 0x1 **SEND_START** — Send a start condition before this transaction.
18 SLAVE_ ADDRESS_ ENABLE	Set this bit to one to enable the slave address decoder. When an address match occurs, the I2C bus clock is frozen, by setting HW_I2C_CTRL0_CLOCK_HELD, and an interrupt is generated. 0x0 **DISABLED** — Disable the slave address decoder. 0x1 **ENABLED** — Enable the slave address decoder.
17 MASTER_MODE	Set this bit to one to select master mode. Set it zero to select slave mode. 0x0 **SLAVE** — Operate in slave mode. 0x1 **MASTER** — Operate in master mode.
16 DIRECTION	Set this bit to one to select an I2C transmit operation in either slave or master mode. XMIT = write in master mode, read in slave mode. Set this bit to zero to select an I2C receive operation in either slave or master mode. 0x0 **RECEIVE** — I2C receive operation for slave or master. 0x1 **TRANSMIT** — I2C transmit operation for slave or master.
15–0 XFER_COUNT	Number of bytes to transfer. This field decrements as bytes are transferred.

27.5.2 I2C Timing Register 0 (HW_I2C_TIMING0)

The timing for various phases of I2C Controller Commands are further defined by fields in the I2C Timing Register 0.

HW_I2C_TIMING0: 0x010

HW_I2C_TIMING0_SET: 0x014

HW_I2C_TIMING0_CLR: 0x018

HW_I2C_TIMING0_TOG: 0x01C

This register is primarily used for clock and timing generation.

EXAMPLE

```
HW_I2C_TIMING0_WR(0x00780030); // high time = 120 clocks, read bit at 48 for 95KHz at 24mhz
HW_I2C_TIMING0_WR(0x000F0007); // high time = 15 clocks, read bit at 7 for 400KHz at 24mhz
```

Address: HW_I2C_TIMING0 – 8005_8000h base + 10h offset = 8005_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2						HIGH_COUNT										RSVD1						RCV_COUNT									
W	▨	▨	▨	▨	▨	▨											▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

HW_I2C_TIMING0 field descriptions

Field	Description
31–26 RSVD2	Always set this bit field to zero.
25–16 HIGH_COUNT	Load this bit field with the APBX clock count for the high period of the I2C clock.
15–10 RSVD1	Always set this bit field to zero.
9–0 RCV_COUNT	Load this bit field with the APBX clock count for capturing read data after the I2C clock goes high.

27.5.3 I2C Timing Register 1 (HW_I2C_TIMING1)

The timing for various phases of I2C Controller Commands are further defined by fields in the I2C Timing Register 1.

HW_I2C_TIMING1: 0x020

HW_I2C_TIMING1_SET: 0x024

HW_I2C_TIMING1_CLR: 0x028

HW_I2C_TIMING1_TOG: 0x02C

This register is primarily used for clock and timing generation.

EXAMPLE

```
HW_I2C_TIMING1_WR(0x00800030); // low time at 128, write bit at 48 for 95 kHz at 24 MHz
HW_I2C_TIMING1_WR(0x001F000F); // low time at 31, write bit at 15 for 400 kHz at 24 MHz
```

Address: HW_I2C_TIMING1 – 8005_8000h base + 20h offset = 8005_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2						LOW_COUNT										RSVD1						XMIT_COUNT									
W	▨	▨	▨	▨	▨	▨											▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

HW_I2C_TIMING1 field descriptions

Field	Description
31–26 RSVD2	Always set this bit field to zero.
25–16 LOW_COUNT	Load this bit field with the APBX clock count for the low period of the I2C clock.
15–10 RSVD1	Always set this bit field to zero.
9–0 XMIT_COUNT	Load this bit field with the APBX clock count for changing transmitted data after the I2C clock goes low. Set this value to produce valid i2c setup and hold times at the desired bit rate for the current APBX clock rate.

27.5.4 I2C Timing Register 2 (HW_I2C_TIMING2)

The timing for various phases of I2C Controller Commands are further defined by fields in the I2C Timing Register 2.

HW_I2C_TIMING2: 0x030

HW_I2C_TIMING2_SET: 0x034

HW_I2C_TIMING2_CLR: 0x038

HW_I2C_TIMING2_TOG: 0x03C

This register is primarily used for clock and timing generation.

EXAMPLE

```
HW_I2C_TIMING2_WR(0x0015000d); //  bus free count of 21 lead in count of 13
```

Address: HW_I2C_TIMING2 – 8005_8000h base + 30h offset = 8005_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2						BUS_FREE										RSVD1						LEADIN_COUNT									
W	▨	▨	▨	▨	▨	▨											▨	▨	▨	▨	▨	▨										
Re-set	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

HW_I2C_TIMING2 field descriptions

Field	Description
31–26 RSVD2	Always set this bit field to zero.
25–16 BUS_FREE	Load this bit field with the APBX clock count for delaying the transition to the bus idle state after entering stop state in the clock generator.

HW_I2C_TIMING2 field descriptions (continued)

Field	Description
15–10 RSVD1	Always set this bit field to zero.
9–0 LEADIN_COUNT	Load this bit field with the APBX clock count for delaying the rising edge of i2c_sck after the kick.

27.5.5 I2C Control Register 1 (HW_I2C_CTRL1)

The I2C Controller Command is further defined by fields in this control extension register. The I2C Control Register 1 is where the I2C slave address is specified. Fast or normal mode is selected here.

HW_I2C_CTRL1: 0x040

HW_I2C_CTRL1_SET: 0x044

HW_I2C_CTRL1_CLR: 0x048

HW_I2C_CTRL1_TOG: 0x04C

This control register is primarily used for interrupt management. It also controls the special slave address matching mode. In addition, it controls the protocol speed, i.e fast or 400KHz versus normal or 100KHz operation.

EXAMPLE

```
HW_I2C_CTRL1_CLR(BM_I2C_CTRL1_SLAVE_IRQ); // clear the slave interrupt
```

Address: HW_I2C_CTRL1 – 8005_8000h base + 40h offset = 8005_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1	RD_QUEUE_IRQ	WR_QUEUE_IRQ	CLR_GOT_A_NAK	ACK_MODE	FORCE_DATA_IDLE	FORCE_CLK_IDLE	BCAST_SLAVE_EN	SLAVE_ADDRESS_BYTE							
W	▨															
Reset	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUS_FREE_IRQ_EN	DATA_ENGINE_CMPLT_IRQ_EN	NO_SLAVE_ACK_IRQ_EN	OVERSIZE_XFER_TERM_IRQ_EN	EARLY_TERM_IRQ_EN	MASTER_LOSS_IRQ_EN	SLAVE_STOP_IRQ_EN	SLAVE_IRQ_EN	BUS_FREE_IRQ	DATA_ENGINE_CMPLT_IRQ	NO_SLAVE_ACK_IRQ	OVERSIZE_XFER_TERM_IRQ	EARLY_TERM_IRQ	MASTER_LOSS_IRQ	SLAVE_STOP_IRQ	SLAVE_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_CTRL1 field descriptions

Field	Description
31 RSVD1	Always set this bit field to zero.
30 RD_QUEUE_IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller because the read queue threshold criterion has been met. This bit is cleared by software by writing a one to its SCT clear address. Note that the enable bit for this interrupt is in the HW_I2C_QUEUECTRL register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
29 WR_QUEUE_ IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller because the write queue threshold criterion has been met. This bit is cleared by software by writing a one to its SCT clear address. Note that the enable bit for this interrupt is in the HW_I2C_QUEUECTRL register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
28 CLR_GOT_A_ NAK	Setting this bit will clear the got_a_nak. 0x0 **DO_NOTHING** — 0x1 **CLEAR** — Clear got_a_nak.
27 ACK_MODE	This setting affects the behavior of the ACK pulse when RETAIN_CLOCK=1. 0x0 **ACK_AFTER_HOLD_LOW** — ACK will occur after clock is held low at start of next access. 0x1 **ACK_BEFORE_HOLD_LOW** — ACK will occur at end of access before clock is held low.
26 FORCE_DATA_ IDLE	Writing a one to this bit will force the data state machine to return to its idle state and stay there.
25 FORCE_CLK_ IDLE	Writing a one to this bit will force the clock generator state machine to return to its idle state and stay there.
24 BCAST_SLAVE_ EN	Set this bit to one to enable the slave address search machine to look for both a match to the programmed slave address as well as a match to the broadcast address of all zeroes. 0x0 **NO_BCAST** — Do not watch for broadcast address while matching programmed slave address. 0x1 **WATCH_BCAST** — Watch for the all zeroes broadcast address while matching programmed slave address.
23–16 SLAVE_ ADDRESS_BYTE	Slave address byte, note the slave address is only seven bits long. The slave address search state machine will respond to either a read or a write command issued to the seven bit address. Set the LSB (bit 0) to one to match ALL 7 bit i2c addresses.
15 BUS_FREE_ IRQ_EN	Set this bit to one to enable bus free interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
14 DATA_ENGINE_ CMPLT_IRQ_EN	Set this bit to one to enable data engine complete interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
13 NO_SLAVE_ ACK_IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller.

HW_I2C_CTRL1 field descriptions (continued)

Field	Description
	0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
12 OVERSIZE_ XFER_TERM_ IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
11 EARLY_TERM_ IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
10 MASTER_LOSS_ IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
9 SLAVE_STOP_ IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
8 SLAVE_IRQ_EN	Set this bit to one to enable interrupt requests to be routed to the interrupt collector. Set to zero to disable interrupts from the I2C controller. The corresponding HW_I2C_CTRL1_SLAVE_IRQ interrupt bit is set by the slave search engine to indicate that it has stopped searching due to an address match or error. 0x0 **DISABLED** — No Interrupt Request Pending. 0x1 **ENABLED** — Interrupt Request Pending.
7 BUS_FREE_IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller because the bus has become free. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that the I2C bus, which was busy, has just become free. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
6 DATA_ENGINE_ CMPLT_IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller because the data engine transfer has completed. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that the data engine has completed a DMA transfer in either master or slave mode. This notification is useful for pio mode master write (transmit) or slave read (transmit) operations, i.e., data engine transmit operations. PIO receive operations are not supported. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
5 NO_SLAVE_ ACK_IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller because the slave addressed by a master transfer did not respond with an acknowledge. This bit is cleared by software by writing a one to its SCT clear address. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.

HW_I2C_CTRL1 field descriptions (continued)

Field	Description
4 OVERSIZE_ XFER_TERM_ IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master DMA transfer did not complete by the end of the transfer size. This is indicated by the slave acknowledging the last byte of a write transfer instead of NAKing it. The master should then send additional bytes of data if desired. This interrupt is only used in slave mode. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
3 EARLY_TERM_ IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master write transfrom from the STMP38xx to a slave device was NAKed by the slave before the transfer was completed. In slave mode, it indicates that the master NAKed a byte transmitted by the slave causing early termination of the expected transfer. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
2 MASTER_LOSS_ IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This interrupt indicates that a master read or write transaction lost an arbitration with another master. Master loss is indicated by the master attempting to transmit a one to the bus at the same time as another master writes a zero. The wired and bus produces a zero on the bus which is detected by the lossing master. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
1 SLAVE_STOP_ IRQ	This bit is set to indicate that an I2C Stop Condition was received by the slave address search engine after it had found a start command addressed to its slave address. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
0 SLAVE_IRQ	This bit is set to indicate that an interrupt is requested by the I2C controller. This bit is cleared by software by writing a one to its SCT clear address. This bit is set by the slave search engine to indicate that it has stopped searching due to an address match or error. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.

27.5.6 I2C Status Register (HW_I2C_STAT)

The I2C Controller reports status information in the I2C Status Register.

The status register provides read-only access to the function presence bits, as well as the busy indicators for the slave and master state machines.

EXAMPLE

```
while(HW_I2C_STAT.SLAVE_BUSY != BV_I2C_STAT_SLAVE_BUSY__IDLE_VAL);// then wait till it finishes
```

Address: HW_I2C_STAT – 8005_8000h base + 50h offset = 8005_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MASTER_ PRESENT	SLAVE_ PRESENT	ANY_ ENABLED_IRQ	GOT_A_NAK	RSVD1				RCVD_SLAVE_ADDR							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SLAVE_ ADDR_EQ_ ZERO	SLAVE_ FOUND	SLAVE_ SEARCHING	DATA_ ENGINE_ DMA_WAIT	BUS_ BUSY	CLK_GEN_ BUSY	DATA_ ENGINE_ BUSY	SLAVE_ BUSY	BUS_FREE_ IRQ_ SUMMARY	DATA_ENGINE_ CMPLT_IRQ_ SUMMARY	NO_SLAVE_ ACK_IRQ_ SUMMARY	OVERSIZE_ XFER_TERM_ IRQ_SUMMARY	EARLY_TERM_ IRQ_SUMMARY	MASTER_LOSS_ IRQ_SUMMARY	SLAVE_STOP_ IRQ_SUMMARY	SLAVE_IRQ_ SUMMARY
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_STAT field descriptions

Field	Description
31 MASTER_ PRESENT	This read-only bit indicates that the I2C master function is present when it reads back a one. This I2C function is not available on a device that returns a zero for this bit field. 0x0 **UNAVAILABLE** — I2C is not present in this product. 0x1 **AVAILABLE** — I2C is present in this product.
30 SLAVE_ PRESENT	This read-only bit indicates that the I2C slave function is present when it reads back a one. This I2C function is not available on a device that returns a zero for this bit field. 0x0 **UNAVAILABLE** — I2C is not present in this product. 0x1 **AVAILABLE** — I2C is present in this product.
29 ANY_ENABLED_ IRQ	This read-only bit indicates that the I2C controller has at least one enable interrupt requesting service. It is the logic OR of all of the IRQ summary bits. 0x0 **NO_REQUESTS** — No enabled interrupts are requesting service. 0x1 **AT_LEAST_ONE_REQUEST** — At least one of the summary interrupt bits is set.
28 GOT_A_NAK	Read-only view of the got-a-nak signal. 0x0 **NO_NAK** — I2C master has not detected a NAK. 0x1 **DETECTED_NAK** — I2C master has detected a NAK.
27–24 RSVD1	Always set this bit field to zero.
23–16 RCVD_SLAVE_ ADDR	This read-only byte indicates that the state of the slave I2C address byte received, including the read/write bit received from an address byte that matched our slave address.

HW_I2C_STAT field descriptions (continued)

Field	Description
15 SLAVE_ADDR_ EQ_ZERO	This read-only bit indicates that the I2C slave function was searching for a transaction that matches the current slave address. When set to one, it indicates that an address match was found for the exact adderss 0x00. 0x0 **ZERO_NOT_MATCHED** — I2C slave search did not match a zero. 0x1 **WAS_ZERO** — I2C has found an address match against address 0x00.
14 SLAVE_FOUND	This read-only bit indicates that the I2C slave function was searching for a transaction that matches the current slave address. When set to one, it indicates that an address match was found and the I2C clock is frozen by the slave search. This bit is cleared by starting the appropriate slave DMA transfer or restarting a slave search. 0x0 **IDLE** — I2C slave search is idle. 0x1 **WAITING** — I2C has found an address match and is holding the I2C clock line low.
13 SLAVE_ SEARCHING	This read-only bit indicates that the I2C slave function is searching for a transaction that matches the current slave address. 0x0 **IDLE** — I2C slave search is idle. 0x1 **ACTIVE** — I2C is actively searching for an address match.
12 DATA_ENGINE_ DMA_WAIT	This read-only bit is set to one when the data engine is waitng for data from a DMA device. This bit can be used to transmit short I2C transactions without using a DMA channel. This generally works for up to three data bytes transmitted with one address byte. 0x0 **CONTINUE** — I2C master is not waiting on data from the DMA. 0x1 **WAITING** — I2C master is waiting on data from the DMA.
11 BUS_BUSY	This read-only bit indicates that the I2C bus is busy with a transaction. It is set by a start condition and reset by a detected stop condition. 0x0 **IDLE** — I2C bus is idle, i.e. reset state or at least one stop condition detected. 0x1 **BUSY** — I2C bus is busy, i.e. at least one start condition has been detected.
10 CLK_GEN_BUSY	This read-only bit indicates that the I2C clock generator is busy with a transaction. 0x0 **IDLE** — I2C clock generator is idle. 0x1 **BUSY** — I2C clock generator is busy performing a command.
9 DATA_ENGINE_ BUSY	This read-only bit indicates that the I2C data transfer engine is busy with a data transmit or recieve opertion. In addition, it can be busy, as a master, sending a start or stop condition. 0x0 **IDLE** — I2C Data Engine is idle. 0x1 **BUSY** — I2C is Data Engine busy performing a data transfer.
8 SLAVE_BUSY	This read-only bit indicates that the I2C slave address search engine is busy with a transaction. This bit will go high when an address search is started and will remain high until the slave search engine returns to its idle state. 0x0 **IDLE** — I2C slave search engine is idle. 0x1 **BUSY** — I2C slave search engine is busy searching for an address match.
7 BUS_FREE_ IRQ_SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
6 DATA_ENGINE_ CMPLT_IRQ_ SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit.

HW_I2C_STAT field descriptions (continued)

Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
5 NO_SLAVE_ ACK_IRQ_ SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
4 OVERSIZE_ XFER_TERM_ IRQ_SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
3 EARLY_TERM_ IRQ_SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
2 MASTER_LOSS_ IRQ_SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
1 SLAVE_STOP_ IRQ_SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
0 SLAVE_IRQ_ SUMMARY	This bit is set to indicate that an interrupt is requested by the I2C controller. It is a logical AND of the corresponding interrupt status bit and interrupt enable bit. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.

27.5.7 I2C Queue control reg. (HW_I2C_QUEUECTRL)

The I2C Control Register implements the controls for I2C PIO Queue mode.

HW_I2C_QUEUECTRL: 0x060

HW_I2C_QUEUECTRL_SET: 0x064

HW_I2C_QUEUECTRL_CLR: 0x068

HW_I2C_QUEUECTRL_TOG: 0x06C

Address: HW_I2C_QUEUECTRL – 8005_8000h base + 60h offset = 8005_8060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3											RD_THRESH				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2			WR_THRESH					RSVD1		QUEUE_RUN	RD_CLEAR	WR_CLEAR	PIO_ QUEUE_ MODE	RD_QUEUE_ IRQ_EN	WR_ QUEUE_ IRQ_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_QUEUECTRL field descriptions

Field	Description
31–21 RSVD3	Always set this bit field to zero.
20–16 RD_THRESH	This field specified the threshold value of the number of read queue words, or more, that should be in the queue before a read queue interrupt is generated to the CPU. The valid range for this field is 0-7. A value of 0 indicates a threshold of an empty queue.
15–13 RSVD2	Always set this bit field to zero.
12–8 WR_THRESH	This field specified the threshold value of the number of write queue words, or less, that should be in the queue before a write queue interrupt is generated to the CPU. The valid range for this field is 0-8. A value of 0 indicates a threshold of an empty queue.
7–6 RSVD1	Always set this bit field to zero.
5 QUEUE_RUN	Asserting this bit essentially tells the system to begin executing commands and data that are in the queue. This allows software to kick off a series of commands when it chooses. 0x0 **STOP** — Don't process commands or stop after processing the currently executing command. 0x1 **START** — Start processing commands.
4 RD_CLEAR	Asserting this bit clears the read queue that holds data read from the I2C port.
3 WR_CLEAR	Asserting this bit clears the write queue that holds commands and data written through the QUEUECTRL and DATA registers.
2 PIO_QUEUE_ MODE	When this bit is set writes to the QUEUECTRL and DATA registers are queued up. When the HW_I2C_QUEUECTRL_QUEUE_RUN bit is later set, all the commands (and associated data) will be executed in the order they are written.
1 RD_QUEUE_ IRQ_EN	Set this bit to one to enable receiving interrupts from the RD_QUEUE_IRQ source to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — Interrupt source disabled. 0x1 **ENABLED** — Interrupt source enabled.

HW_I2C_QUEUECTRL field descriptions (continued)

Field	Description
0 WR_QUEUE_ IRQ_EN	Set this bit to one to enable receiving interrupts from the WR_QUEUE_IRQ source to the interrupt collector. Set to zero to disable interrupts from the I2C controller. 0x0 **DISABLED** — Interrupt source disabled. 0x1 **ENABLED** — Interrupt source enabled.

27.5.8 I2C Queue Status Register. (HW_I2C_QUEUESTAT)

This register exposes the state of the internal read and write queues used in PIO Queue mode.

HW_I2C_QUEUESTAT: 0x070

HW_I2C_QUEUESTAT_SET: 0x074

HW_I2C_QUEUESTAT_CLR: 0x078

HW_I2C_QUEUESTAT_TOG: 0x07C

This information can be useful during debug or by the CPU during normal operation.

Address: HW_I2C_QUEUESTAT – 8005_8000h base + 70h offset = 8005_8070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2 [15:15]	RD_ QUEUE_ FULL	RD_ QUEUE_ EMPTY	RD_QUEUE_CNT					RSVD1	WR_ QUEUE_ FULL	WR_ QUEUE_ EMPTY	WR_QUEUE_CNT				
W																
Reset	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0

HW_I2C_QUEUESTAT field descriptions

Field	Description
31–15 RSVD2	Always set this bit field to zero.
14 RD_QUEUE_ FULL	A read-only view of the read queue full signal.
13 RD_QUEUE_ EMPTY	A read-only view of the read queue empty signal.
12–8 RD_QUEUE_ CNT	A read-only view of how many words are currently in the read queue.
7 RSVD1	Always set this bit field to zero.
6 WR_QUEUE_ FULL	A read-only view of the write queue full signal.
5 WR_QUEUE_ EMPTY	A read-only view of the write queue empty signal.
4–0 WR_QUEUE_ CNT	A read-only view of how many words are currently in the write queue.

27.5.9 I2C Queue command reg (HW_I2C_QUEUECMD)

The I2C Command Register accepts commands that are to be queued up for later execution in PIO queue mode.

HW_I2C_QUEUECMD: 0x080

HW_I2C_QUEUECMD_SET: 0x084

HW_I2C_QUEUECMD_CLR: 0x088

HW_I2C_QUEUECMD_TOG: 0x08C

This register is either written by the CPU to queue up I2C transaction commands for later execution.

Address: HW_I2C_QUEUECMD – 8005_8000h base + 80h offset = 8005_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2				PRE_ ACK	ACKNOWLEDGE	SEND_NAK_ON_ LAST	RSVD1	MULTI_MASTER	CLOCK_HELD	RETAIN_CLOCK	POST_SEND_ STOP	PRE_SEND_ START	SLAVE_ ADDRESS_ ENABLE	MASTER_MODE	DIRECTION
W	▨	▨	▨	▨				▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	XFER_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_QUEUECMD field descriptions

Field	Description
31–28 RSVD2	Always set this bit field to zero.
27 PRE_ACK	Reserved for Freescale use.
26 ACKNOWLEDGE	Set this bit to one to cause a pending acknowledge bit (prior to DMA transfer) to be acknowledged. Set it to zero to NAK the pending acknowledge bit. This bit is set to one by the slave search engine if the criteria is met for acknowledging a slave address. Software can reset the bit to slave-not-acknowledge the address. This bit defines the state of the i2c_data line during the address acknowledge bit time. The slave search engine holds the clock at this point for a software decision. This bit has no effect when the presend start option is selected. 0x0 **SNAK** — slave not acknowledge when the held clock is released. 0x1 **ACK** — slave acknowledge when the held clock is released.
25 SEND_NAK_ON_ LAST	Set this bit to one to cause the DMA transfer engine to send a NAK on the last byte. 0x0 **ACK_IT** — Send an ACK on the last byte received. 0x1 **NAK_IT** — Send a NAK on the last byte received.
24 RSVD1	Always set this bit field to zero.
23 MULTI_MASTER	Set this bit to one to enable the master state machine to monitor the start conditions generated by other masters. The bus is assumed to be busy from the first start condition generated by another master until a stop condition is generated. 0x0 **SINGLE** — Assume we are the only master. 0x1 **MULTIPLE** — Enable multiple master bus busy monitoring from start detects.
22 CLOCK_HELD	This bit is set to one by the I2C controller state machines. It holds the I2C clock line low until cleared. It must be cleared by firmware, either by CPU instructions or DMA PIO transactions. It is set high when a slave address is matched by the slave controller. It is also set high at the end of a master or slave transaction that had the RETAIN_CLOCK bit set high. Software should not set this bit to one. 0x0 **RELEASE** — Release the clock line. 0x1 **HELD_LOW** — The clock line is currently being held low.

HW_I2C_QUEUECMD field descriptions (continued)

Field	Description
21 RETAIN_CLOCK	Set this bit to one to retain the clock at the end of this transaction. This has the effect of holding the clock low until the start of the next transaction. 0x0 **RELEASE** — Release the clock line after this data transfer. 0x1 **HOLD_LOW** — Hold the clock line low after this data transfer.
20 POST_SEND_ STOP	Set this bit to one to send a stop condition after transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. 0x0 **NO_STOP** — Do not send a stop condition before this transaction. 0x1 **SEND_STOP** — Send a stop condition before this transaction.
19 PRE_SEND_ START	Set this bit to one to send a start condition before transferring the data associated with this transaction. This bit is automatically cleared by the hardware after the operation has been performed. 0x0 **NO_START** — Do not send a start condition before this transaction. 0x1 **SEND_START** — Send a start condition before this transaction.
18 SLAVE_ ADDRESS_ ENABLE	Set this bit to one to enable the slave address decoder. When an address match occurs, the I2C bus clock is frozen, by setting HW_I2C_CTRL0_CLOCK_HELD, and an interrupt is generated. 0x0 **DISABLED** — Disable the slave address decoder. 0x1 **ENABLED** — Enable the slave address decoder.
17 MASTER_MODE	Set this bit to one to select master mode. Set it zero to select slave mode. 0x0 **SLAVE** — Operate in slave mode. 0x1 **MASTER** — Operate in master mode.
16 DIRECTION	Set this bit to one to select an I2C transmit operation in either slave or master mode. XMIT = write in master mode, read in slave mode. Set this bit to zero to select an I2C receive operation in either slave or master mode. 0x0 **RECEIVE** — I2C receive operation for slave or master. 0x1 **TRANSMIT** — I2C transmit operation for slave or master.
15–0 XFER_COUNT	Number of bytes to transfer. This field decrements as bytes are transferred.

27.5.10 I2C Controller Read Data Register for queue mode only. (HW_I2C_QUEUEDATA)

For queue mode, I2C transfer data is read from this register.

Transfer data is read from this register only when in queue mode. This register is somewhat analogous to the HW_I2C_DATA register. This is a read only register.

Address: HW_I2C_QUEUEDATA – 8005_8000h base + 90h offset = 8005_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_QUEUEDATA field descriptions

Field	Description
31–0 DATA	Software should read from this address to retreive data from I2C read operations when .

27.5.11 I2C Controller DMA Read and Write Data Register (HW_I2C_DATA)

The I2C Controller DMA Read and Write Data Register is the target for both source and destination DMA and PIO transfers.

DMA reads and writes are directed to this register.

EXAMPLE

```
The DMA data register is used by the DMA to read or write data from the I2C controller, as
mediated by the I2C controller's DMA request signal.
```

Address: HW_I2C_DATA – 8005_8000h base + A0h offset = 8005_80A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_DATA field descriptions

Field	Description
31–0 DATA	The source DMA channel writes to this address. The Destination DMA channel reads from this address.

27.5.12 I2C Device Debug Register 0 (HW_I2C_DEBUG0)

The I2C Device Debug Register 0 provides a diagnostic view into various internal states and controls.

HW_I2C_DEBUG0: 0x0b0

HW_I2C_DEBUG0_SET: 0x0b4

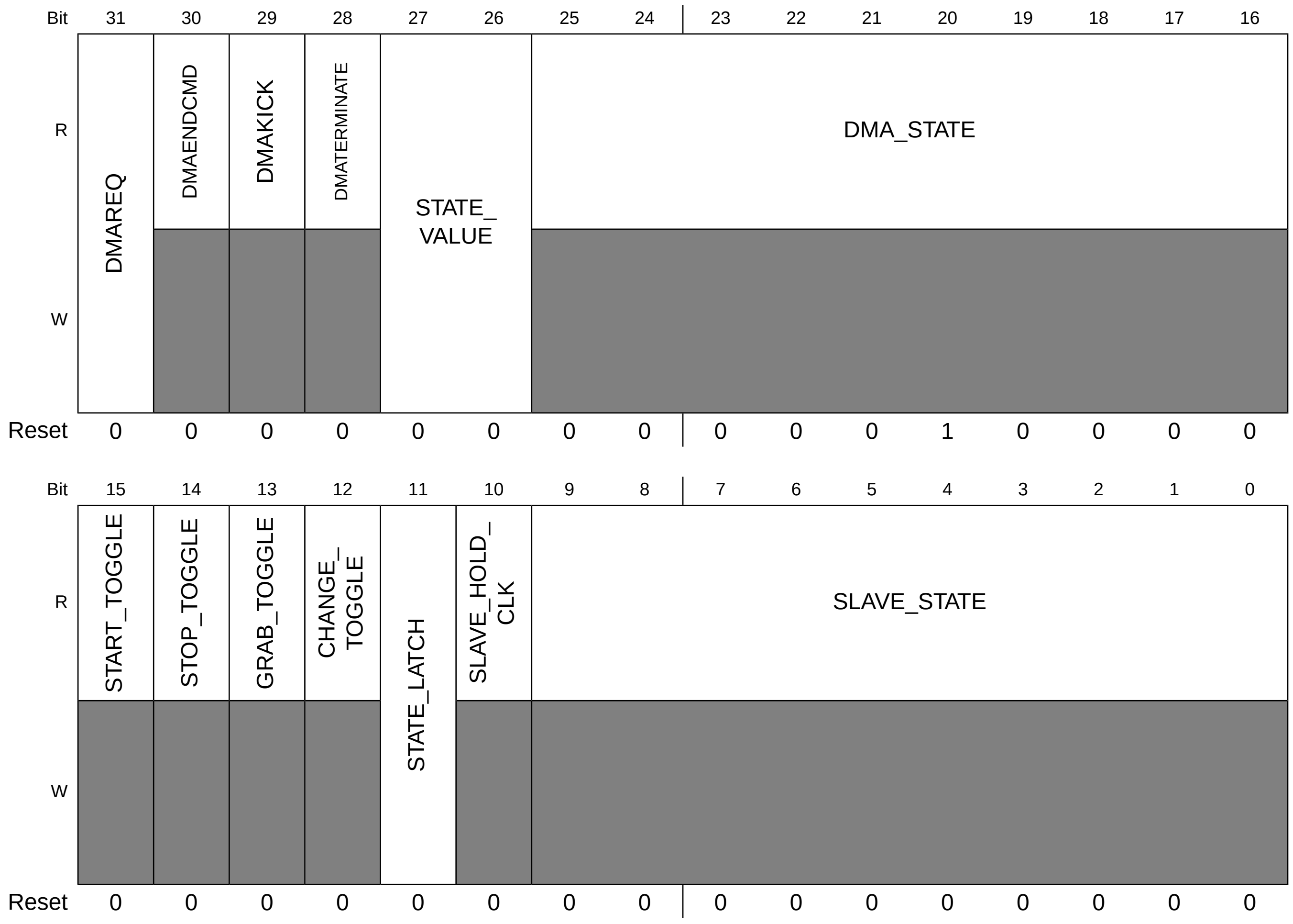
HW_I2C_DEBUG0_CLR: 0x0b8

HW_I2C_DEBUG0_TOG: 0x0bC

EXAMPLE

```
while(HW_I2C_DEBUG0.DMAREQ == old_dma_req_value); // wait for next dma request toggle
old_dma_req_value = HW_I2C_DEBUG0.DMAREQ;         // remember the new state of the dma request
 toggle
```

Address: HW_I2C_DEBUG0 – 8005_8000h base + B0h offset = 8005_80B0h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
DMAREQ
DMAENDCMD
DMAKICK
DMATERMINATE
STATE_ VALUE
DMA_STATE
Reset 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
START_TOGGLE
STOP_TOGGLE
GRAB_TOGGLE
CHANGE_ TOGGLE
STATE_LATCH
SLAVE_HOLD_ CLK
SLAVE_STATE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_I2C_DEBUG0 field descriptions

Field	Description
31 DMAREQ	The DMA request signal has toggled, active high.

HW_I2C_DEBUG0 field descriptions (continued)

Field	Description
30 DMAENDCMD	Read-only view of the toggle state of the DMA End Command signal.
29 DMAKICK	Read-only view of the toggle state of the DMA Kick signal.
28 DMATERMINATE	Read-only view of the toggle state of the DMA Terminate signal.
27–26 STATE_VALUE	This field contains the lower two bit values of the DMA state machine variable. This value is fixed or free-running based on the STATE_LATCH bit in this register. This field is used for debug purposes.
25–16 DMA_STATE	Current state of the DMA state machine.
15 START_TOGGLE	Read-only view of the start detector. Toggles once for each detected start condition.
14 STOP_TOGGLE	Read-only view of the stop detector. Toggles once for each detected stop condition.
13 GRAB_TOGGLE	Read-only view of the grab receive data timing point. Toggles once for each read timing point, as delayed from rising clock.
12 CHANGE_ TOGGLE	Read-only view of the change xmit data timing point. Toggles once for each change xmit data timing point, as delayed from falling clock.
11 STATE_LATCH	Changing this bit from a 1 to 0 latches the lower two bits of the current DMA state machine variable into the STATE_VALUE field of this register. A value of 1 causes the STATE_VALUE field to be free-runing.
10 SLAVE_HOLD_ CLK	Current State of the Slave Address Search FSM clock hold register.
9–0 SLAVE_STATE	Current State of the Slave Address Search FSM.

27.5.13 I2C Device Debug Register 1 (HW_I2C_DEBUG1)

The I2C Device Debug Register 1 provides a diagnostic view of the external bus and provides OE control for the clock and data.

HW_I2C_DEBUG1: 0x0c0

HW_I2C_DEBUG1_SET: 0x0c4

HW_I2C_DEBUG1_CLR: 0x0c8

HW_I2C_DEBUG1_TOG: 0x0cC

EXAMPLE

```
while(HW_I2C_DEBUG1.I2C_CLK_IN == 0); // wait for I2C clock line to go high
```

Address: HW_I2C_DEBUG1 – 8005_8000h base + C0h offset = 8005_80C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	I2C_ CLK_ IN	I2C_ DATA_ IN	RSVD4		DMA_BYTE_ENABLES				CLK_GEN_STATE							
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2					LST_MODE		LOCAL_ SLAVE_ TEST	RSVD1			FORCE_ CLK_ON	FORCE_ ARB_LOSS	FORCE_ RCV_ACK	FORCE_I2C_ DATA_OE	FORCE_I2C_ CLK_OE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_DEBUG1 field descriptions

Field	Description
31 I2C_CLK_IN	A copy of the pad input signal for the I2C clock pad.
30 I2C_DATA_IN	A copy of the pad input signal for the I2C data pad.
29–28 RSVD4	Always set this bit field to zero.
27–24 DMA_BYTE_ ENABLES	A read-only view of the byte enables for HW_I2C_DATA register writes. These bits are used in the I2C DMA state machine to track the number of bytes written by the DMA. Individual bits are cleared as they are consummed.
23–16 CLK_GEN_ STATE	A read-only view on current state of clock generation state machine.
15–11 RSVD2	Always set this bit field to zero.
10–9 LST_MODE	When in local slave test mode, this bit field defines the type of address generated for the slave. 0x0 **BCAST** — Broadcast, i.e. i2c address 0x00. 0x1 **MY_WRITE** — Send to my slave address with a RW bit equal 0. 0x2 **MY_READ** — Send to my slave address with a RW bit equal 1. 0x3 **NOT_ME** — Send to an address that is not mine, i.e. bit four is complemented.
8 LOCAL_SLAVE_ TEST	Writting a one to this bit places the slave in local test mode. one of three slave address can be sent in either read or write mode.
7–5 RSVD1	Always set this bit field to zero.
4 FORCE_CLK_ON	Writing a one to this bit will force the clock generator to send a continuous stream of clocks on the I2C bus.

HW_I2C_DEBUG1 field descriptions (continued)

Field	Description
3 FORCE_ARB_ LOSS	Writing a one to this bit will force the appearance of an arbitration loss on the next one a master attempts to transmit.
2 FORCE_RCV_ ACK	Writing a one to this bit will force the appearance of a receive acknowledge to the byte level state machine at bit 9 of the transfer.
1 FORCE_I2C_ DATA_OE	Writting a one to this bit will force an output enable at the pad. The pad data line is tied to zero. Thus the I2C data line will either be hi-z or zero.
0 FORCE_I2C_ CLK_OE	Writing a one to this bit will force an output enable at the pad. The pad data line is tied to zero. Thus the I2C clock line will either be hi-z or zero.

27.5.14 I2C Version Register (HW_I2C_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_I2C_VERSION.B.MAJOR != 1) Error();
```

Address: HW_I2C_VERSION – 8005_8000h base + D0h offset = 8005_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_I2C_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 28
Pulse-Width Modulator (PWM) Controller

28.1 Pulse Width Modulator (PWM) Overview

The device has eight PWM output controllers that can be used in place of GPIO pins. Applications include HSADC driving signals and LED & backlight brightness control. Independent output control of each phase allows 0, 1, or high-impedance to be independently selected for the active and inactive phases. Individual outputs can be run in lock step with guaranteed non-overlapping portions for differential drive applications.

Figure 28-1 shows the block diagram of the PWM controller. The controller does not use DMA. Initial values of Period, Active, and Inactive widths are set for each desired channel. The outputs are selected by phase and then the desired PWM channels are simultaneously enabled. This effectively launches the PWM outputs to autonomously drive their loads without further intervention.

28.2 Operation

Each PWM channel has two control registers that are used to specify the channel output: HW_PWM_ACTIVEn and HW_PWM_PERIODn.

When programming a channel, it is important to remember that there is an order dependence for register writes.

- The HW_PWM_ACTIVEn register must be written first, followed by HW_PWM_PERIODn.
- If the order is reversed, the parameters written to the HW_PWM_ACTIVEn register will not take effect in the hardware.

The hardware waits for a HW_PWM_PERIODn register write to update the hardware with the values in both registers. This register write order dependence allows smooth on-the-fly reprogramming of the channel. Also, when the user reprograms the channel in this manner, the new register values will not take effect until the beginning of a new output period. This eliminates the potential for output glitches that could occur if the registers were updated while the channel was enabled and in the middle of a cycle.

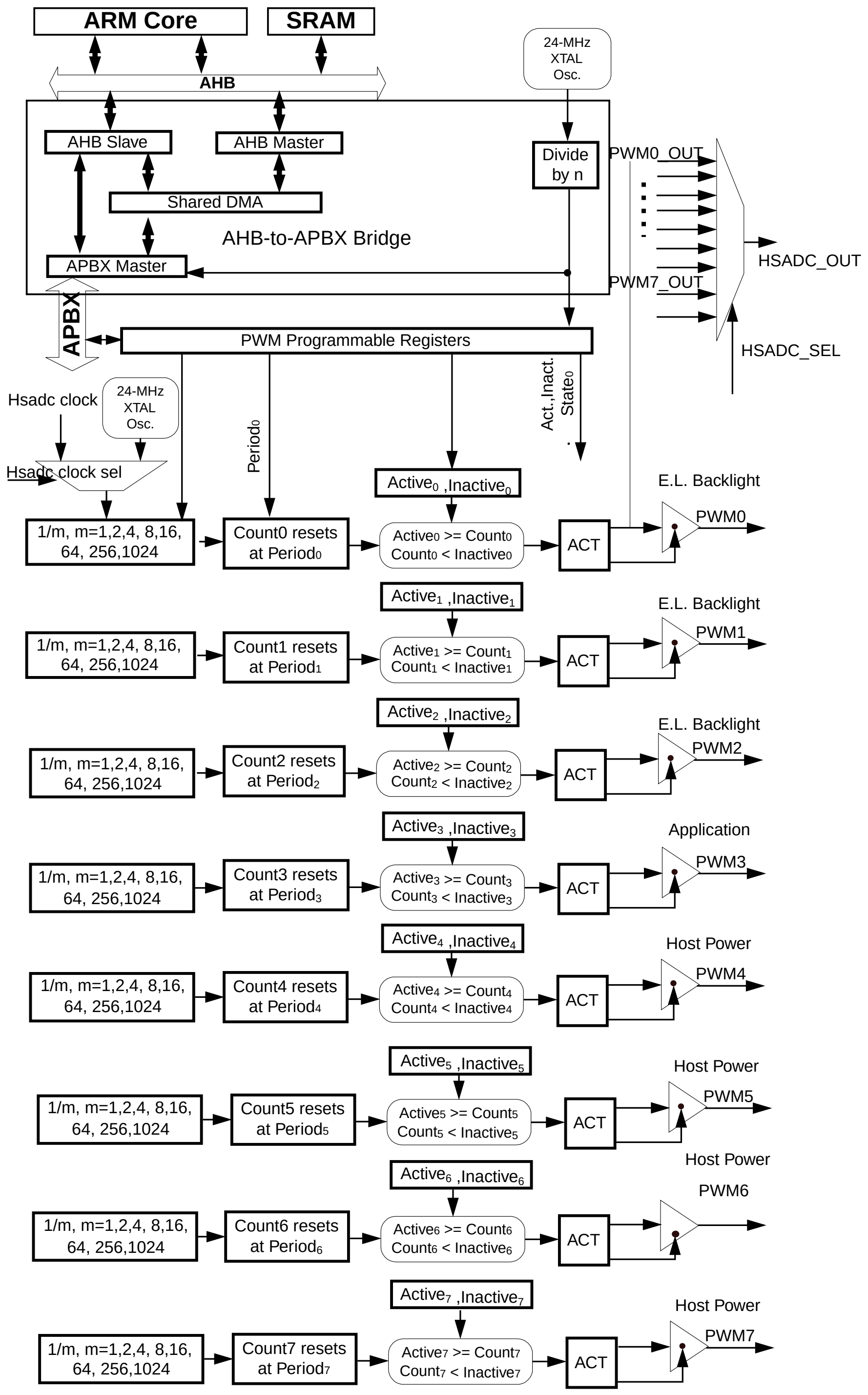


Figure 28-1. Pulse-Width Modulation Controller (PWM) Block Diagram

Each channel has a dedicated internal 16-bit counter that increments once for each divided clock period presented from the clock divider.

- The internal counter resets when it reaches the value stored in the channel control registers, for example, HW_PWM_PERIOD0_PERIOD.
- The Active flip-flop is set to 1 when the internal counter reaches the value stored in HW_PWM_ACTIVE0_ACTIVE.
- It remains high until the internal counter exceeds the value stored in HW_PWM_ACTIVE0_INACTIVE.

These two values define the starting and ending points for the logically *"active"* portion of the waveform. As shown in Figure 28-2, the actual state on the output for each phase, for example, active or inactive, is completely controlled by the active and inactive state values in the channel control registers.

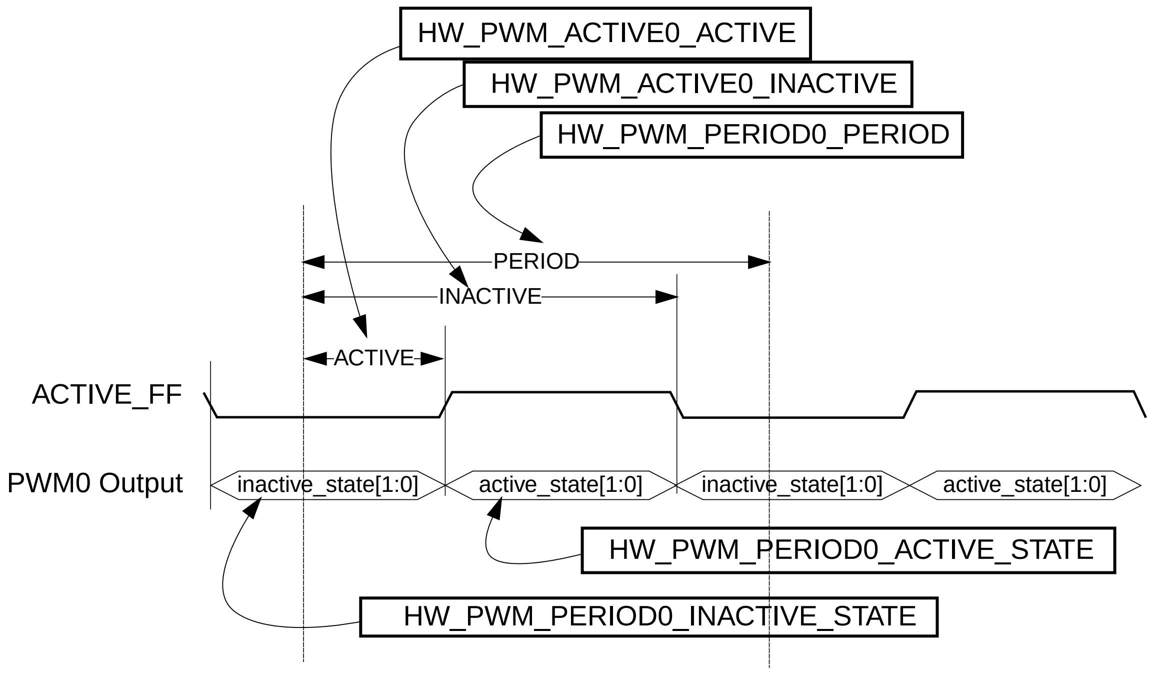


Figure 28-2. PWM Output Example

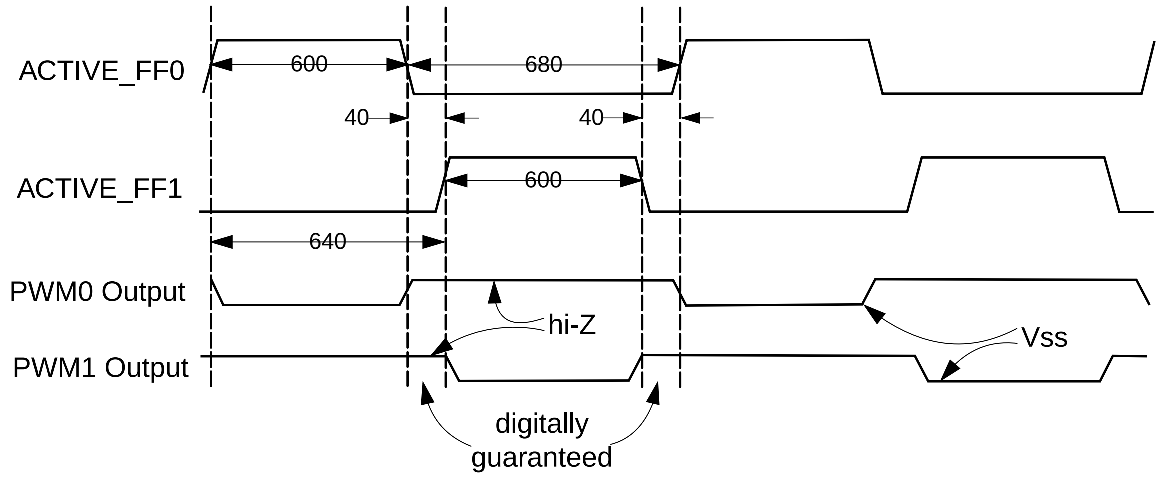
The actual values obtainable on the output are shown in Figure 28-2. Notice that one possible state is to turn off the output driver to provide a high-Z output. This is useful for external circuits that drive E.L. backlights and for direct drive of LEDs.

28.2.1 XTAL OSC Driving Mode

By default, every PWM channel outputs signals in XTAL OSC clock domain. It’s a 24 MHz XTAL OSC.

By setting up two channels in lock step and by setting their low and high states to opposite values, one can generate a differential signal pair that alternates between pulling to Vss and floating to high-Z. By creating an appropriate offset in the settings of the two channels with the same period and the same enables, one can generate differential drive pulses with digitally guaranteed non-overlapping intervals suitable for controlling high-voltage switches.

In Figure 28-3, a differential pair is established using Channel 0 and Channel 1. The period is set for 1280 divided clocks for both channels. All active phases are set for 600 divided clocks. There is a 40 divided clock guaranteed off-time between each active phase. Since this is based on a crystal oscillator, it is a very stable non-overlapping period. The total period is also a very stable crystal-oscillator-based time interval. In this example, the active phases are pulled to Vss (ground), while the inactive phases are allowed to float to a high-Z state.



HW_PWM_PERIOD0_PERIOD = 1280	HW_PWM_PERIOD1= 1280
HW_PWM_ACTIVE0_ACTIVE = 0	HW_PWM_ACTIVE1 = 640
HW_PWM_ACTIVE0_INACTIVE = 600	HW_PWM_INACTIVE1 = 1240
HW_PWM_PERIOD0_ACTIVE_STATE = 10	HW_PWM_ACTIVE1_STATE = 10
HW_PWM0BR_INACTIVE_STATE = 00	HW_PWM_INACTIVE1_STATE = 00

Figure 28-3. PWM Differential Output Pair Example

Figure 28-5 shows the generation of the PWM Channel 3 output. This channel controls the output pin when PWM control is selected in PINCTRL block and HW_PWM_CTRL_PWM3_ENABLE is set to 1. The output pin can be set to a 0, a 1, or left to float in the high-impedance state. These choices can be made independently for either the active or inactive phase of the output.

28.2.2 HSADC Driving Mode

The HSADC driving mode ouputs signals in HSADC (high-speed ADC) sample clock domain, other than traditional 24 MHz XTAL OSC domain.

Although the main target of HSADC block is to drive TOSHIBA TCD1304DG linear image scanner sensor, there is potential requirement for HSADC to drive other linear image scanner sensors as well. To improve flexibility, PWM is used to generate these driving signals; and to co-work with block HSADC, HSADC's sample clock is used for PWM output signals.

Typically, as which is showed in Figure 28-4, three output driving signals are needed for HSADC. And one extra trigger, which is also synchronous with high-speed ADC block, might be needed to start the conversion of ADC according to HSADC's specified trigger mode.

As a result, one clock mux is added for every PWM instance each to output signals synchronous with HSADC sample clock, and there is a hard-wired connection between every PWM instance and high-speed ADC block for HSADC's PWM trigger mode.

For every PWM instance, software can choose whether HSADC sample clock or 24 MHz XTAL OSC clock is used for PWM output by writing 1'b1 to bit **HSADC_CLK_SEL** of Register **PWM Channelx Period Register**(x = 0-7). Meanwhile, it's programmable whether this output signal is routed to HSADC internally by writing 1'b1 to bit **HSADC_OUT** of Register **PWM Channelx Period Register**(x = 0-7). When on HSADC driving mode, it is advisable to wait enough time for another write access to the same PWM register, such as HW_PWM_CTR, HW_PWM_ACTIVEx(x = 0-7) and HW_PWM_PERIODx(x = 0-7). Because handshake is applied, if several write access to the same register is consecutive to each other, it is not guaranteed what's the content of programmed register.

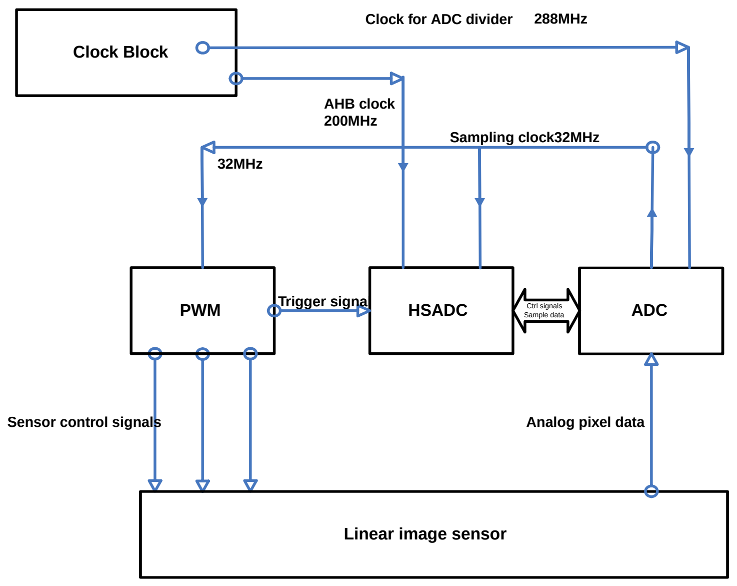


Figure 28-4. PWM HSADC Usecase

28.2.3 Multi-Chip Attachment Mode

The multi-chip attachment mode (MATT) allows a 24 MHz or 32 KHz crystal clock that is an input to the i.MX28 to be routed to the PWM output pins. In this case, the normal PWM programming parameters (for example, PERIOD, ACTIVE, and so on) are ignored. This mode allows for supplying and controlling the crystal clock for external application interfaces, as shown in Figure 28-5.

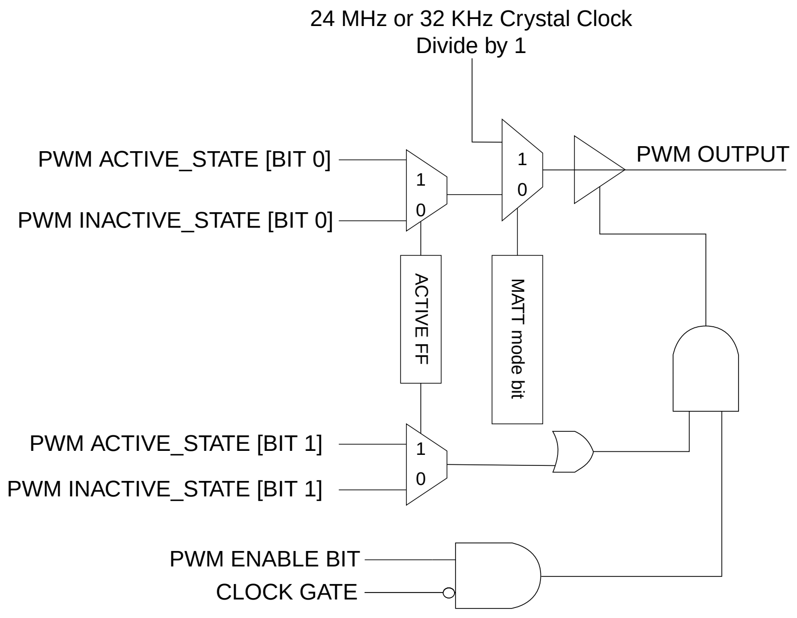


Figure 28-5. PWM Output Driver

28.2.4 Channel 2 Analog Enable Function

The output generation for channel 2 is slightly different than shown in Figure 28-5. In this channel, there is an additional enable that is controlled from the analog LRADC block. This signal is synchronized in the XTAL domain and ANDed with the PWM ENABLE bit. So, in this case, either enable source can disable the output. Also, this analog enable control signal can be disabled through the PWM2_ANA_CTRL_ENABLE bit in the HW_PWM_CTRL register. When disabled, Channel 2 behaves identically to the other channels.

28.2.5 Channel Output Cutoff Using Module Clock Gate

Whenever the clkgate is enabled (gated) the output from all PWM channels is hi-Z. If the clock gate is asserted while PWM is enabled and generating an output signal the output is immediately disabled. This will not affect the current state, programming or enables of the pwm channels themselves. When the clkgate is de-asserted, the PWM outputs will resume according to their programmed parameters and current states. Therefore glitches on the enabled channel outputs will likely occur when the clkgate state is changed. All 8 channels will function identically in this regard.

28.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

28.4 Programmable Registers

PWM Hardware Register Format Summary

HW_PWM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8006_4000	PWM Control and Status Register (HW_PWM_CTRL)	32	R/W	FFC0_0000h	28.4.1/1782
8006_4010	PWM Channel 0 Active Register (HW_PWM_ACTIVE0)	32	R/W	0000_0000h	28.4.2/1784
8006_4020	PWM Channel 0 Period Register (HW_PWM_PERIOD0)	32	R/W	0000_0000h	28.4.3/1785
8006_4030	PWM Channel 1 Active Register (HW_PWM_ACTIVE1)	32	R/W	0000_0000h	28.4.4/1787

HW_PWM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8006_4040	PWM Channel 1 Period Register (HW_PWM_PERIOD1)	32	R/W	0000_0000h	28.4.5/1788
8006_4050	PWM Channel 2 Active Register (HW_PWM_ACTIVE2)	32	R/W	0000_0000h	28.4.6/1789
8006_4060	PWM Channel 2 Period Register (HW_PWM_PERIOD2)	32	R/W	0000_0000h	28.4.7/1790
8006_4070	PWM Channel 3 Active Register (HW_PWM_ACTIVE3)	32	R/W	0000_0000h	28.4.8/1792
8006_4080	PWM Channel 3 Period Register (HW_PWM_PERIOD3)	32	R/W	0000_0000h	28.4.9/1792
8006_4090	PWM Channel 4 Active Register (HW_PWM_ACTIVE4)	32	R/W	0000_0000h	28.4.10/1794
8006_40A0	PWM Channel 4 Period Register (HW_PWM_PERIOD4)	32	R/W	0000_0000h	28.4.11/1795
8006_40B0	PWM Channel 5 Active Register (HW_PWM_ACTIVE5)	32	R/W	0000_0000h	28.4.12/1797
8006_40C0	PWM Channel 5 Period Register (HW_PWM_PERIOD5)	32	R/W	0000_0000h	28.4.13/1797
8006_40D0	PWM Channel 6 Active Register (HW_PWM_ACTIVE6)	32	R/W	0000_0000h	28.4.14/1799
8006_40E0	PWM Channel 6 Period Register (HW_PWM_PERIOD6)	32	R/W	0000_0000h	28.4.15/1800
8006_40F0	PWM Channel 7 Active Register (HW_PWM_ACTIVE7)	32	R/W	0000_0000h	28.4.16/1802
8006_4100	PWM Channel 7 Period Register (HW_PWM_PERIOD7)	32	R/W	0000_0000h	28.4.17/1802
8006_4110	PWM Version Register (HW_PWM_VERSION)	32	R	0106_0000h	28.4.18/1804

28.4.1 PWM Control and Status Register (HW_PWM_CTRL)

The PWM Control and Status Register specifies the reset state, availability, and the enables for the eight PWM units.

HW_PWM_CTRL: 0x000

HW_PWM_CTRL_SET: 0x004

HW_PWM_CTRL_CLR: 0x008

HW_PWM_CTRL_TOG: 0x00C

EXAMPLE

```
HW_PWM_CTRL_WR(0x000000ff);    // Enable all channels
```

Address: HW_PWM_CTRL – 8006_4000h base + 0h offset = 8006_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	PWM7_ PRESENT	PWM6_ PRESENT	PWM5_ PRESENT	PWM4_ PRESENT	PWM3_ PRESENT	PWM2_ PRESENT	PWM1_ PRESENT	PWM0_ PRESENT	RSRVD1[21:16]					
W																
Reset	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:10]						OUTPUT_CUTOFF_ EN	RSRVD2	PWM7_ENABLE	PWM6_ENABLE	PWM5_ENABLE	PWM4_ENABLE	PWM3_ENABLE	PWM2_ENABLE	PWM1_ENABLE	PWM0_ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_CTRL field descriptions

Field	Description
31 SFTRST	This bit must be cleared to 0 for normal operation. When set to 1, it forces a block-wide reset.
30 CLKGATE	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block.
29 PWM7_ PRESENT	0 = PWM7 is not present in this product.
28 PWM6_ PRESENT	0 = PWM6 is not present in this product.
27 PWM5_ PRESENT	0 = PWM5 is not present in this product.
26 PWM4_ PRESENT	0 = PWM4 is not present in this product.
25 PWM3_ PRESENT	0 = PWM3 is not present in this product.
24 PWM2_ PRESENT	0 = PWM2 is not present in this product.
23 PWM1_ PRESENT	0 = PWM1 is not present in this product.

HW_PWM_CTRL field descriptions (continued)

Field	Description
22 PWM0_ PRESENT	0 = PWM0 is not present in this product.
21–10 RSRVD1	Reserved.
9 OUTPUT_ CUTOFF_EN	When asserted this bit enables the block to automatically Hi-Z state the outputs whenever the clkgate is asserted. The default is disabled.
8 RSRVD2	Reserved.
7 PWM7_ENABLE	Enables PWM channel 7 to begin cycling when set to 1. To enable PWM7 onto the output pin, the pin control registers must programmed accordingly.
6 PWM6_ENABLE	Enables PWM channel 6 to begin cycling when set to 1. To enable PWM6 onto the output pin, the pin control registers must programmed accordingly.
5 PWM5_ENABLE	Enables PWM channel 5 to begin cycling when set to 1. To enable PWM5 onto the output pin, the pin control registers must programmed accordingly.
4 PWM4_ENABLE	Enables PWM channel 4 to begin cycling when set to 1. To enable PWM4 onto the output pin, the pin control registers must programmed accordingly.
3 PWM3_ENABLE	Enables PWM channel 3 to begin cycling when set to 1. To enable PWM3 onto the output pin, the pin control registers must programmed accordingly.
2 PWM2_ENABLE	Enables PWM channel 2 to begin cycling when set to 1. To enable PWM2 onto the output pin, the pin control registers must programmed accordingly.
1 PWM1_ENABLE	Enables PWM channel 1 to begin cycling when set to 1. To enable PWM1 onto the output pin, the pin control registers must programmed accordingly.
0 PWM0_ENABLE	Enables PWM channel 0 to begin cycling when set to 1. To enable PWM0 onto the output pin, the pin control registers must programmed accordingly.

28.4.2 PWM Channel 0 Active Register (HW_PWM_ACTIVE0)

The PWM Channel 0 Active Register specifies the active time and inactive time for Channel 0.

HW_PWM_ACTIVE0: 0x010

HW_PWM_ACTIVE0_SET: 0x014

HW_PWM_ACTIVE0_CLR: 0x018

HW_PWM_ACTIVE0_TOG: 0x01C

EXAMPLE

```
HW_PWM_ACTIVEn_WR(0, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE0 – 8006_4000h base + 10h offset = 8006_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE0 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.3 PWM Channel 0 Period Register (HW_PWM_PERIOD0)

The PWM Channel 0 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD0: 0x020

HW_PWM_PERIOD0_SET: 0x024

HW_PWM_PERIOD0_CLR: 0x028

HW_PWM_PERIOD0_TOG: 0x02C

EXAMPLE

```
HW_PWM_PERIODn_WR(0, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD0 – 8006_4000h base + 20h offset = 8006_4020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD0 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC. Setting this bit to 1'b1, output of PWM channel 0 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 0 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM0 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64. 0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to hi-Z. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).

HW_PWM_PERIOD0 field descriptions (continued)

Field	Description
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.4 PWM Channel 1 Active Register (HW_PWM_ACTIVE1)

The PWM Channel 1 Active Register specifies the active time and inactive time for Channel 1.

HW_PWM_ACTIVE1: 0x030

HW_PWM_ACTIVE1_SET: 0x034

HW_PWM_ACTIVE1_CLR: 0x038

HW_PWM_ACTIVE1_TOG: 0x03C

EXAMPLE

```
HW_PWM_ACTIVEn_WR(1, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE1 – 8006_4000h base + 30h offset = 8006_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE1 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.5 PWM Channel 1 Period Register (HW_PWM_PERIOD1)

The PWM Channel 1 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD1: 0x040

HW_PWM_PERIOD1_SET: 0x044

HW_PWM_PERIOD1_CLR: 0x048

HW_PWM_PERIOD1_TOG: 0x04C

EXAMPLE

```
HW_PWM_PERIODn_WR(1, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD1 – 8006_4000h base + 40h offset = 8006_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_ CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_ STATE		ACTIVE_ STATE	
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD1 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC.Setting this bit to 1'b1, output of PWM channel 1 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 1 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.

HW_PWM_PERIOD1 field descriptions (continued)

Field	Description
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM1 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64. 0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.6 PWM Channel 2 Active Register (HW_PWM_ACTIVE2)

The PWM Channel 2 Active Register specifies the active time and inactive time for Channel 2.

HW_PWM_ACTIVE2: 0x050

HW_PWM_ACTIVE2_SET: 0x054

HW_PWM_ACTIVE2_CLR: 0x058

HW_PWM_ACTIVE2_TOG: 0x05C

EXAMPLE

```
HW_PWM_ACTIVEn_WR(2, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE2 – 8006_4000h base + 50h offset = 8006_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE2 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.7 PWM Channel 2 Period Register (HW_PWM_PERIOD2)

The PWM Channel 2 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD2: 0x060

HW_PWM_PERIOD2_SET: 0x064

HW_PWM_PERIOD2_CLR: 0x068

HW_PWM_PERIOD2_TOG: 0x06C

EXAMPLE

```
HW_PWM_PERIODn_WR(2, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD2 – 8006_4000h base + 60h offset = 8006_4060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_ CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_ STATE		ACTIVE_ STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERIOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD2 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC.Setting this bit to 1'b1, output of PWM channel 2 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 2 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM2 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64. 0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.8 PWM Channel 3 Active Register (HW_PWM_ACTIVE3)

The PWM Channel 3 Active Register specifies the active time and inactive time for Channel 3.

HW_PWM_ACTIVE3: 0x070

HW_PWM_ACTIVE3_SET: 0x074

HW_PWM_ACTIVE3_CLR: 0x078

HW_PWM_ACTIVE3_TOG: 0x07C

EXAMPLE

```
HW_PWM_ACTIVEn_WR(3, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE3 – 8006_4000h base + 70h offset = 8006_4070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE3 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.9 PWM Channel 3 Period Register (HW_PWM_PERIOD3)

The PWM Channel 3 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD3: 0x080

HW_PWM_PERIOD3_SET: 0x084

HW_PWM_PERIOD3_CLR: 0x088

HW_PWM_PERIOD3_TOG: 0x08C

EXAMPLE

```
HW_PWM_PERIODn_WR(3, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD3 – 8006_4000h base + 80h offset = 8006_4080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD3 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC. Setting this bit to 1'b1, output of PWM channel 3 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 3 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM3 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal clock frequency (24.0 MHz) that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64.

HW_PWM_PERIOD3 field descriptions (continued)

Field	Description
	0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.10 PWM Channel 4 Active Register (HW_PWM_ACTIVE4)

The PWM Channel 4 Active Register specifies the active time and inactive time for Channel 4.

HW_PWM_ACTIVE4: 0x090

HW_PWM_ACTIVE4_SET: 0x094

HW_PWM_ACTIVE4_CLR: 0x098

HW_PWM_ACTIVE4_TOG: 0x09C

EXAMPLE

```
HW_PWM_ACTIVEn_WR(4, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE4 – 8006_4000h base + 90h offset = 8006_4090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE4 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.11 PWM Channel 4 Period Register (HW_PWM_PERIOD4)

The PWM Channel 4 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD4: 0x0A0

HW_PWM_PERIOD4_SET: 0x0A4

HW_PWM_PERIOD4_CLR: 0x0A8

HW_PWM_PERIOD4_TOG: 0x0AC

EXAMPLE

```
HW_PWM_PERIODn_WR(4, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD4 – 8006_4000h base + A0h offset = 8006_40A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD4 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC. Setting this bit to 1'b1, output of PWM channel 4 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 4 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM4 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64. 0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.12 PWM Channel 5 Active Register (HW_PWM_ACTIVE5)

The PWM Channel 5 Active Register specifies the active time and inactive time for Channel 5.

HW_PWM_ACTIVE5: 0x0b0

HW_PWM_ACTIVE5_SET: 0x0b4

HW_PWM_ACTIVE5_CLR: 0x0b8

HW_PWM_ACTIVE5_TOG: 0x0bC

EXAMPLE

```
HW_PWM_ACTIVEn_WR(5, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE5 – 8006_4000h base + B0h offset = 8006_40B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE5 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.13 PWM Channel 5 Period Register (HW_PWM_PERIOD5)

The PWM Channel 5 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD5: 0x0c0

HW_PWM_PERIOD5_SET: 0x0c4

HW_PWM_PERIOD5_CLR: 0x0c8

HW_PWM_PERIOD5_TOG: 0x0cC

EXAMPLE

```
HW_PWM_PERIODn_WR(5, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD5 – 8006_4000h base + C0h offset = 8006_40C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_ CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_ STATE		ACTIVE_ STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD5 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC.Setting this bit to 1'b1, output of PWM channel 5 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 5 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM5 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64.

HW_PWM_PERIOD5 field descriptions (continued)

Field	Description
	0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.14 PWM Channel 6 Active Register (HW_PWM_ACTIVE6)

The PWM Channel 6 Active Register specifies the active time and inactive time for Channel 6.

HW_PWM_ACTIVE6: 0x0d0

HW_PWM_ACTIVE6_SET: 0x0d4

HW_PWM_ACTIVE6_CLR: 0x0d8

HW_PWM_ACTIVE6_TOG: 0x0dC

EXAMPLE

```
HW_PWM_ACTIVEn_WR(6, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE6 – 8006_4000h base + D0h offset = 8006_40D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE6 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.15 PWM Channel 6 Period Register (HW_PWM_PERIOD6)

The PWM Channel 6 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD6: 0x0e0

HW_PWM_PERIOD6_SET: 0x0e4

HW_PWM_PERIOD6_CLR: 0x0e8

HW_PWM_PERIOD6_TOG: 0x0eC

EXAMPLE

```
HW_PWM_PERIODn_WR(6, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD6 – 8006_4000h base + E0h offset = 8006_40E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE	
W	▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD6 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC. Setting this bit to 1'b1, output of PWM channel 6 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 6 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM6 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64. 0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.16 PWM Channel 7 Active Register (HW_PWM_ACTIVE7)

The PWM Channel 7 Active Register specifies the active time and inactive time for Channel 7.

HW_PWM_ACTIVE7: 0x0f0

HW_PWM_ACTIVE7_SET: 0x0f4

HW_PWM_ACTIVE7_CLR: 0x0f8

HW_PWM_ACTIVE7_TOG: 0x0fC

EXAMPLE

```
HW_PWM_ACTIVEn_WR(7, 0x000000ff);   // Set active and inactive counts
```

Address: HW_PWM_ACTIVE7 – 8006_4000h base + F0h offset = 8006_40F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INACTIVE																ACTIVE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_ACTIVE7 field descriptions

Field	Description
31–16 INACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output from the active state to the inactive state. The internal count of the channel is compared for greater than this value to change to the inactive state.
15–0 ACTIVE	Number of divided clock cycles to count from the beginning of the period before changing the output to the active state. The internal count of the channel is compared for greater than this value to change to the active state.

28.4.17 PWM Channel 7 Period Register (HW_PWM_PERIOD7)

The PWM Channel 7 Period Register specifies the multi-chip attachment mode, clock divider value, active high, low values and period.

HW_PWM_PERIOD7: 0x100

HW_PWM_PERIOD7_SET: 0x104

HW_PWM_PERIOD7_CLR: 0x108

HW_PWM_PERIOD7_TOG: 0x10C

EXAMPLE

```
HW_PWM_PERIODn_WR(7, 0x00000b1f);   // Set up period and active/inactive output states
```

Address: HW_PWM_PERIOD7 – 8006_4000h base + 100h offset = 8006_4100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2					HSADC_OUT	HSADC_CLK_SEL	MATT_SEL	MATT	CDIV			INACTIVE_STATE		ACTIVE_STATE	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERIOD															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_PERIOD7 field descriptions

Field	Description
31–27 RSRVD2	Reserved.
26 HSADC_OUT	PWM output to HSADC. Setting this bit to 1'b1, output of PWM channel 7 is routed to HSADC internally.Only one PWM channle's HSADC_OUT should be set to 1'b1. If multiple PWM channels are set as HSADC_OUT, only the channel with the lowest channel number will be routed to HSADC
25 HSADC_CLK_ SEL	HSADC clock select for PWM output. Setting this bit to 1'b1, PWM channel 7 counts on HSADC input clock to drive PWM output.
24 MATT_SEL	Multichip Attachment Mode clock select. When the MATT bit is asserted this bit selects which clock to output. 0: 32 kHz, 1: 24 MHz.
23 MATT	Multichip Attachment Mode. This bit overrides the normal signal generation parameters and enables either the 24 MHz or 32 kHz crystal clock on the PWM7 output pin for inter-chip signaling. When this bit is set to 0x1, bit HSADC_CLK_SET in Period Register must be set to 0x0. Otherwise, something unexpected may happen.
22–20 CDIV	Clock divider ratio to apply to the crystal/HSADC clock frequency that times the PWM output signal. 0x0 **DIV_1** — Divide by 1. 0x1 **DIV_2** — Divide by 2. 0x2 **DIV_4** — Divide by 4. 0x3 **DIV_8** — Divide by 8. 0x4 **DIV_16** — Divide by 16. 0x5 **DIV_64** — Divide by 64.

HW_PWM_PERIOD7 field descriptions (continued)

Field	Description
	0x6 **DIV_256** — Divide by 256. 0x7 **DIV_1024** — Divide by 1024.
19–18 INACTIVE_ STATE	The logical inactive state that is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Inactive state sets PWM output to high-impedance. 0x2 **0** — Inactive state sets PWM output to 0 (low). 0x3 **1** — Inactive state sets PWM output to 1 (high).
17–16 ACTIVE_STATE	The logical active state is mapped to the PWM output signal. Note that the undefined state of 0x1 is mapped to high-impedance. 0x0 **HI_Z** — Active state sets PWM output to high-impedance. 0x2 **0** — Active state sets PWM output to 0 (low). 0x3 **1** — Active state sets PWM output to 1 (high).
15–0 PERIOD	Number of divided clock cycles in the entire period of the PWM waveform, minus 1. For example, to obtain 6 clock cycles in the actual period, set this field to 5.

28.4.18 PWM Version Register (HW_PWM_VERSION)

This register indicates the version of the block for debug purposes.

EXAMPLE

```
if (HW_PWM_VERSION.B.MAJOR != 1) Error();
```

Address: HW_PWM_VERSION – 8006_4000h base + 110h offset = 8006_4110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PWM_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 29
Programmable 3-Port Ethernet Switch with QOS (SWITCH)

29.1 3-Port Ethernet Switch Features

- Integrated Ethernet switch engine compatible with ENET-MAC core
- Three port switch with a fourth DMA bypass port
- Can be configured to operate as a 3-Port Switch (Switch Mode) or as two independent ports (Passthrough Mode)
- Filters and forward traffic at wire-speed on all ports
- Per-queue tail-drop congestion management
- Implements hardware switching look-up mechanism providing a learning capacity of up to 2 K MAC addresses
- Supports configurable VLAN switching when MAC address lookup should be omitted
- Classification and Priority assignment based on Port Number, MAC Address, Ipv4 DiffServ Code Point Field, Ipv6 Class of Service and VLAN Priority (IEEE802.1q)
- Efficient output Queue frame buffering with shared Frame buffer of 24 Kbyte
- Each port implements four priority queues with configurable weighted round-robin selection
- Support Ethernet Multicast, Broadcast with flooding control to avoid unnecessary duplication of frames
- Programmable Multicast destination port mask to restrict frame duplication for individual multicast addresses
- Multicast and Broadcast resolution with VLAN domain filtering providing a strict separation of up to 32 VLANs
- IP Snooping with programmable protocol and port number registers

- Programmable Ingress and Egress VLAN tag addition, removal and manipulation supporting single and double-tagged VLAN frames
- Event and status signals which can be used to monitor port activity, severe error conditions or any user specific event
- Programmable firmware operation with Static or Dynamic (Learning, Aging) switching tables
- Switch firmware source available to provide customer specific software development capability
- Support for 1588 precise time stamping applications
- Supports Aggregation and Redundant Backplane applications

29.2 Block Diagram

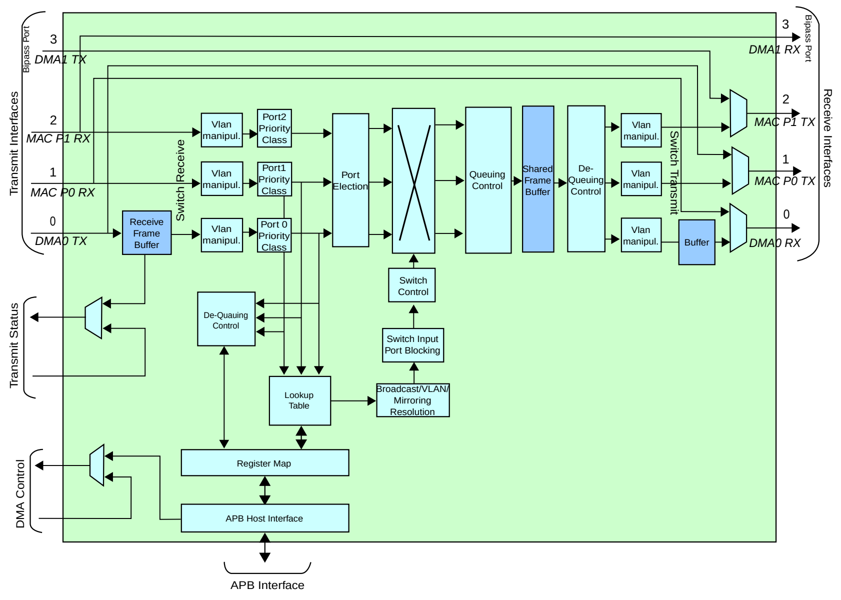


Figure 29-1. Block Diagram

Note

- The left side is termed "Transmit Interfaces" as the interfaces are driven by the DMA transmit interfaces in pass-through mode. If the switch is enabled, these interfaces connect to the switch internal receive interfaces. The right side is named "Receive Interfaces" as the interfaces represent (are driven from) the respective MAC receive interfaces in pass-through mode connected to the DMA RX. If the switch is enabled these are driven by the switch internal transmit interfaces. See 4 page 26 for a description of the operational modes.

- All descriptions related to switch functions refer to the switch internal receive/transmit interfaces.
- Receive Frame buffer (DMA0 TX interface). Can hold at least one complete frame transferred from DMA0 to the switch. This is a local 64-bit wide FIFO to absorb a burst from the DMA, provide the necessary transaction handshake to the DMA, and forward the frame then to the switch logic.
- Decoupling buffer only (DMA0 RX interface) to absorb switch latency (for example, 16/32 words) resulting from a rdy deassertion from DMA0.
- The APB Host Interface module implements an indirect addressing scheme to the internal registers to allow arbitrary length access cycles (such as address table read/write and configuration register read/write). However, the DMA control registers are directly mapped and accessed through APB.

29.3 Switch Configuration

29.3.1 Overview

The Switch Core is designed to be seamlessly connected to the MorethanIP ENET-MAC Core and DMA controllers. For control and configuration, the switch implements an APB Register interface and multiple maskable interrupts.

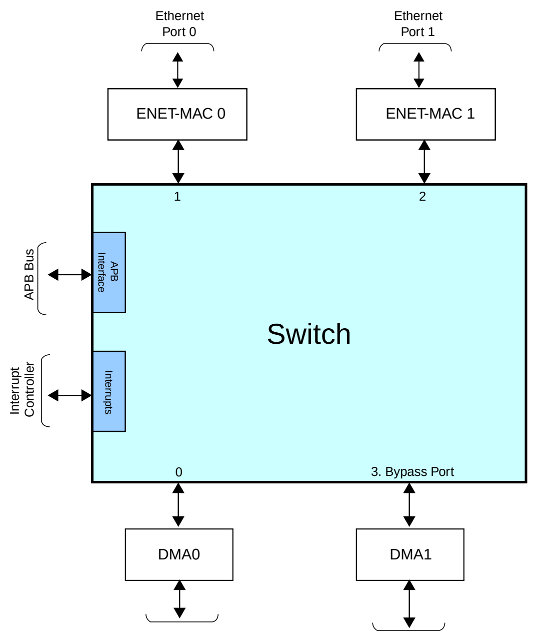


Figure 29-2. Switch Interface

The switch port assignment is listed in Table 29-1 and should be strictly followed when programming and integrating the Switch Core.

Table 29-1. Port Assignment

Switch Port	Assignment
0	DMA 0
1	ENET-MAC 0
2	ENET-MAC 1
3 (bypass port)	DMA 1

The Switch, controlled with the configuration pin sx_ena, can be programmed to operate is two modes:

- Passthrough mode: The switch logic is disabled and bypassed.
- Switch mode: The switch logic is enabled

29.3.2 Passthrough Mode

When configuration signal sx_ena is set to '0', the switch logic is bypassed and can be totally powered down and disabled with, for example, the Switch clocks clk and pclk stopped and the Switch reset signals reset_clk and reset_pclk set to '1'.

The Switch APB interface and interrupt signals are disabled and should not be used. To control the Frame transfer from DMA0 and DMA1, the ENET-MAC 0 and the ENET-MAC 1 APB interfaces and interrupt signals should be used.

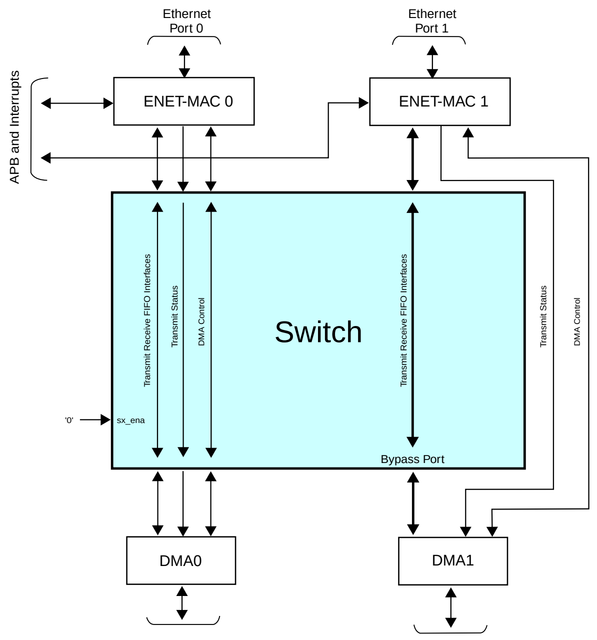


Figure 29-3. Passthrough Mode Configuration Overview

29.3.3 Switch Mode

When the Switch is programmed to operate in Switch Mode (sx_ena set to '1'), the Bypass Mode (Port 1) interface is disabled and should not be used.

Frame transfers to and from the Line are performed on Port 0 only (DMA 0). The Transmit status signals are generated from the Switch Port 0 Receive Buffer and the DMA control signals from the Switch Register Space. The ENET-MAC 0 and ENET-MAC 1 Transmit status and DMA control signals are not used.

The ENET-MAC 0 and ENET-MAC 1 APB interfaces and interrupts are enabled and can be used to monitor the line activity and gather the line statistic information.

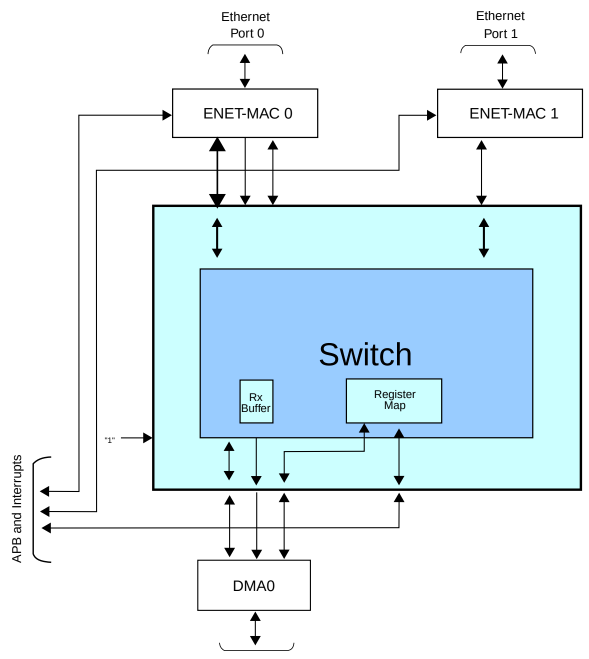


Figure 29-4. Switch Mode Configuration Overview

29.3.4 Port 0 Input Buffer

A dedicated input buffer of at least 2 Kbyte storage is implemented at the port0 (DMA0) input interface, when the switch is operating in switch mode. In bypass mode, this buffer is bypassed.

The input buffer is normally operated in store&forward mode of operation, absorbing a DMA burst and forwarding the frame to the switch (internally) when the complete frame has been stored in the input buffer.

However, to allow for jumbo-frame support, the input buffer can be operated in cut-through mode (see register MODE_CONFIG). In cut-through mode, a frame is forwarded to the switch internal input, when the buffer has reached a fill level as 1536 bytes (192 words).

If a frame of less than this threshold is written by DMA0, it is forwarded as in store&forward with all options (ff_tx_xxx). If a frame larger than this threshold is written, then the input buffer will start forwarding the frame to the switch, when the threshold is reached. It is then expected that DMA0 guarantees writing data fast enough to avoid a buffer underflow.

When the input buffer is operating in cut-through mode, the additional transmit options (ff_tx_crc_fwd0, ff_tx_ipchk_ins0, ff_tx_protchk_ins0, ff_tx_ts_frm0) can be used only for frames not exceeding the cut-through threshold. Frames exceeding the threshold cannot use these functions and they must be set to 0 (and they will operate as defined for 0 accordingly).

29.3.5 Port 0 Input Backpressure/Congestion Indication

When frames are transferred from DMA0 to the switch port0 transmit interface, the interface signal ff_tx_rdy0 indicates if data can be transferred (written) to the port 0 input. When ff_tx_rdy0 deasserts (0), it indicates a stop request to the DMA0. The DMA0 might continue to write a few more words (typically up to 4) and is then expected to stop writing and wait for assertion (1) of the signal again. The signal is controlled by the port 0 local input buffer and deasserts when the buffer reaches an almost full threshold.

In addition, a special backpressure mechanism for Port0 is implemented to pause DMA0 transfers when the output queues' shared memory (see Overview) becomes full to a programmable threshold. Respecting the output queues' shared memory fill level can avoid that the switch due to memory congestion discards frames written by the DMA0. The threshold is configured through QMGR_MINCELLSP0 configuration register. The threshold is used as follows:

If the shared memory has less than QMGR_MINCELLSP0 number of free cells available, the switch stops serving port 0. That is, the switch will not start to read a frame from the port0 input buffer. The port 0 input buffer will continue to accept data from DMA0 until it becomes full and as a result deasserts ff_tx_rdy0.

Note that the backpressure only considers the total amount of memory available, not a specific queue. Hence, it may still happen that an outgoing frame from DMA0 is discarded by the switch, if the output queue for the frame is congested while the total amount of memory would have free space available.

The backpressure threshold (QMGR_MINCELLSP0) should be set higher than the memory full threshold (QMGR_MINCELLS) to stop the DMA0 before a memory full situation, leading to discard of frames, can occur.

29.4 Switch Functional Description

29.4.1 Overview

The Switch implements the following main functions:

- Input/Output VLAN Processing
- IP Snooping
- Input Frame Parsing and Priority Extraction
- Input Port Selection

- Output Port(s) Resolution
- Frame Queuing
- Output Queue Scheduling

The standard Firmware together with the switching Hardware and, optionally, MAC Cores from MorethanIP already provides a complete Ethernet switching solution. As the Switch Firmware is developed in a modular way, MorethanIP can implement, upon request, specific functions like for example, Spanning Tree)

The Hardware and Software designs of the switch are also highly tightly optimized together to provide high performances and to be scalable to higher throughput switching solutions.

29.4.2.1 Overview

The VLAN input processing function is used on each switch input port to inspect and manipulate the VLAN tag of frames entering the switch. It performs the following functions:

- Input Frame Parsing
- VLAN tag insertion or manipulation

Based on the information of the Input Processing function, the frame can be switched to the corresponding output port or will be discarded.

29.4.2.2 Terms and Definitions

VLAN Information: The 16-bit field following the VLAN type field within a frame.

VLAN-ID: The lower 12 bits of the VLAN information field.

VLAN-Priority: The upper 3 bits of the VLAN information field. Used to prioritize incoming frames. A value of 0 represents lowest priority, a value of 7 represents highest priority.

29.4.2.3 Configuration Information

The switch management provides the following information to configure and control the operation of the function:

- SYSTEM_TAGINFOn: 16 bit value. The VLAN information field (VLAN-ID and priority) used for tag insertion operations.
- Mode of operation: There are different modes of operation, which define how incoming frames must be processed for a port. The function can be enabled and configured individually per port: See registers VLAN_IN_MODE_ENA and VLAN_IN_MODE.

Note

If the vlan input processing function is not enabled (VLAN_IN_MODE_ENA bit of the port=0), the mode setting has no effect.

29.4.2.4 Modes of Operation

29.4.2.4.1 Frame Processing

The VLAN input processing function modifies the frames before they enter the switching engine. This means, if a VLAN tag is inserted, the switch will only act on the inserted VLAN tag (for example, priority) and any original tag that was found in the frame before the modification, if any, has no effect within the switch.

In addition, if VLAN verification is enabled for a port (see register VLAN_VERIFY), the VLAN-id used for insertion (SYSTEM_TAGINFOn) must also be configured in the global VLAN resolution table (see register VLAN_RES_TABLE_0 to VLAN_RES_TABLE_31), to ensure the switch accepts frames, which contain the inserted tag.

When, in any of the modes, a tag is inserted, it is always inserted as first tag (outer) and its information field is set as programmed in the SYSTEM_TAGINFO0..2 register for the port n where the frame is received.

29.4.2.4.2 Mode 1 -- Single Tagging with Passthrough

Insert Tag only if untagged frame, leave frame unmodified if tagged.

29.4.2.4.3 Mode 2 -- Single Tagging with Replace

If untagged, add the tag, if single tagged, overwrite the tag.

29.4.2.4.4 Mode 3 -- Double Tagging with Passthrough

Insert a tag on untagged and tagged. This results in a single tagged frame when an untagged is received, and a double tagged frame, when a single tagged frame is received. When a double tagged frame is received the frame is left unmodified.

29.4.2.4.5 Mode 4 -- Double Tagging with Replace

Insert Tag on untagged and single tagged. If double tagged, overwrite outer tag.

29.4.3 IP Snooping

Programmable snooping for up to eight programmable IP protocols. If the protocol field of an IPv4 or IPv6 frame matches one of the programmed values and snooping is enabled for that entry, the frame can be processed as follows:

- Forward to designated management port only
- Copy to management port and normal forward/flood
- Discard on match

The management port is identified by the port number set in register MGMT_CONFIG.

The function is configured using registers IPSNOOP1..8 of the register map.

The snooping function can be enabled/disabled individually for each of the entries. If no protocol matches, or a match occurs but snooping is disabled or the frame is coming from the management port itself, the frame is processed normally.

Note

Snooping respects any optional VLAN tags (that is, extracts next after last VLAN tag).

29.4.4 TCP/UDP Port Number Snooping

Programmable snooping for up to eight programmable TCP or UDP port numbers. If the source or destination port number field (configurable) within and TCP/IP or UDP/IP frame (IPv4 and IPv6) is matching the compare value and snooping is enabled for that entry, the frame can be processed as follows:

- Forward to designated management port only
- Copy to management port and normal forward/flood
- Discard on match

The management port is identified by the port number set in register MGMT_CONFIG.

The function is configured using registers PORTSNOOP1..8 of the register map.

The snooping function can be enabled/disabled individually for each of the entries. If no entry matches, or a match occurs but snooping is disabled or the frame is coming from the management port itself, the frame is processed normally.

Note

Port Number Snooping is possible only if the IP header ends up to 10 words (40 bytes) after the MAC header. If the IP header is ending later (e.g. IPv6+VLAN or IPv4+>20byte Options) the port numbers cannot be parsed any more and the port number snooping will be ignored (protocol based snooping is not affected by this limit).

Note

For IPv6 frames the port number can only be compared if the UDP or TCP header is the very next header to the IPv6 header (i.e. it does not detect such headers if any extension headers are present in an IPv6 frame before the TCP or UDP header).

29.4.5.1 Overview

The VLAN output processing function is used on a switch output port to manipulate the VLAN tag of the outgoing frames that leave the switch. Frames are processed based on the output Processing mode, and the number of Tags the frame contains.

29.4.5.2 Configuration Information

The switch management provides the information on operating mode to configure and control the operation of the function using the register VLAN_OUT_MODE of a port. There are three different modes of operation, which define how the outgoing frames should be processed.

29.4.5.3 Modes of Operation

29.4.5.3.1 Overview

The VLAN output processing function is configured to operate in one of the following modes, which define the way outgoing frames should be treated.

29.4.5.3.2 Mode 0: disabled

No frame manipulation occurs.

29.4.5.3.3 Mode 1: Strip Mode

In Strip mode, all the Tags (Single or double) are removed from incoming frame

29.4.5.3.4 Mode 2: Tag Thru Mode

In Tag Thru mode, the inner Tag is passed thru while the outer Tag is removed for a double Tagged frame. The following rules apply:

- When a single Tagged frame is received, strip the tag from the frame.
- When a double Tagged frame is received, strip the outer tag from the frame.

29.4.5.3.5 Mode 3: Transparent Mode

In Transparent mode, single tagged frame is kept unchanged. The following rules apply:

- When a single Tagged frame is received, frame is kept unchanged.
- When a double Tagged frame is received, strip the outer tag from the frame.

29.4.6.1 Overview

When a Frame is received on an input port, several information are extracted from the frame as the Ethernet MAC Address, VLAN tag and IP Headers to determine the Frame Type and perform the relevant Classification actions.

In addition, the MAC Address table can provide a priority indication for the destination MAC address if the switch management has programmed the address table accordingly (static entry).

The Frame is classified into four priority levels (priority0=lowest, priority3=highest) and is eventually queued in the corresponding priority queue at the output port.

29.4.6.2 VLAN Priority Look-Up

On each port, an 8 entry programmable priority table is implemented. The registers VLAN_PRIORITY0..2 contain the priority mapping for port n.

The switch uses 3-Bits from the VLAN tag information (3-bit VLAN priority) to extract the corresponding bits from the Table, which indicates which priority the Frame should be finally classified.

The index in the mapping table is the 3 bits of the first octet of the VLAN Tag Data with bit 5 (prio0) being the LSB and Bit 7 (prio(2)) being the MSB.

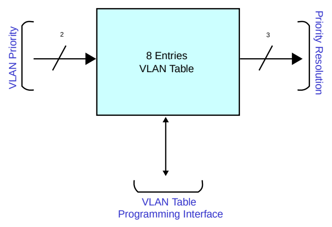


Figure 29-5. VLAN Table Overview

29.4.6.3 Ipv4 and Ipv6 Priority Look Up

29.4.6.3.1 Overview

The switch can classify both Ipv4 and Ipv6 frames: A 64-Entry Table is implemented per port to classify the IPv4 Frames and a 256-Entry Table is implemented per port to classify IPv6 Frames (The IP COS Tables) (see registers IP_PRIORITY).

On the IPv4 COS table entry, the Frame's 6-Bit DiffServ field is provided and the Table returns the 3-bit priority information.

On the Ipv6 COS table entry, the 8-Bit Class of Service field is provided and the Table returns the 3-bit priority information.

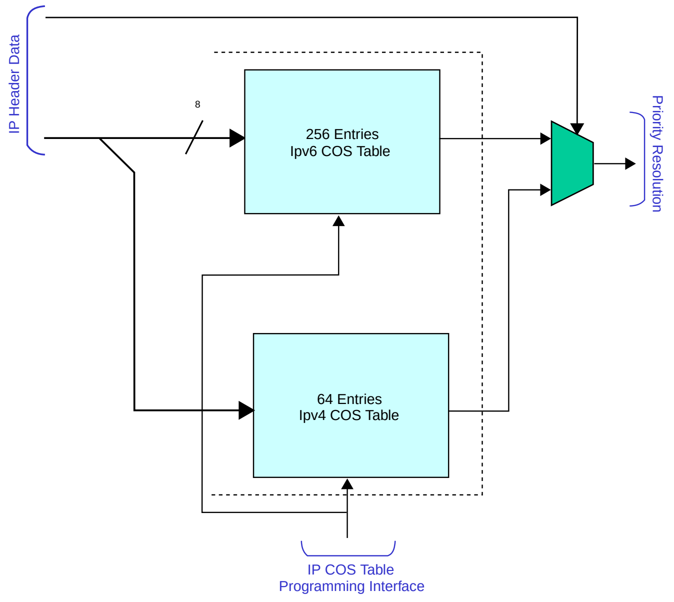


Figure 29-6. IP COS Tables Overview

29.4.6.3.2 Classification Table Programming Model

To program the mapping tables a indirect addressing scheme is implemented using a single register IP_PRIORITY within the register interface.

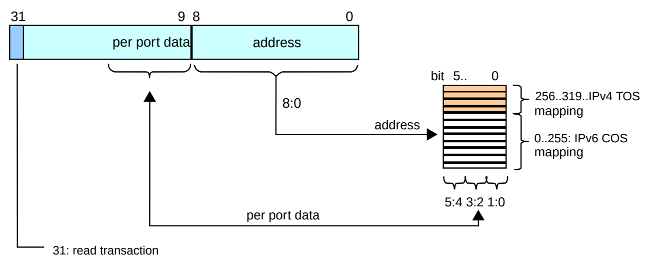


Figure 29-7. IP_PRIORITY Mapping Table Programming Model

The table is implemented in a single 320 deep table which is used for both IPv4 6-bit TOS and IPv6 8-bit COS mappings.

- The first 256 entries represent the IPv6 COS field mapping. The received COS field of a frame is used to address a row from 0..255. The value stored is read and used as priority for the frame.
- The last 64 entries represent the IPv4 DiffServ field mapping. The received DiffServ (upper 6-bits of TOS) value of a frame is used to address a row from 256..319.
- Each entry of the table provides the priority mapping for port0 in bits 1:0 (00=queue0, 01=queue1, 10=queue2, 11=queue3), for port 1 in bits 3:2 and port 2 in bits 5:4.

To write a table row into the table the address is provided in bits 8:0 and the data in bits 13:9, where bit 9 represents bit0 of the data and bit13 represents bit5 of the data.

To read a table row, the read bit must be set when writing into the IP_PRIORITY register. This will trigger a read transaction to the address provided in bits 8:0 of the register. After this, reading the register provides the data returned from the table for this address.

When writing an entry into the table, software can only write the mapping for all ports in one write transaction. Therefore software must implement a read-modify-write scheme, to first read the current table contents, modify the priority bits for the port of interest without modifying the other ports bits and then write back the complete data word into the table.

29.4.6.4 Priority Resolution

The Priority Resolution function, for a Port n, is, on each port independently, programmable with the registers to enable or disable VLAN or IP or MAC address based classification.

The priority resolution follows the following ruleset depending on which classifications are enabled (PRIORITY_CFGn) and which fields are found within the frame:

- if IP classification is enabled and IP header found => map priority according IP_PRIORITY table

- else if VLAN classification is enabled and VLAN tag found => map priority according VLAN_PRIORITY table
- else if MAC classification is enabled and MAC address found => take priority from address table, if it is a static entry
- else use default priority as specified in PRIORITY_CFG.

29.4.6.5 Bridge Control Protocol Identification

To allow for implementation of bridge control protocols like the Spanning tree protocol, all control frames (Bridge Protocol Data Units, BPDU) are marked when they enter the switch. The mark then can be used by the Input Port Blocking function to drop the frame after the address learning (see also 5.10.5 page 43).

In addition, the function can be configured to pass all frames or to pass only control frames (that is, covering spanning tree port states such as blocking, listening, learning) and discard all other frames.

29.4.7 Input Port Selection

The port selection constantly checks (polling) all input ports for available data and if any data is available, a port is selected and frame data is read from the input. After one frame has been read, another port is selected, even if further data is available on the current port.

This means for the application on a port atlantic input interface, that it is not allowed to perform back-to-back frame transfers to the switch. Instead the application must wait for a new selection after one frame has been transferred.

29.4.8.1 Overview

A hash code is calculated using the frame destination MAC address. It is used as an entry (address) to a table, which contains, for each hash value MAC addresses with destination port number and validity information.

As one hash code value can represent more than one MAC address, the memory implements for each pointer, up to eight MAC address entries, which are searched linearly.

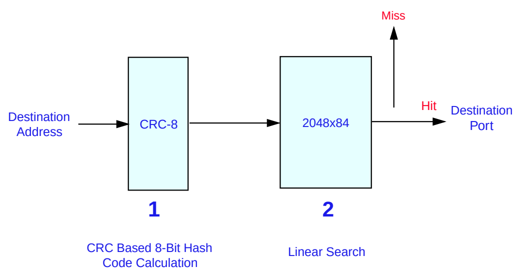


Figure 29-8. Port Look-Up Overview

29.4.8.2 Hash Code

For a MAC address table up to 2048 entries, an 8-bit hash value is calculated from the 48-bits of the destination MAC address. The hash code is using a CRC-8:

- x^8 + x^2 + x + 1 (0x07)

29.4.8.3 Address Memory

The address memory is divided into blocks. Each block contains eight records, which contain 64-bit of information each. Each record contains the 48-Bit MAC address and provides the necessary forwarding information as well as priority or time stamp information.

Two types of records are defined.

1. Dynamic Record: The dynamic entry provides the MAC address together with a 10-bit timestamp and destination port number. These entries are created by the learning function based on received frames to enable forwarding of frames to dedicated ports. Dynamic entries are deleted by the aging function if not updated.
2. Static Multiport/Priority Record: Switch Management can also write static entries in the table, which can include priority information as well as multiple destination ports for forwarding. The MAC address can be unicast or multicast. These records can be used to for example, specify the ports to participate in a specific multicast domain or to assign a MAC address based priority to a frame. The aging and learning functions ignore static records.

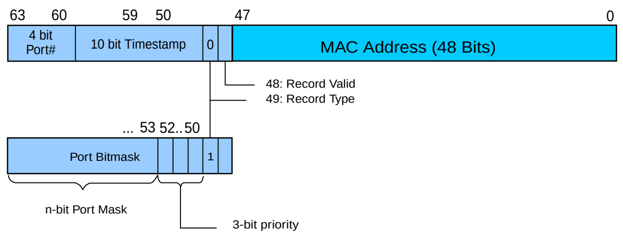


Figure 29-9. Address Memory Record Types

The record's bit 49 decides which type of record is found in the table:

- If 0, a dynamic entry is available and the 10-bit timestamp and 4-bit port number are given in the upper bits of the record.
- If 1, a static entry is available with a 3-bit priority field followed by a port bit mask of 3 bits. The record bit 53 represents Port 0, bit 54 Port 1 and bit 55 port 2. The frame will be forwarded to all ports that have a 1 in the port bitmask. The source port will be removed dynamically from the bitmask during forwarding (a frame is never forwarded to the port where it came from).

The 48-bit MAC address is stored with the first octet in bits 7:0 and the sixth octet in 47:40 of the record.

29.4.9.1 MAC Address Lookup

The 48-Bit destination MAC address of each frame received by the Switch, on any of its interfaces, is extracted by the Hardware and provided to the look-up engine together with the physical interface number. If the frame carries a VLAN tag, the tag information is also extracted and provided to the look-up engine.

If the received frame is a Unicast or Multicast frame, a Two-Stage lookup process is implemented. The look-up engine first calculates the hash value from the MAC address. The hash code is used as an entry to the Switch address table. The look-up function can provide three results with tree different associated actions performed by the switch hardware:

1. The address is in the table and associated with a correct port number:
    - The switch forwards the frame only to the looked up port.
2. The address is in the table but is associated with the port on which it was received:
    - The switch discards the frame and does not forward it to any port.
3. The address is not found in the table:

- The Switch engine sends the received frame to all ports except the port on which it was received (Flooding).

If a Broadcast frame is received, the switch Hardware sends the received frame to all output ports, except the one from which it was received (Flooding).

Note

Flooding and additional frame filtering can be controlled, for example, to avoid the duplication of critical information to unwanted destinations, with the mechanism described in Section 5.10.3 page 41.

29.4.9.2 Forced Forwarding

The MAC address lookup result can be overwritten using the forced forwarding configuration available in the register FORCE_FWD_P0. This feature is available only for frames coming from the local port (Port 0).

When forced forwarding is enabled for a frame, the frame will be forwarded to the forced destination port(s), ignoring any results from the MAC destination address lookup. Forced forwarding only replaces the mac lookup function, all other filtering functions (for example, VLAN verification) act as normal.

29.4.9.3 Learning

The Switch Hardware extracts the source MAC address of each frame received on each of the switch ports and provides it, through the registers LRN_REC_0 and LRN_REC_1, to the Switch Firmware which implements the learning task. The register LRN_STATUS indicates availability of learning data.

Learning Interface

The interface implements a FIFO buffer that stores up to 32 words of 32-bit each.

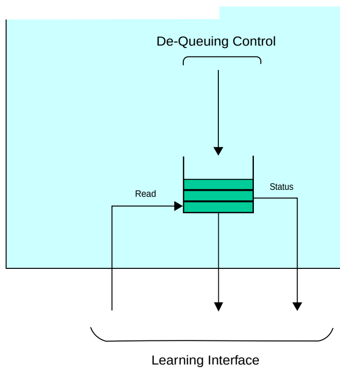


Figure 29-10. Learning Interface Overview

For each Frame processed by the switch engine, two 32-Bit records (Record A(0) and Record B(1)) are written in the Information FIFO, Record A is written first.

The MAC address available in Records A(0) and B(1) is the source MAC address of the Frame. Record A holds the first 4 bytes of the frame source address (1st=A[7:0], 4th=A[31:24]), Record B the last 2 bytes (5th=B[7:0], 6th=B[15:8]).

The Hash code in Record B is calculated, with the Source MAC Address, with the same Hash Polynomial used for Look-Up (as defined in 5.8.2 page 36) and the 4-Bit port number defines the Port / MAC Address association.

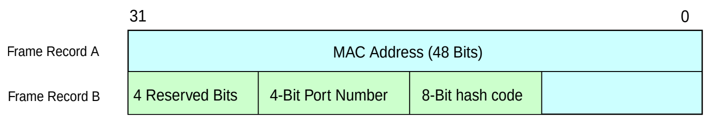


Figure 29-11. Frame Information Records - 8-Bit Hash Values

When information for at least one Frame (Two Records) is available, the status indication in register LRN_STATUS is set to '1'. To read the Frame records, the register LRN_REC_0 (Record A) must be read first followed by a read to LRN_REC_1 (Record B).

Note

A read to LRN_REC_1 triggers the retrieval of the next record pair from the FIFO if any.

The learning task (software) is using this information and then executes as follows:

1. For every frame received, the source address with port and timestamp information is stored in the address lookup table. The following information are stored for each entry:
    - MAC address: the 48-bit address is stored in the table.

- Time stamp: a 10-bit value, used to determine the age of an entry.
- Port number: a 4-bit value, which indicates the port through which the frame was received.

2. If there is no more space in the MAC address table, a new entry replaces the oldest entry with an identical matching hash value.

29.4.9.4 Migration

If the Firmware received a MAC address, which is already in the Switch table but is associated to a different physical port number, the current entry is overwritten with the new information and the timestamp is set to the current time.

29.4.9.5 Aging

Aging refers to deleting old entries within the table. It proceeds as follows:

1. The time stamp is stored with all the entries and is updated each time the source address appears again. The time stamp is a 10-bit value.
2. If a record is not updated for a longer period of time, it is removed from the table if it is a dynamic entry.
3. Static entries are not affected by the aging process and kept always.

This process runs continuously in the background of the firmware. The aging period is software controlled and programmable, defaulting to 4 seconds per step giving a range of 4 seconds to 68 minutes.

29.4.10.1 Overview

When an input port has been selected the frame is forwarded to its corresponding output port. Output port resolution and switching is based on the information from the two-stage MAC Address look-up (see 5.9 page 37) followed by additional resolution functions to allow frame duplication and flooding control, which are described in the following sections.

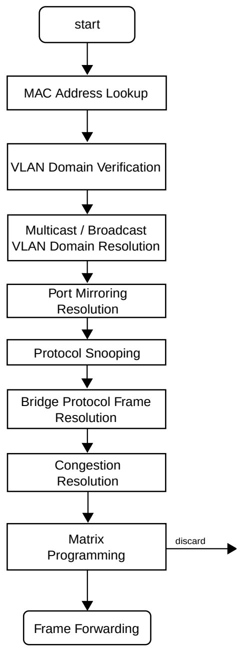


Figure 29-12. Frame Forwarding Tasks Overview

29.4.10.2 VLAN Domain Verification

When the L2 MAC Address Lookup was successful and identified a dedicated output port for the frame, the output and input port can be verified to be within the correct VLAN domain if VLAN verification is enabled for the port (register VLAN_VERIFY) and the frame contains a VLAN tag. The VLAN resolution table (see 5.10.3.2 page 41 below) is used as follows:

- If the frame's VLAN id is found within the table but the output port number is not a member of the VLAN domain, or the input port is not a member of the VLAN domain, the frame will be marked invalid and will be discarded eventually.
- If the frame's VLAN id is found within the table and the output port and input port are both members of the VLAN domain the frame will be forwarded normally.
- If the frame's VLAN id is not found in the VLAN table, or the frame has no VLAN tag, the frame will be forwarded normally (default broadcast domain), or, if the discard bit for the port is set (also in register VLAN_VERIFY) it will be discarded.

29.4.10.3 Broadcast Multicast VLAN Domain Resolution

29.4.10.3.1 Overview

To ensure that traffic within VLAN channels are always routed to the correct ports, for example to avoid the duplication of critical information through a Network, the Switch implements a resolution mechanism that, for any frame that is switched to multiple ports, checks the VLAN ID provided with the current frame.

The VLAN resolution mechanism searches the VLAN Resolution Table (See Registers VLAN_RES_TABLE_0 to VLAN_RES_TABLE_31), which stores up to 32 unique VLAN IDs, each associated to a port bit mask. The resolution mechanism is used for the following conditions:

- Unicast Frames with a destination MAC address that are not in the Table of the Layer 2 engine
- Mulicast Frames with a destination MAC address that are not in the Table of the Layer 2 engine
- Any Broadcast Frame

29.4.10.3.2 VLAN Resolution Table

The VLAN resolution Table (Registers VLAN_RES_TABLE_0 to VLAN_RES_TABLE_31) provides a unique VLAN ID / port bit mask association for up to 32 VLANs. A default entry (register BCAST_DEFAULT_MASK) provides an additional port bit mask. The port bit mask implements one bit for each port.

Each Port Bit indicates, if set to '1', that it is member of the VLAN and frames with the corresponding VLAN ID can be switched to the port. If the Port Bit is set to '0' a frame with the corresponding VLAN ID will not be switched to that port. If no VLAN ID matches, the default mask is applied.

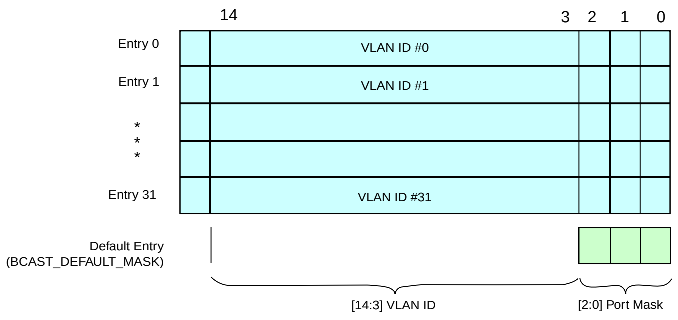


Figure 29-13. VLAN Resolution Table Overview

29.4.10.3.3 VLAN Switching and Resolution Mechanism

The VLAN table is used for both, VLAN domain verification (see section VLAN Domain Verification) as well as VLAN resolution. Once the frame has passed any VLAN domain verification (that is, will not be discarded by the verification function already) the forwarding resolution applies.

- If the destination MAC address (Unicast or Multicast) is found in the MAC address table and
  - The frame carries a VLAN tag that is found in the VLAN table, the frame can be forwarded only to the ports within the VLAN domain and will be discarded if the destination port is not member of the VLAN domain.
  - Otherwise, if the frame carries a VLAN tag that is not found in the VLAN table, or does not contain a VLAN tag, it is forwarded as indicated by the lookup table (note that VLAN domain verification can be configured to discard the frame in this case if enabled).
- If the destination MAC address (Unicast or Multicast) is not found in the MAC address table, or if the destination address is the Broadcast address, the frame is forwarded according to the following rules:
  - If the frame carries a VLAN tag, the VLAN resolution table is searched for a matching VLAN ID and the frame is sent to all ports that are associated with the VLAN ID.
  - If the frame carries a VLAN tag and the VLAN ID does not match any entry in the VLAN Resolution Table, the frame is forwarded to all ports that are enabled to receive broadcast frames by the default entry.
  - If the frame does not carry a VLAN tag the frame is forwarded to all ports that are enabled to receive broadcast frames by the default entry.
  - The frame is discarded, if it cannot be associated with any VLAN group and if the default (broadcast) group has been set to all zero.

To disable the VLAN resolution, set all VLAN IDs to 0xFFF or (0x000 if that id is not used) and all Port Mask bits to '1'. If the VLAN resolution is disabled, normal port flooding is implemented as described in section VLAN Priority Look-Up. The default entry can still be used to restrict broadcast to only dedicated ports, if not programmed to all '1'.

29.4.10.4 Port Mirroring

The function allows duplicating traffic to a dedicated mirror port. Any one of the ports can be assigned to act as a mirror port (register MIRROR_CONTROL).

The mirror port then is always added to the list of output ports and therefore receives a copy of the frame, if any of the following rules matches with the currently processed frame:

- Ingress Port Number Match

  When a frame is received on port N and the corresponding bit in the register MIRROR_ING_MAP is set to 1, the frame is mirrored.

- Egress Port Number Match

  When a frame is forwarded to port N and the corresponding bit in the register MIRROR_EG_MAP is set to 1, the frame will be mirrored.

- MAC Ingress SA Match

  When the Ingress Port Number match succeeded (see above) and the MAC source address matches the MIRROR_ISRC, the frame will be mirrored.

- MAC Ingress DA Match

  When the Ingress Port Number match succeeded (see above) and the MAC destination address matches the MIRROR_IDST, the frame will be mirrored.

- MAC Egress SA Match

  When the Egress Port Number match succeeded (see above) and the MAC source address matches the MIRROR_ESRC, the frame will be mirrored.

- MAC Egress DA Match

  When the Egress Port Number match succeeded (see above) and the MAC destination address matches the MIRROR_EDST, the frame will be mirrored.

In addition, a counter is implemented (Register MIRROR_CNT) that allows specifying that only every Nth frame that matches any of the above criteria is mirrored. If the counter is set to 1 or 0, every frame is mirrored that matches any of the above criteria.

29.4.10.5 Protocol Snooping

The incoming frames are parsed for IPv4 and IPv6 headers and UDP/TCP if available. The snooping function can be programmed to redirect specific protocols exclusively to the management port. See section IP Snooping and TCP/UDP Port Number Snooping for a description of the snooping options.

The snooping is active only for frames received from the external ports. When a frame is transmitted from the management port itself, snooping does not apply and the frames are forwarded normally (MAC lookup).

29.4.10.6 Bridge Protocol Frame Resolution

29.4.10.6.1 Overview

To implement bridge control protocols like the Spanning Tree protocol, the following control functions are performed by the Protocol Frame Resolution function:

29.4.10.6.2 Input Port Blocking

The input port blocking function is used to avoid forwarding of frames after address learning. The firmware can program the register INPUT_LEARN_BLOCK and if a frame is received on a port N that should be blocked (blocking bit N=1) and the frame is not a bridge protocol frame (see below), the frame will be marked for discard and will not be forwarded to any output port.

29.4.10.6.3 Input Port Learning Disable

To reduce processing load from the firmware, a port can be configured for exclusion from learning (see register INPUT_LEARN_BLOCK).

When learning is disabled on a port no source address extraction happens for incoming frames, with the exception of incoming BPDU frames. BPDU frame source addresses are always extracted and forwarded to the learning interface.

29.4.10.6.4 Management Port Forwarding

Bridge Protocol Frames are, if enabled, always forwarded to the dedicated management port (see MGMT_CONFIG) independent of any address lookup or other resolution functions.

Bridge Protocol Frames are identified by its destination address being any of the following:

- 01-80-c2-00-00-00 to

  01-80-c2-00-00-0F (Spanning Tree, IEEE 802.1d, Table 7-9)

- 01-80-c2-00-00-10 (Bridge Management Address, 802.1d, Table 7-10)
- "01-80-c2-00-00-20 to

  01-80-c2-00-00-2F (Generic Attribute Registration Protocol, 802.1d, Table 12-1)

29.4.10.6.5 Management Frame Forwarding

If the management port transmits Frames, they are forwarded according to the port mask defined in the configuration register MGMT_CONFIG. A handshaking mechanism is implemented that can be used by the firmware to configure the destination port mask on a frame-by-frame basis for management frames.

Note

VLAN domain verification/discard (see register VLAN_VERIFY) should be switched off for the management port to avoid that the switch discards management frames.

29.4.10.7 Congestion Resolution

29.4.10.7.1 Overview

The congestion resolution function is used whenever an output port is not available and data needs to be sent to that port. An output port is defined to be available if the port is enabled (bit in PORT_ENA set 1) and the output buffer (shared memory) is not congested. If, for a port, one of these conditions is not valid, the port is not available and frames cannot be switched to that port.

The congestion resolution function determines whether the frame should be processed further or discarded according to the following rules:

29.4.10.7.2 Unique Destination (one input to one output)

If the output port is enabled and can accept a frame the frame will be forwarded normally.

In any other case the frame will be discarded. If a frame switched to port N, the counter ODISCN is incremented.

29.4.10.7.3 Multiple Destinations (Flooding)

After broadcast / flooding resolution a frame needs to be switched to multiple output ports.

- Output disabled:

  All disabled ports are removed from the list of outputs.

- Output congestion:

If any of the outputs cannot accept a frame (as indicated by the output queue management for the port, implementation specific) it is also removed from the list of outputs.

If no output port is left in the list of outputs, the frame is discarded.

If a frame switched to port N, the counter ODISCN is incremented.

29.4.10.8 Switching

After the output port(s) have been determined, the switch control enables the corresponding path though the Switch Matrix and the frame is forwarded to the output queue(s).

In a similar fashion, if a Frame should be switched to multiple ports (for example, Broadcast), the switch control enables the corresponding paths though the Switch Matrix and the frame is forwarded to all the destination output ports.

29.5 Output Frame Queuing

29.5.1 Overview

The memory controller implements a shared memory architecture to store Frames of arbitrary size for multiple destination ports.

Each destination port implements 4 priority queues. The memory controller implements a single write input port and three output ports with the capability to perform virtual frame duplication on the output ports (Multiple reads on multiple ports of a single frame stored in the buffer).

A single large memory, partitioned in 256 byte cells, is implemented to efficiently share the available space for small and large frames without leaving large unused spaces when storing small frames.

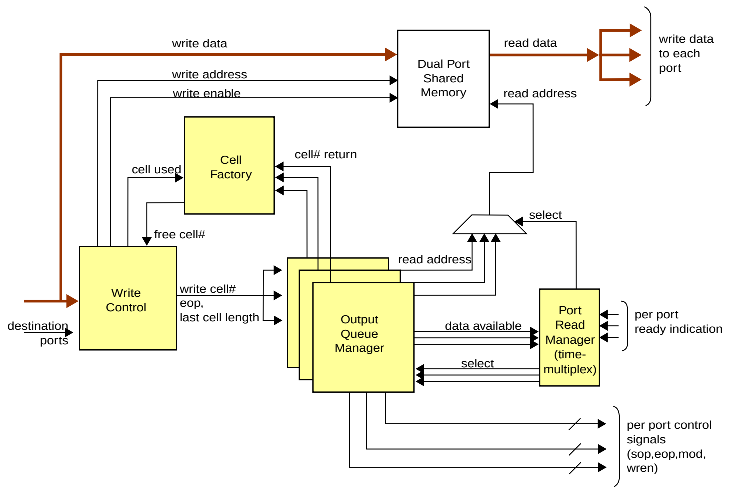


Figure 29-14. Memory Controller Overview

29.5.2 Cell and Queue Concept

The shared memory is partitioned in 256 byte cells using a 32-bit datapath implementation. This results in a cell holding 64 32-bit words.

Incoming frames are stored, partitioned in cells, in the shared memory and only the cell numbers are managed by the individual port queues.

Due to the arbitrary length of incoming frames, the last cell may not be fully utilized. A frame can spread from one to any number of cells. The number of bytes used in the last cell is also stored in the queue FIFO together with the individual cell numbers.

Cells can be stored anywhere in the shared memory. A single frame must not necessarily be stored in consecutive cells but instead can be scattered over the complete memory at arbitrary positions. The start of a cell is fixed to a 64-word boundary (that is, the memory start address of a cell is simply the cell # multiplied by 64).

Per port, a queue FIFO is implemented that stores the cell numbers for the frames The number of bytes used in the last cell is also stored in the queue FIFO together with the individual cell numbers.

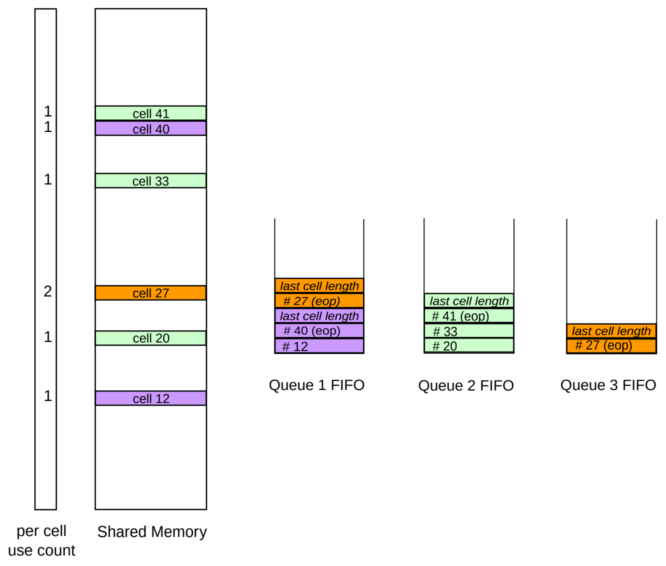


Figure 29-15. Cell Storage Concept

The example in Figure 29-15 shows the storage of three frames with one frame duplicated and stored in two queues (see the orange cells in the figure).

29.5.3 Write Control Module

The Write Control Module receives frames from the Switch engine, partitions and stores the frames in the shared memory. The cell numbers used to store the frame are forwarded to the port output managers.

29.5.4 Cell Factory Module

The Cell Factory implements the cell management, it always provides a free cell number to the write control module so, the write control module can immediately start writing into memory to avoid any write latency.

29.5.5.1 Overview

The Output Queue Manager implements, per port, the individual queue FIFOs (One queue FIFO per priority). The queue FIFOs are addressed by the priority information extracted by the Switch classification engine. Per port, eight prioritized queues are implemented.

When more than one queue has data available, the read logic selects one of the queues with a Weighted Fair Queuing scheduling algorithm.

29.5.5.2 Weighted Fair Queuing scheduling algorithm

The weight of each queue, common for all ports, can be configured between 0 and 30 with the register QM_WEIGHTS. A Queue with a higher weight is served more often than a queue with lower weight.

Queue 0 represents the lowest priority, Queue 3 the highest priority queue.

The scheduler first serves the queue with the highest weight. Each time the scheduler serves a queue, the weight of the other queues is increased and the weight of the selected queue is reset (decreased) to its programmed weight value. This guarantees that all queues are served eventually.

When multiple queues have the same weight, the queue with the higher number is served first.

If all weights are programmed to 0 (default), a strict priority scheme is active where the higher priority queues are served as long as they are not empty.

29.5.6 Congestion Management

The Write control logic is protected against memory overflow. When data is written is the cell factory has no more free cells (Number of available cells less than value programmed in register QMGR_MINCELLS), the frame is discarded or terminated with an error (i.e. forwarded to the output queue manager with the end-of-packet and error indication).

If the congestion persists, the Switch resolves the congestion as specified in section "Congestion Resolution".

29.5.7 Implementation Notes

Memory Size

The memory controller is capable of addressing up to 32768 bytes of memory. The external address bus to the shared memory provides the necessary bits to address a memory of size 8192x32 (deep x wide).

However, only 24 Kbyte will be used and the controller will only use addresses from 0 to 6143.

Datapath

The switch and all datapath has a width of 32 bit.

29.6 Reset and Stop Functions

29.6.1 Stop Controls

The register MODE_CONFIG offers several bits that control output pins. In addition, some controls have an effect on internal logic functions:

- stop_en: no internal function.
- switch_en: when de-asserted, all DMA registers are cleared.
- switch_reset: no internal function.

An external logic may use the controls to disable or enable the switch function as necessary.

29.6.2 Port Disable

The switch toplevel offers a disable input for each port (port_dis(2:0). When a pin is asserted (1), the corresponding port enable bits within register PORT_ENA for both transmit and receive will be cleared.

This results in the following behavior:

- If the port-enable bits of port0 are cleared, it also resets the input buffer and output buffer at the port0 DMA interface.
    - The ready output to DMA0 (ff_tx_rdy0) will be asserted allowing the application to continue writing data at the interface, which will be ignored (application flush). No further transmitted frame status (tx_ts_val0) will be given (that is, for any currently stored if any, as well as the currently ignored).
    - If the transmit enable is cleared while the interface is currently transferring a frame to the DMA, the frame is aborted (output buffer reset). The eop is not produced. Therefore, the connected DMA module must be reset to ensure proper restart after re-enabling the port.

- If any port's transmit enable bit is cleared, the shared memory will continue delivering the frames stored currently for a port as normal (that is, flushing the memory). New frames will be discarded before they are written into the shared memory. That is, no invalid frame will appear on the MAC interfaces after disabling or re-enabling a port.
- If any port's receive enable bit is cleared while a frame is transferred, this frame will be aborted with an error internally. The port's ready indication will stay asserted (output ff_tx_rdy=1) to flush any application data. Reenabling the port at any time will ignore any input data until a sop starts a new frame.

29.6.3 Port 0 Input Protection

The port 0 input buffer is protected for application errors that abort a frame without writing a proper eop to the interface. The next frame then written to the port 0 transmit interface will be concatenated with whatever data was already written before, but the frame will be marked with an error and hence will be forwarded and transmitted with an error indication (mii tx error).

If the port0 input buffer is reset (by deasserting PORT_ENA receive enable bit) while a frame is transferred to the switch internally, the frame transfer will be aborted in a clean way (producing an eop with error indication) to avoid blocking the switch.

29.6.4 Port 1,2 Input Protection

Ports 1 and 2 are protected for a second SOP in case the MAC is reset in the middle of a receive transaction to the switch hence did not produce a proper EOP to the switch. The next frame will be concatenated with whatever was provided to the switch before and marked with an error.

If the MAC is stopped in the middle of a transaction, the switch is blocked, waiting for the EOP, not serving any of the other ports. Clearing the PORT_ENA receive enable bit in this situation will terminate the frame with an error internally to the switch hence remove the blocking condition.

29.6.5 DMA Bus Error

When the DMA bus error input (dma_eberr_int) is asserted, the DMA registers are all cleared. If the corresponding interrupt was enabled the ipi_eberr_int pin will be asserted.

29.7 Firmware Architecture and Tasks

29.7.1 Overview

The switch firmware provides the necessary functions and a reference implementation to operate the switch.

29.7.2 Firmware Environment

The Switch is typically used together with an embedded 32-Bit processor subsystem, connected to the register, learning and address table interfaces. The firmware is available in generic C source code to allow for ease of use and migration to any processor environment.

29.7.3 Firmware Overview

The firmware can be used in a low memory footprint stand-alone C environment or together with any operating system. The firmware can be divided in the following functional modules.

29.7.3.1 Application Main

The task performs the following functions:

1. Initialization: On system startup, all the variables, tables and memory spaces are initialized and allocated.
2. Supervision: It takes care of time and ensures all other tasks are called to maintain runtime operation.

29.7.3.2 Learning Task

The learning task reads records from the learning interface and updates the address lookup table with new entries and updates timestamps of existing entries.

29.7.3.3 Timer Task

A timer task is called typically every 4 seconds (configurable) and increments the local time that is used for time stamping table entries. The aging process to determine old entries also uses the time.

29.7.3.4 Aging Task

The task periodically checks the address table for old entries. If an entry is older than a configurable maximum, the entry is removed from the table.

29.8 FIFO Interface Data Structure

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet Line. Byte 0 is sent and received first to/from the line.

Table 29-2. FIFO Interface Data Structure

	63	56	55	48	47	40	39	32	31	24	23	16	15	8	7	0
Word 0	Byte 7		Byte 6		Byte 5		Byte 4		Byte 3		Byte 2		Byte 1		Byte 0	
Word 1	Byte 15	Byte 14		Byte 13		Byte 12		Byte 11		Byte 10		Byte 9		Byte 8		
...	...															

The size of a Frame on the FIFO interface may not be a modulo of 64-Bit. Together with the last word of the frame, the valid byte(s) of data is (are) defined by the interface signal ff_tx_modN(2:0) for transmit and ff_rx_modN(2:0) for receive respectively.

Table 29-3. Transmit/Receive FIFO Interface Last Word Modulo Definition

..._mod(2:0)	Valid Bytes

000	ff_tx_data(63:0)
	ff_rx_data(63:0)

001	ff_tx_data(7:0)
	ff_rx_data(7:0)

010	ff_tx_data(15:0)
	ff_rx_data(15:0)

011	ff_tx_data(23:0)
	ff_rx_data(23:0)

100	ff_tx_data(31:0)
	ff_rx_data(31:0)

101	ff_tx_data(39:0)
	ff_rx_data(39:0)

..._mod(2:0)	Valid Bytes

110	ff_tx_data(47:0)
	ff_rx_data(47:0)

111	ff_tx_data(55:0)
	ff_rx_data(55:0)

29.9 Programmable Registers

The SWITCH module implements a register space of 1K 32-bit registers, base address is 0x800F8000. All writes must provide 32-bit of data and all reads return 32-bits of data. Reserved bits should be written with 0 and ignored on read to allow future extension. Unused registers read back 0 and a write has no effect.

HW_ENET_SWI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_8000	ENET SWI revision (HW_ENET_SWI_REVISION)	32	R	0001_0121h	29.9.1/1845
800F_8000	ENET SWI lookup MAC address memory start (HW_ENET_SWI_LOOKUP_MEMORY_START)	32	R/W	0000_0000h	29.9.118/1845
800F_8004	ENET SWI Scratch Register (HW_ENET_SWI_SCRATCH)	32	R/W	0000_0000h	29.9.2/1846
800F_8008	ENET SWI Port Enable Bits. (HW_ENET_SWI_PORT_ENA)	32	R/W	0000_0000h	29.9.3/1846
800F_8010	ENET SWI Verify VLAN domain. (HW_ENET_SWI_VLAN_VERIFY)	32	R/W	0000_0000h	29.9.4/1847
800F_8014	ENET SWI Default broadcast resolution. (HW_ENET_SWI_BCAST_DEFAULT_MASK)	32	R/W	0000_0000h	29.9.5/1848
800F_8018	ENET SWI Default multicast resolution. (HW_ENET_SWI_MCAST_DEFAULT_MASK)	32	R/W	0000_0000h	29.9.6/1849
800F_801C	ENET SWI Define port in blocking state and enable or disable learning. (HW_ENET_SWI_INPUT_LEARN_BLOCK)	32	R/W	0000_0000h	29.9.7/1850
800F_8020	ENET SWI Bridge Management Port Configuration. (HW_ENET_SWI_MGMT_CONFIG)	32	R/W	0000_0000h	29.9.8/1851
800F_8024	ENET SWI Defines several global configuration settings. (HW_ENET_SWI_MODE_CONFIG)	32	R/W	0000_0000h	29.9.9/1853
800F_8028	ENET SWI Define behavior of VLAN input manipulation function (HW_ENET_SWI_VLAN_IN_MODE)	32	R/W	0000_0000h	29.9.10/1854
800F_802C	ENET SWI Define behavior of VLAN output manipulation function (HW_ENET_SWI_VLAN_OUT_MODE)	32	R/W	0000_0000h	29.9.11/1855
800F_8030	ENET SWI Enable the input processing according to the VLAN_IN_MODE for a port (HW_ENET_SWI_VLAN_IN_MODE_ENA)	32	R/W	0000_0000h	29.9.12/1856

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_8034	ENET SWI The VLAN type field value to expect to identify a VLAN tagged frame. (HW_ENET_SWI_VLAN_TAG_ID)	32	R/W	0000_8100h	29.9.13/1857
800F_8040	ENET SWI Port Mirroring configuration. (HW_ENET_SWI_MIRROR_CONTROL)	32	R/W	0000_0000h	29.9.14/1857
800F_8044	ENET SWI Port Mirroring Egress port definitions. (HW_ENET_SWI_MIRROR_EG_MAP)	32	R/W	0000_0000h	29.9.15/1859
800F_8048	ENET SWI Port Mirroring Ingress port definitions. (HW_ENET_SWI_MIRROR_ING_MAP)	32	R/W	0000_0000h	29.9.16/1860
800F_804C	ENET SWI Ingress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ISRC_0)	32	R/W	0000_0000h	29.9.17/1861
800F_8050	ENET SWI Ingress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ISRC_1)	32	R/W	0000_0000h	29.9.18/1861
800F_8054	ENET SWI Ingress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_IDST_0)	32	R/W	0000_0000h	29.9.19/1862
800F_8058	ENET SWI Ingress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_IDST_1)	32	R/W	0000_0000h	29.9.20/1862
800F_805C	ENET SWI Egress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ESRC_0)	32	R/W	0000_0000h	29.9.21/1863
800F_8060	ENET SWI Egress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ESRC_1)	32	R/W	0000_0000h	29.9.22/1863
800F_8064	ENET SWI Egress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_EDST_0)	32	R/W	0000_0000h	29.9.23/1864
800F_8068	ENET SWI Egress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_EDST_1)	32	R/W	0000_0000h	29.9.24/1864
800F_806C	ENET SWI Count Value for Mirror filtering. (HW_ENET_SWI_MIRROR_CNT)	32	R/W	0000_0000h	29.9.25/1865
800F_8080	ENET SWI Memory Manager Status. (HW_ENET_SWI_OQMGR_STATUS)	32	R/W	0060_004Ah	29.9.26/1865
800F_8084	ENET SWI Low Memory threshold. (HW_ENET_SWI_QMGR_MINCELLS)	32	R/W	0000_0009h	29.9.27/1867
800F_8088	ENET SWI Statistic providing the lowest number of free cells reached in memory (HW_ENET_SWI_QMGR_ST_MINCELLS)	32	R/W	0000_0000h	29.9.28/1867
800F_808C	ENET SWI Port Congestion status (internal). (HW_ENET_SWI_QMGR_CONGEST_STAT)	32	R	0000_0000h	29.9.29/1868
800F_8090	ENET SWI Switch input and output interface status (internal). (HW_ENET_SWI_QMGR_IFACE_STAT)	32	R	0000_0007h	29.9.30/1869
800F_8094	ENET SWI Queue weights for each queue. (HW_ENET_SWI_QM_WEIGHTS)	32	R/W	0000_0000h	29.9.31/1870
800F_809C	ENET SWI Define congestion threshold for Port0 backpressure. (HW_ENET_SWI_QMGR_MINCELLSP0)	32	R/W	0000_0009h	29.9.32/1871
800F_80BC	ENET SWI Enable forced forwarding for a frame processed from port 0 (HW_ENET_SWI_FORCE_FWD_P0)	32	R/W	0000_0000h	29.9.33/1871

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_80C0	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP1)	32	R/W	0000_0000h	29.9.34/1872
800F_80C4	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP2)	32	R/W	0000_0000h	29.9.35/1874
800F_80C8	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP3)	32	R/W	0000_0000h	29.9.36/1875
800F_80CC	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP4)	32	R/W	0000_0000h	29.9.37/1876
800F_80D0	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP5)	32	R/W	0000_0000h	29.9.38/1877
800F_80D4	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP6)	32	R/W	0000_0000h	29.9.39/1878
800F_80D8	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP7)	32	R/W	0000_0000h	29.9.40/1880
800F_80DC	ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP8)	32	R/W	0000_0000h	29.9.41/1881
800F_80E0	ENET SWI IP Snooping function1 (HW_ENET_SWI_IPSNOOP1)	32	R/W	0000_0000h	29.9.42/1882
800F_80E4	ENET SWI IP Snooping function2 (HW_ENET_SWI_IPSNOOP2)	32	R/W	0000_0000h	29.9.43/1883
800F_80E8	ENET SWI IP Snooping function3 (HW_ENET_SWI_IPSNOOP3)	32	R/W	0000_0000h	29.9.44/1884
800F_80EC	ENET SWI IP Snooping function4 (HW_ENET_SWI_IPSNOOP4)	32	R/W	0000_0000h	29.9.45/1885
800F_80F0	ENET SWI IP Snooping function5 (HW_ENET_SWI_IPSNOOP5)	32	R/W	0000_0000h	29.9.46/1886
800F_80F4	ENET SWI IP Snooping function6 (HW_ENET_SWI_IPSNOOP6)	32	R/W	0000_0000h	29.9.47/1887
800F_80F8	ENET SWI IP Snooping function7 (HW_ENET_SWI_IPSNOOP7)	32	R/W	0000_0000h	29.9.48/1888
800F_80FC	ENET SWI IP Snooping function8 (HW_ENET_SWI_IPSNOOP8)	32	R/W	0000_0000h	29.9.49/1889
800F_8100	ENET SWI Port 0 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY0)	32	R/W	0000_0000h	29.9.50/1890
800F_8104	ENET SWI Port 1 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY1)	32	R/W	0000_0000h	29.9.51/1891
800F_8108	ENET SWI Port 2 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY2)	32	R/W	0000_0000h	29.9.52/1892
800F_8140	ENET SWI IPv4 and IPv6 priority resolution table programming (HW_ENET_SWI_IP_PRIORITY)	32	R/W	0000_0000h	29.9.53/1893
800F_8180	ENET SWI Port 0 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG0)	32	R/W	0000_0000h	29.9.54/1894

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_8184	ENET SWI Port 1 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG1)	32	R/W	0000_0000h	29.9.55/1896
800F_8188	ENET SWI Port 2 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG2)	32	R/W	0000_0000h	29.9.56/1897
800F_8200	ENET SWI Port 0 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO0)	32	R/W	0000_0000h	29.9.57/1898
800F_8204	ENET SWI Port 1 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO1)	32	R/W	0000_0000h	29.9.58/1898
800F_8208	ENET SWI Port 2 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO2)	32	R/W	0000_0000h	29.9.59/1899
800F_8280	ENET SWI VLAN domain resolution entry 0. (HW_ENET_SWI_VLAN_RES_TABLE_0)	32	R/W	0000_0000h	29.9.60/1900
800F_8284	ENET SWI VLAN domain resolution entry 1. (HW_ENET_SWI_VLAN_RES_TABLE_1)	32	R/W	0000_0000h	29.9.61/1900
800F_8288	ENET SWI VLAN domain resolution entry 2. (HW_ENET_SWI_VLAN_RES_TABLE_2)	32	R/W	0000_0000h	29.9.62/1901
800F_828C	ENET SWI VLAN domain resolution entry 3. (HW_ENET_SWI_VLAN_RES_TABLE_3)	32	R/W	0000_0000h	29.9.63/1902
800F_8290	ENET SWI VLAN domain resolution entry 4. (HW_ENET_SWI_VLAN_RES_TABLE_4)	32	R/W	0000_0000h	29.9.64/1902
800F_8294	ENET SWI VLAN domain resolution entry 5. (HW_ENET_SWI_VLAN_RES_TABLE_5)	32	R/W	0000_0000h	29.9.65/1903
800F_8298	ENET SWI VLAN domain resolution entry 6. (HW_ENET_SWI_VLAN_RES_TABLE_6)	32	R/W	0000_0000h	29.9.66/1904
800F_829C	ENET SWI VLAN domain resolution entry 7. (HW_ENET_SWI_VLAN_RES_TABLE_7)	32	R/W	0000_0000h	29.9.67/1904
800F_82A0	ENET SWI VLAN domain resolution entry 8. (HW_ENET_SWI_VLAN_RES_TABLE_8)	32	R/W	0000_0000h	29.9.68/1905
800F_82A4	ENET SWI VLAN domain resolution entry 9. (HW_ENET_SWI_VLAN_RES_TABLE_9)	32	R/W	0000_0000h	29.9.69/1906
800F_82A8	ENET SWI VLAN domain resolution entry 10. (HW_ENET_SWI_VLAN_RES_TABLE_10)	32	R/W	0000_0000h	29.9.70/1906
800F_82AC	ENET SWI VLAN domain resolution entry 11. (HW_ENET_SWI_VLAN_RES_TABLE_11)	32	R/W	0000_0000h	29.9.71/1907
800F_82B0	ENET SWI VLAN domain resolution entry 12. (HW_ENET_SWI_VLAN_RES_TABLE_12)	32	R/W	0000_0000h	29.9.72/1908
800F_82B4	ENET SWI VLAN domain resolution entry 13. (HW_ENET_SWI_VLAN_RES_TABLE_13)	32	R/W	0000_0000h	29.9.73/1908
800F_82B8	ENET SWI VLAN domain resolution entry 14. (HW_ENET_SWI_VLAN_RES_TABLE_14)	32	R/W	0000_0000h	29.9.74/1909
800F_82BC	ENET SWI VLAN domain resolution entry 15. (HW_ENET_SWI_VLAN_RES_TABLE_15)	32	R/W	0000_0000h	29.9.75/1910

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_82C0	ENET SWI VLAN domain resolution entry 16. (HW_ENET_SWI_VLAN_RES_TABLE_16)	32	R/W	0000_0000h	29.9.76/1910
800F_82C4	ENET SWI VLAN domain resolution entry 17. (HW_ENET_SWI_VLAN_RES_TABLE_17)	32	R/W	0000_0000h	29.9.77/1911
800F_82C8	ENET SWI VLAN domain resolution entry 18. (HW_ENET_SWI_VLAN_RES_TABLE_18)	32	R/W	0000_0000h	29.9.78/1912
800F_82CC	ENET SWI VLAN domain resolution entry 19. (HW_ENET_SWI_VLAN_RES_TABLE_19)	32	R/W	0000_0000h	29.9.79/1912
800F_82D0	ENET SWI VLAN domain resolution entry 20. (HW_ENET_SWI_VLAN_RES_TABLE_20)	32	R/W	0000_0000h	29.9.80/1913
800F_82D4	ENET SWI VLAN domain resolution entry 21. (HW_ENET_SWI_VLAN_RES_TABLE_21)	32	R/W	0000_0000h	29.9.81/1914
800F_82D8	ENET SWI VLAN domain resolution entry 22. (HW_ENET_SWI_VLAN_RES_TABLE_22)	32	R/W	0000_0000h	29.9.82/1914
800F_82DC	ENET SWI VLAN domain resolution entry 23. (HW_ENET_SWI_VLAN_RES_TABLE_23)	32	R/W	0000_0000h	29.9.83/1915
800F_82E0	ENET SWI VLAN domain resolution entry 24. (HW_ENET_SWI_VLAN_RES_TABLE_24)	32	R/W	0000_0000h	29.9.84/1916
800F_82E4	ENET SWI VLAN domain resolution entry 25. (HW_ENET_SWI_VLAN_RES_TABLE_25)	32	R/W	0000_0000h	29.9.85/1916
800F_82E8	ENET SWI VLAN domain resolution entry 26. (HW_ENET_SWI_VLAN_RES_TABLE_26)	32	R/W	0000_0000h	29.9.86/1917
800F_82EC	ENET SWI VLAN domain resolution entry 27. (HW_ENET_SWI_VLAN_RES_TABLE_27)	32	R/W	0000_0000h	29.9.87/1918
800F_82F0	ENET SWI VLAN domain resolution entry 28. (HW_ENET_SWI_VLAN_RES_TABLE_28)	32	R/W	0000_0000h	29.9.88/1918
800F_82F4	ENET SWI VLAN domain resolution entry 29. (HW_ENET_SWI_VLAN_RES_TABLE_29)	32	R/W	0000_0000h	29.9.89/1919
800F_82F8	ENET SWI VLAN domain resolution entry 30. (HW_ENET_SWI_VLAN_RES_TABLE_30)	32	R/W	0000_0000h	29.9.90/1920
800F_82FC	ENET SWI VLAN domain resolution entry 31. (HW_ENET_SWI_VLAN_RES_TABLE_31)	32	R/W	0000_0000h	29.9.91/1920
800F_8300	ENET SWI Total number of incoming frames processed (HW_ENET_SWI_TOTAL_DISC)	32	R	0000_0000h	29.9.92/1921
800F_8304	ENET SWI Sum of bytes of frames counted in TOTAL_DISC (HW_ENET_SWI_TOTAL_BYT_DISC)	32	R	0000_0000h	29.9.93/1921
800F_8308	ENET SWI Total number of incoming frames processed (HW_ENET_SWI_TOTAL_FRM)	32	R	0000_0000h	29.9.94/1922
800F_830C	ENET SWI Sum of bytes of frames counted in TOTAL_FRM (HW_ENET_SWI_TOTAL_BYT_FRM)	32	R	0000_0000h	29.9.95/1922
800F_8310	ENET SWI Port 0 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC0)	32	R	0000_0000h	29.9.96/1923

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_8314	ENET SWI Port 0 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN0)	32	R	0000_0000h	29.9.97/1923
800F_8318	ENET SWI Port 0 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED0)	32	R	0000_0000h	29.9.98/1924
800F_831C	ENET SWI Port 0 incoming frames discarded (after learning) as port is configured in blocking mode (HW_ENET_SWI_IDISC_BLOCKED0)	32	R	0000_0000h	29.9.99/1924
800F_8320	ENET SWI Port 1 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC1)	32	R	0000_0000h	29.9.100/1925
800F_8324	ENET SWI Port 1 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN1)	32	R	0000_0000h	29.9.101/1925
800F_8328	ENET SWI Port 1 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED1)	32	R	0000_0000h	29.9.102/1926
800F_832C	ENET SWI Port 1 incoming frames discarded (after learning) as port is configured in blocking mode (HW_ENET_SWI_IDISC_BLOCKED1)	32	R	0000_0000h	29.9.103/1926
800F_8330	ENET SWI Port 2 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC2)	32	R	0000_0000h	29.9.104/1927
800F_8334	ENET SWI Port 2 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN2)	32	R	0000_0000h	29.9.105/1927
800F_8338	ENET SWI Port 2 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED2)	32	R	0000_0000h	29.9.106/1928
800F_833C	ENET SWI Port 2 incoming frames discarded (after learning) as port is configured in blocking mod (HW_ENET_SWI_IDISC_BLOCKED2)	32	R	0000_0000h	29.9.107/1928
800F_8400	ENET SWI Interrupt Event Register (HW_ENET_SWI_EIR)	32	R/W	0000_0020h	29.9.108/1929
800F_8404	ENET SWI Interrupt Mask Register (HW_ENET_SWI_EIMR)	32	R/W	0000_0000h	29.9.109/1930
800F_8408	ENET SWI Pointer to Receive Descriptor Ring (HW_ENET_SWI_ERDSR)	32	R/W	0000_0000h	29.9.110/1931
800F_840C	ENET SWI Pointer to Transmit Descriptor Ring (HW_ENET_SWI_ETDSR)	32	R/W	0000_0000h	29.9.111/1932
800F_8410	ENET SWI Maximum Receive Buffer Size (HW_ENET_SWI_EMRBR)	32	R/W	0000_0000h	29.9.112/1932
800F_8414	ENET SWI Receive Descriptor Active Register (HW_ENET_SWI_RDAR)	32	R/W	0000_0000h	29.9.113/1933
800F_8418	ENET SWI Transmit Descriptor Active Register (HW_ENET_SWI_TDAR)	32	R/W	0000_0000h	29.9.114/1933
800F_8500	ENET SWI Learning Records A (0) and B (1) (HW_ENET_SWI_LRN_REC_0)	32	R	0000_0000h	29.9.115/1934

HW_ENET_SWI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
800F_8504	ENET SWI Learning Record B(1) (HW_ENET_SWI_LRN_REC_1)	32	R	0000_0000h	29.9.116/1934
800F_8508	ENET SWI Learning data available status. (HW_ENET_SWI_LRN_STATUS)	32	R	0000_0000h	29.9.117/1935
8010_7FFC	ENET SWI lookup MAC address memory end (HW_ENET_SWI_LOOKUP_MEMORY_END)	32	R/W	0000_0000h	29.9.119/1936

29.9.1 ENET SWI revision (HW_ENET_SWI_REVISION)

Address: HW_ENET_SWI_REVISION – 800F_8000h base + 0h offset = 800F_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CUSTOMER_REVISION																CORE_REVISION															
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1

HW_ENET_SWI_REVISION field descriptions

Field	Description
31–16 CUSTOMER_ REVISION	Customer Revision
15–0 CORE_ REVISION	Core Revision

29.9.118 ENET SWI lookup MAC address memory start (HW_ENET_SWI_LOOKUP_MEMORY_START)

Address: HW_ENET_SWI_LOOKUP_MEMORY_START – 800F_8000h base + 0h offset = 800F_8000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	MEMORY_DATA																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_LOOKUP_MEMORY_START field descriptions

Field	Description
31–0 MEMORY_DATA	Memroy cell.

29.9.2 ENET SWI Scratch Register (HW_ENET_SWI_SCRATCH)

The Scratch Register provides a memory location to test the register access. It returns all data written to it in inverted form.

Address: HW_ENET_SWI_SCRATCH – 800F_8000h base + 4h offset = 800F_8004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SCRATCH																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_SCRATCH field descriptions

Field	Description
31–0 SCRATCH	The Scratch Register provides a memory location to test the register access. It returns all data written to it in inverted form.

29.9.3 ENET SWI Port Enable Bits. (HW_ENET_SWI_PORT_ENA)

Port Enable Bits. Two bits per port. The transmit and receive direction for each port can be independently enabled. When the transmit direction is enabled a frame can be forwarded to the port. When disabled, all frames forwarded to the port are discarded. When the receive direction is enabled the port is selected and a frame is accepted if it indicates data available. If the receive direction is disabled the input is ignored and never selected for frame reception.

Address: HW_ENET_SWI_PORT_ENA – 800F_8000h base + 8h offset = 800F_8008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1													ENA_ RECEIVE_2	ENA_ RECEIVE_1	ENA_ RECEIVE_0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0													ENA_ TRANSMIT_2	ENA_ TRANSMIT_1	ENA_ TRANSMIT_0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORT_ENA field descriptions

Field	Description
31–19 RSRVD1	Reserved bits. Write as 0.
18 ENA_RECEIVE_2	enable receive on port 2.
17 ENA_RECEIVE_1	enable receive on port 1.
16 ENA_RECEIVE_0	enable receive on port 0.
15–3 RSRVD0	Reserved bits. Write as 0.
2 ENA_ TRANSMIT_2	enable transmit on port 2.
1 ENA_ TRANSMIT_1	enable transmit on port 1.
0 ENA_ TRANSMIT_0	enable transmit on port 0.

29.9.4 ENET SWI Verify VLAN domain. (HW_ENET_SWI_VLAN_VERIFY)

Address: HW_ENET_SWI_VLAN_VERIFY – 800F_8000h base + 10h offset = 800F_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1													DISCARD_P2	DISCARD_P1	DISCARD_P0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0													VLAN_ VERIFY_2	VLAN_ VERIFY_1	VLAN_ VERIFY_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_VERIFY field descriptions

Field	Description
31–19 RSRVD1	Reserved bits. Write as 0.
18 DISCARD_P2	Discard unknown on port 2. When set, and a frame is received with a VLAN ID that is unknown, or has no VLAN tag, the frame is discarded and not forwarded (i.e. the default bcast is ignored).
17 DISCARD_P1	Discard unknown on port 1. When set, and a frame is received with a VLAN ID that is unknown, or has no VLAN tag, the frame is discarded and not forwarded (i.e. the default bcast is ignored).
16 DISCARD_P0	Discard unknown on port 0. When set, and a frame is received with a VLAN ID that is unknown, or has no VLAN tag, the frame is discarded and not forwarded (i.e. the default bcast is ignored).
15–3 RSRVD0	Reserved bits. Write as 0.
2 VLAN_VERIFY_2	Verify VLAN domain on port 2. When enabled (Bit=1) a frame is accepted from the port as valid only when the input and output ports are members of the VLAN domain of the frame. When disabled (Bit=0) frames are routed to the output port without VLAN domain checking.
1 VLAN_VERIFY_1	Verify VLAN domain on port 1. When enabled (Bit=1) a frame is accepted from the port as valid only when the input and output ports are members of the VLAN domain of the frame. When disabled (Bit=0) frames are routed to the output port without VLAN domain checking.
0 VLAN_VERIFY_0	Verify VLAN domain on port 0. When enabled (Bit=1) a frame is accepted from the port as valid only when the input and output ports are members of the VLAN domain of the frame. When disabled (Bit=0) frames are routed to the output port without VLAN domain checking.

29.9.5 ENET SWI Default broadcast resolution. (HW_ENET_SWI_BCAST_DEFAULT_MASK)

For broadcast/flooding resolution, the default output port list

Address: HW_ENET_SWI_BCAST_DEFAULT_MASK – 800F_8000h base + 14h offset = 800F_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													BCAST_ DEFAULT_ MASK_2	BCAST_ DEFAULT_ MASK_1	BCAST_ DEFAULT_ MASK_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_BCAST_DEFAULT_MASK field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 BCAST_ DEFAULT_ MASK_2	port 2
1 BCAST_ DEFAULT_ MASK_1	port 1
0 BCAST_ DEFAULT_ MASK_0	port 0

29.9.6 ENET SWI Default multicast resolution. (HW_ENET_SWI_MCAST_DEFAULT_MASK)

Used for broadcast/flooding resolution. The default output port list used instead of the BCST_DEFAULT_MASK, when the received frame carries a multicast address.

Address: HW_ENET_SWI_MCAST_DEFAULT_MASK – 800F_8000h base + 18h offset = 800F_8018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													MCAST_ DEFAULT_ MASK_2	MCAST_ DEFAULT_ MASK_1	MCAST_ DEFAULT_ MASK_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

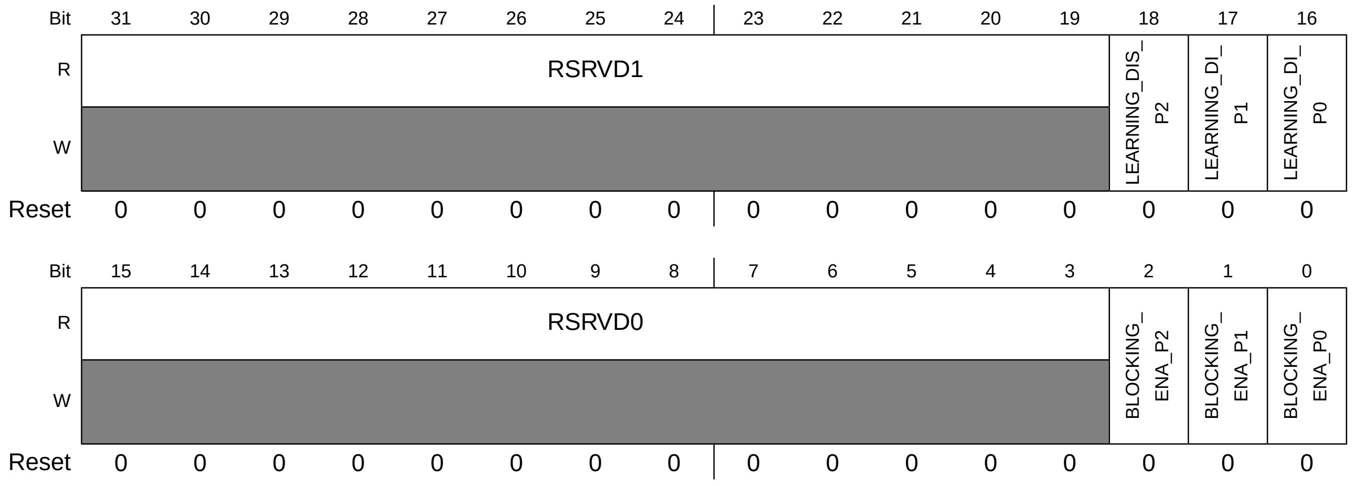
HW_ENET_SWI_MCAST_DEFAULT_MASK field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 MCAST_ DEFAULT_ MASK_2	port 2
1 MCAST_ DEFAULT_ MASK_1	port 1
0 MCAST_ DEFAULT_ MASK_0	port 0

29.9.7 ENET SWI Define port in blocking state and enable or disable learning. (HW_ENET_SWI_INPUT_LEARN_BLOCK)

When blocking is enabled for a port (Bit=1) only Bridge Protocol data units are accepted on that input, all other frames are discarded. When learning is disabled for a port (Bit=1), only Bridge Protocol Data Unit frames will be learned. Other frames will be ignored for learning. Both functions operate independently from each other.

Address: HW_ENET_SWI_INPUT_LEARN_BLOCK – 800F_8000h base + 1Ch offset = 800F_801Ch



HW_ENET_SWI_INPUT_LEARN_BLOCK field descriptions

Field	Description
31–19 RSRVD1	Reserved bits. Write as 0.
18 LEARNING_DIS_P2	disable learning on port 2.
17 LEARNING_DI_P1	disable learning on port 1.
16 LEARNING_DI_P0	disable learning on port 0.
15–3 RSRVD0	Reserved bits. Write as 0.
2 BLOCKING_ENA_P2	enable blocking on port 2.
1 BLOCKING_ENA_P1	enable blocking on port 1.
0 BLOCKING_ENA_P0	enable blocking on port 0.

29.9.8 ENET SWI Bridge Management Port Configuration. (HW_ENET_SWI_MGMT_CONFIG)

Enables and defines the management port that receives Bridge Protocol Frames

Address: HW_ENET_SWI_MGMT_CONFIG – 800F_8000h base + 20h offset = 800F_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2													PORTMASK		
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRIORITY			RSRVD1					DISCARD	ENABLE	MESSAGE_ TRANSMITTED	RSRVD0	PORT			
W				▨								▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

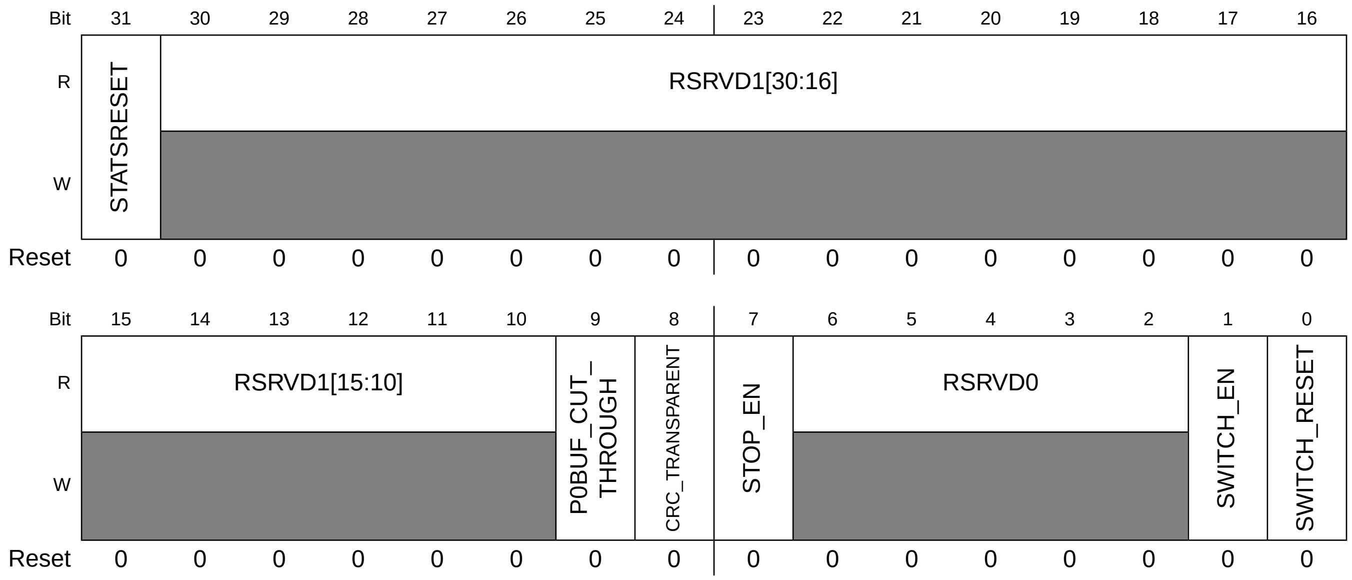
HW_ENET_SWI_MGMT_CONFIG field descriptions

Field	Description
31–19 RSRVD2	Reserved bits. Write as 0.
18–16 PORTMASK	Portmask for transmission of BPDU frames as defined in 5.10.6.4 page 44. When the management port transmits a BPDU frame to the switch, it is forwarded to all ports in this portmask (bit16=port0, bit17=port1, bit 18=port2).
15–13 PRIORITY	Priority to use for transmitted management frames. Used to e.g. put a management frame in a high-priority output queue for fast delivery.
12–8 RSRVD1	Reserved bits. Write as 0.
7 DISCARD	If set, BPDU frames are discarded always. Setting has no effect, when the enable bit is set.
6 ENABLE	If set, all BPDU frames are forwarded exclusively to the management port specified in bits 3:0. If cleared, BPDU frames are forwarded as any other frame, or discarded if the discard bit is set.
5 MESSAGE_ TRANSMITTED	Set (latched) when a BPDU frame as defined in 5.10.6.4 page 44 was transmitted from the management port to any output port. Bit is reset by writing into the register.
4 RSRVD0	Reserved bits. Write as 0.
3–0 PORT	The Port number of the port that should act as a management port. Relevant to all functions that forward frames to the management port (i.e. BPDU processing, snooping). Note: It must be set 0 in this switch configuration (Port 0 to DMA0 is the management port).

29.9.9 ENET SWI Defines several global configuration settings. (HW_ENET_SWI_MODE_CONFIG)

Defines several global configuration settings.

Address: HW_ENET_SWI_MODE_CONFIG – 800F_8000h base + 24h offset = 800F_8024h



HW_ENET_SWI_MODE_CONFIG field descriptions

Field	Description
31 STATSRESET	Reset Statistics Counters Command. When set during a write, all statistics counters are cleared. When set, all other bits are ignored and do not influence the currently stored value in the register (i.e. it is not necessary to read/preserve the register contents prior to writing this bit).
30–10 RSRVD1	Reserved bits. Write as 0.
9 P0BUF_CUT_ THROUGH	Enable Port0 input buffer cut-through mode. Cut-through mode may be used for allowing jumbo frames to be transferred. When cleared (0, default) the input buffer operates in store/forward mode, which is the recommended mode of operation.
8 CRC_ TRANSPARENT	When enabled (1) the MAC ports are expected to process frames to/from the switch including CRC. Frames from a MAC port to a MAC port will then have their respective ff_rx_crc_fwd1/2 pin (wired to MAC's ff_tx_crc_fwd) asserted, indicating that no CRC should be appended. However, even when enabled, the DMA0 port is not expected to provide frames with crc: Frames from port 0 (DMA0 transmit) are always forwarded with the crc option as defined by the input ff_tx_crc_fwd0, independent of the crc_transparent setting. This means, if the DMA0 port will never transmit frames with CRC, ff_tx_crc_fwd0 can be wired to permanently 0. Note that the MAC configuration bit RCR(CRC_FWD) must be set to ensure the MAC forwards received frames with CRC to the switch.

HW_ENET_SWI_MODE_CONFIG field descriptions (continued)

Field	Description
	When disabled (0, default) the output pins ff_rx_crc_fwd1/2 to the MACs are always 0 to ensure a CRC is appended to outgoing frames no matter from which port they were received. DMA0 still can influence the crc append through its ff_tx_crc_fwd0 input pin if required. Note: The ff_tx_crc_fwd0 input to the switch must be valid throughout the complete frame (from sop to eop). This is in contrast to the MAC definition for this signal, which requires validity during eop only.
7 STOP_EN	Controls toplevel output pin stop_en. No internal function.
6–2 RSRVD0	Reserved bits. Write as 0.
1 SWITCH_EN	Controls toplevel output pin switch_en. When deasserted (0), all DMA registers are cleared.
0 SWITCH_RESET	Controls toplevel output pin switch_reset. No internal function.

29.9.10 ENET SWI Define behavior of VLAN input manipulation function (HW_ENET_SWI_VLAN_IN_MODE)

Define behavior of VLAN input manipulation function, if such a function is present on an input port.

Each input port can operate in one of four modes individually defining 2 bits per port: "00": Mode 1, Single Tag passthrough "01": Mode 2, Single Tag overwrite "10": Mode 3, Double Tag passthrough "11": Mode 4, Double Tag overwrite The settings have effect only, when enabled by setting the corresponding bit in VLAN_IN_MODE_ENA below.

Address: HW_ENET_SWI_VLAN_IN_MODE – 800F_8000h base + 28h offset = 800F_8028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:6]										VLAN_IN_MODE_2		VLAN_IN_MODE_1		VLAN_IN_MODE_0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_IN_MODE field descriptions

Field	Description
31–6 RSRVD0	Reserved bits. Write as 0.
5–4 VLAN_IN_ MODE_2	port 2.
3–2 VLAN_IN_ MODE_1	port 1.
1–0 VLAN_IN_ MODE_0	port 0.

29.9.11 ENET SWI Define behavior of VLAN output manipulation function (HW_ENET_SWI_VLAN_OUT_MODE)

Define behavior of VLAN output manipulation function, if such a function is present on an output port.

Each output port can operate in one of three modes individually defining 2 bits per port: "00": no output manipulation "01": Mode 1, Strip Mode "10": Mode 2, Tag Thru "11": Mode 3, Transparent

Address: HW_ENET_SWI_VLAN_OUT_MODE – 800F_8000h base + 2Ch offset = 800F_802Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:6]										VLAN_OUT_ MODE_2		VLAN_OUT_ MODE_1		VLAN_OUT_ MODE_0	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_OUT_MODE field descriptions

Field	Description
31–6 RSRVD0	Reserved bits. Write as 0.
5–4 VLAN_OUT_ MODE_2	port 2.
3–2 VLAN_OUT_ MODE_1	port 1.
1–0 VLAN_OUT_ MODE_0	port 0.

29.9.12 ENET SWI Enable the input processing according to the VLAN_IN_MODE for a port (HW_ENET_SWI_VLAN_IN_MODE_ENA)

Enable the input processing according to the VLAN_IN_MODE for a port (1 bit per port).

When disabled (bit=0) the VLAN_IN_MODE setting for that port has no effect and the frames are processed unmodified.

Address: HW_ENET_SWI_VLAN_IN_MODE_ENA – 800F_8000h base + 30h offset = 800F_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													VLAN_IN_ MODE_ENA_ 2	VLAN_IN_ MODE_ENA_ 1	VLAN_IN_ MODE_ENA_ 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_IN_MODE_ENA field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 VLAN_IN_ MODE_ENA_2	port 2.
1 VLAN_IN_ MODE_ENA_1	port 1.
0 VLAN_IN_ MODE_ENA_0	port 0.

29.9.13 ENET SWI The VLAN type field value to expect to identify a VLAN tagged frame. (HW_ENET_SWI_VLAN_TAG_ID)

The VLAN type field value to expect to identify a VLAN tagged frame. A valid 802.1Q VLAN tagged frame must use the value 0x8100.

Address: HW_ENET_SWI_VLAN_TAG_ID – 800F_8000h base + 34h offset = 800F_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																SWI_VLAN_TAG_ID															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_TAG_ID field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 SWI_VLAN_ TAG_ID	Defaults to 0x8100 upon reset.

29.9.14 ENET SWI Port Mirroring configuration. (HW_ENET_SWI_MIRROR_CONTROL)

Define mirror port and filtering conditions.

Address: HW_ENET_SWI_MIRROR_CONTROL – 800F_8000h base + 40h offset = 800F_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:11]					EG_DA_MATCH	EG_SA_MATCH	ING_DA_MATCH	ING_SA_MATCH	EG_MAP_ENABLE	ING_MAP_ENABLE	MIRROR_ENABLE	PORTX			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_CONTROL field descriptions

Field	Description
31–11 RSRVD0	Reserved bits. Write as 0.
10 EG_DA_MATCH	If set, only frames transmitted on an egress port with a destination address matching the value programmed in register MIRROR_EDST are mirrored. Other frames are not mirrored. Note: if the egress map is not enabled (eg_map_enable=0) then any frame with a matching destination address will be mirrored.
9 EG_SA_MATCH	If set, only frames transmitted on an egress port with a source address matching the value programmed in register MIRROR_ESRC are mirrored. Other frames are not mirrored. Note: if the egress map is not enabled (eg_map_enable=0) then any frame with a matching source address will be mirrored.
8 ING_DA_MATCH	If set, only frames received on an ingress port with a destination address matching the value programmed in register MIRROR_IDST are mirrored. Other frames are not mirrored. Note: if the ingress map is not enabled (ing_map_enable=0) then any frame with a matching destination address will be mirrored.
7 ING_SA_MATCH	If set, only frames received on an ingress port with a source address matching the value programmed in register MIRROR_ISRC are mirrored. Other frames are not mirrored. Note: if the ingress map is not enabled (ing_map_enable=0) then any frame with a matching source address will be mirrored.
6 EG_MAP_ ENABLE	If set the egress map is enabled (MIRROR_EG_MAP). A frame forwarded to an output port that has a bit set in the egress map will be mirrored. If cleared, the egress port map has no effect.
5 ING_MAP_ ENABLE	If set the ingress map is enabled (MIRROR_ING_MAP). A frame received on an ingress port that has a bit set in the ingress map will be mirrored. If cleared, the ingress port map has no effect.

HW_ENET_SWI_MIRROR_CONTROL field descriptions (continued)

Field	Description
4 MIRROR_ ENABLE	Enable (1) / Disable (0) mirroring.
3–0 PORTX	The port number of the port that should act as the mirror port and receive all mirrored frames.

29.9.15 ENET SWI Port Mirroring Egress port definitions. (HW_ENET_SWI_MIRROR_EG_MAP)

1 Bit per Port. If enabled (Bit=1) frames destined to the port(s) are mirrored to the mirror port

Address: HW_ENET_SWI_MIRROR_EG_MAP – 800F_8000h base + 44h offset = 800F_8044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													MIRROR_EG_MAP_2	MIRROR_EG_MAP_1	MIRROR_EG_MAP_0
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_EG_MAP field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 MIRROR_EG_ MAP_2	port 2.
1 MIRROR_EG_ MAP_1	port 1.
0 MIRROR_EG_ MAP_0	port 0.

29.9.16 ENET SWI Port Mirroring Ingress port definitions. (HW_ENET_SWI_MIRROR_ING_MAP)

1 Bit per Port. If enabled (Bit=1) frames from the port(s) are mirrored on the mirror port

Address: HW_ENET_SWI_MIRROR_ING_MAP – 800F_8000h base + 48h offset = 800F_8048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													MIRROR_ING_MAP_2	MIRROR_ING_MAP_1	MIRROR_ING_MAP_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_ING_MAP field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 MIRROR_ING_MAP_2	port 2.
1 MIRROR_ING_MAP_1	port 1.
0 MIRROR_ING_MAP_0	port 0.

29.9.17 ENET SWI Ingress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ISRC_0)

Address: HW_ENET_SWI_MIRROR_ISRC_0 – 800F_8000h base + 4Ch offset = 800F_804Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MIRROR_ISRC_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_ISRC_0 field descriptions

Field	Description
31–0 MIRROR_ISRC_0	First 4 bytes of address. First byte of MAC address is 7:0, .. , 4th byte is 31:24.

29.9.18 ENET SWI Ingress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ISRC_1)

Address: HW_ENET_SWI_MIRROR_ISRC_1 – 800F_8000h base + 50h offset = 800F_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																MIRROR_ISRC_1															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_ISRC_1 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 MIRROR_ISRC_1	Last 2 bytes of address. 5th Byte in 7:0 and 6th byte in 15:8

29.9.19 ENET SWI Ingress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_IDST_0)

Address: HW_ENET_SWI_MIRROR_IDST_0 – 800F_8000h base + 54h offset = 800F_8054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MIRROR_IDST_0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIRROR_IDST_0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_IDST_0 field descriptions

Field	Description
31–0 MIRROR_IDST_0	First 4 bytes of address (as ISRC_0).

29.9.20 ENET SWI Ingress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_IDST_1)

Address: HW_ENET_SWI_MIRROR_IDST_1 – 800F_8000h base + 58h offset = 800F_8058h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIRROR_IDST_1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_IDST_1 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 MIRROR_IDST_1	Last 2 bytes of address (as ISRC_1).

29.9.21 ENET SWI Egress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ESRC_0)

Address: HW_ENET_SWI_MIRROR_ESRC_0 – 800F_8000h base + 5Ch offset = 800F_805Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MIRROR_ESRC_0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_ESRC_0 field descriptions

Field	Description
31–0 MIRROR_ESRC_0	First 4 bytes of address (as ISRC_0).

29.9.22 ENET SWI Egress Source MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_ESRC_1)

Address: HW_ENET_SWI_MIRROR_ESRC_1 – 800F_8000h base + 60h offset = 800F_8060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																MIRROR_ESRC_1															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_ESRC_1 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 MIRROR_ESRC_1	Last 2 bytes of address (as ISRC_1).

29.9.23 ENET SWI Egress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_EDST_0)

Address: HW_ENET_SWI_MIRROR_EDST_0 – 800F_8000h base + 64h offset = 800F_8064h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MIRROR_ESRC_0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_EDST_0 field descriptions

Field	Description
31–0 MIRROR_ESRC_ 0	First 4 bytes of address (as ISRC_0).

29.9.24 ENET SWI Egress Destination MAC Address for Mirror filtering. (HW_ENET_SWI_MIRROR_EDST_1)

Address: HW_ENET_SWI_MIRROR_EDST_1 – 800F_8000h base + 68h offset = 800F_8068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																MIRROR_ESRC_1															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_EDST_1 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 MIRROR_ESRC_ 1	Last 2 bytes of address (as ISRC_1).

29.9.25 ENET SWI Count Value for Mirror filtering. (HW_ENET_SWI_MIRROR_CNT)

Every nth frame is forwarded to the mirror port if enabled. A value of 0 or 1 means every frame. Valid values are 0..255. Note: If the egress filtering port map is active, every forwarded frame is considered. Otherwise, frames are counted only if the mirroring decision indicated that the frame should be mirrored.

Address: HW_ENET_SWI_MIRROR_CNT – 800F_8000h base + 6Ch offset = 800F_806Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								MIRROR_CNT							
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_MIRROR_CNT field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 MIRROR_CNT	ENET SWI Count Value for Mirror filtering.

29.9.26 ENET SWI Memory Manager Status. (HW_ENET_SWI_OQMGR_STATUS)

All latched bits are cleared upon a write with any content to the register.

Address: HW_ENET_SWI_OQMGR_STATUS – 800F_8000h base + 80h offset = 800F_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2								CELLS_AVAILABLE							
W	▨															
Reset	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1									DEQUEUE_GRANT	RSRVD0		MEM_FULL_LATCH	MEM_FULL	NO_CELL_LATCH	BUSY_INITIALIZING
W	▨	▨	▨	▨	▨	▨	▨	▨	▨		▨	▨				
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0

HW_ENET_SWI_OQMGR_STATUS field descriptions

Field	Description
31–24 RSRVD2	Reserved bits. Write as 0.
23–16 CELLS_ AVAILABLE	Real-time indication of currently available cells in memory.
15–7 RSRVD1	Reserved bits. Write as 0.
6 DEQUEUE_ GRANT	Indication of if currently inputs are de-queued. Should be set always and is cleared when the memory becomes full (see also mem_full indication). Note: the bit is cleared upon reset, however set shortly after when the memory manager has completed initialization.
5–4 RSRVD0	Reserved bits. Write as 0.
3 MEM_FULL_ LATCH	Latched version of mem_full. Is kept set even when mem_full is cleared again. The bit is cleared when the register is written.
2 MEM_FULL	Current memory full indication. The memory is full when less than the programmed minimum cell threshold is available in memory. This is not an error and the memory controller is working fine. It just indicates that the switch does no longer serve its input ports to avoid memory overrun (no_cell error).
1 NO_CELL_ LATCH	Set, when memory has exceeded the maximum available number of cells. The event is latched and the bit stays set if the event is no longer active. This is a fatal error and must never happen during operation. The minimum cells threshold must be increased if it happens. The bit is always set after reset (during initialization) and must be cleared when the busy initialization (see bit 0) indication is cleared. IMPORTANT NOTE: When this bit is set any time during operation (after initialization completed) the switch is in an inoperable state and must be reset completely to restore correct operation. If such an event happens, the QMGR_MINCELLS setting must be increased during initialization to avoid such situation. The bit is cleared when the register is written.
0 BUSY_ INITIALIZING	When set (1), Memory controller is initializing. The initialization is only preparing the internal data structures within the controller, it does not initialize the shared memory used for frame storage as this is not required. It is asserted after reset and stays set until the memory controller is ready to store frames. The switch must not be enabled before initialization of the memory controller has been completed.

29.9.27 ENET SWI Low Memory threshold. (HW_ENET_SWI_QMGR_MINCELLS)

If less than the programmed number of cells is available in the memory, the switch will discard frames. The value should be chosen to give enough margin of at least 2 full sized frames. A memory overflow due to a too low threshold is a fatal error and may require a device reset.

Address: HW_ENET_SWI_QMGR_MINCELLS – 800F_8000h base + 84h offset = 800F_8084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								QMGR_MINCELLS							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

HW_ENET_SWI_QMGR_MINCELLS field descriptions

Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 QMGR_ MINCELLS	Default: 9 = (2,3Kbyte).

29.9.28 ENET SWI Statistic providing the lowest number of free cells reached in memory (HW_ENET_SWI_QMGR_ST_MINCELLS)

Statistic providing the lowest number of free cells reached in memory during operation since this statistics was last cleared. The value is reset to the maximum when a write to the register with any value is performed.

Address: HW_ENET_SWI_QMGR_ST_MINCELLS – 800F_8000h base + 88h offset = 800F_8088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	QMGR_ST_MINCELLS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_QMGR_ST_MINCELLS field descriptions

Field	Description
31–0 QMGR_ST_ MINCELLS	ENET SWI Statistic providing the lowest number of free cells reached in memory

29.9.29 ENET SWI Port Congestion status (internal). (HW_ENET_SWI_QMGR_CONGEST_STAT)

One bit per port. If any queue of a port is congested the bit is set.

Address: HW_ENET_SWI_QMGR_CONGEST_STAT – 800F_8000h base + 8Ch offset = 800F_808Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:3]													QMGR_CONGEST_STAT_2	QMGR_CONGEST_STAT_1	QMGR_CONGEST_STAT_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_QMGR_CONGEST_STAT field descriptions

Field	Description
31–3 RSRVD0	Reserved bits. Write as 0.
2 QMGR_ CONGEST_ STAT_2	port 2.

HW_ENET_SWI_QMGR_CONGEST_STAT field descriptions (continued)

Field	Description
1 QMGR_ CONGEST_ STAT_1	port 1.
0 QMGR_ CONGEST_ STAT_0	port 0.

29.9.30 ENET SWI Switch input and output interface status (internal). (HW_ENET_SWI_QMGR_IFACE_STAT)

Address: HW_ENET_SWI_QMGR_IFACE_STAT – 800F_8000h base + 90h offset = 800F_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1													INPUT_2	INPUT_1	INPUT_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0													OUTPUT_2	OUTPUT_1	OUTPUT_0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

HW_ENET_SWI_QMGR_IFACE_STAT field descriptions

Field	Description
31–19 RSRVD1	Reserved bits. Write as 0.
18 INPUT_2	input data available on port 2.
17 INPUT_1	input data available on port 1.

HW_ENET_SWI_QMGR_IFACE_STAT field descriptions (continued)

Field	Description
16 INPUT_0	input data available on port 0.
15–3 RSRVD0	Reserved bits. Write as 0.
2 OUTPUT_2	output ready to accept data on port 2.
1 OUTPUT_1	output ready to accept data on port 1.
0 OUTPUT_0	output ready to accept data on port 0.

29.9.31 ENET SWI Queue weights for each queue. (HW_ENET_SWI_QM_WEIGHTS)

Defines the weight for the corresponding queue for all ports. A higher weight gives more priority to the queue. Valid values are between 0 and 30 for each queue setting. Setting all weights to 0 implements a strict priority scheme. Queue 3 is the highest priority queue.

Address: HW_ENET_SWI_QM_WEIGHTS – 800F_8000h base + 94h offset = 800F_8094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3			QUEUE_3					RSRVD2			QUEUE_2					RSRVD1			QUEUE_1					RSRVD0			QUEUE_0				
W	▨	▨	▨						▨	▨	▨						▨	▨	▨						▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_QM_WEIGHTS field descriptions

Field	Description
31–29 RSRVD3	Reserved bits. Write as 0.
28–24 QUEUE_3	Queue 3
23–21 RSRVD2	Reserved bits. Write as 0.
20–16 QUEUE_2	Queue 2
15–13 RSRVD1	Reserved bits. Write as 0.
12–8 QUEUE_1	Queue 1
7–5 RSRVD0	Reserved bits. Write as 0.

HW_ENET_SWI_QM_WEIGHTS field descriptions (continued)

Field	Description
4–0 QUEUE_0	Queue 0

29.9.32 ENET SWI Define congestion threshold for Port0 backpressure. (HW_ENET_SWI_QMGR_MINCELLSP0)

If the total output queues shared memory has less than this amount of free cells available, the switch stops serving the port0 input buffer. This will eventually fill up the input buffer deasserting ff_tx_rdy0. A value of 0 disables the function (no backpressure).

Address: HW_ENET_SWI_QMGR_MINCELLSP0 – 800F_8000h base + 9Ch offset = 800F_809Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																								QMGR_MINCELLSP0							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1

HW_ENET_SWI_QMGR_MINCELLSP0 field descriptions

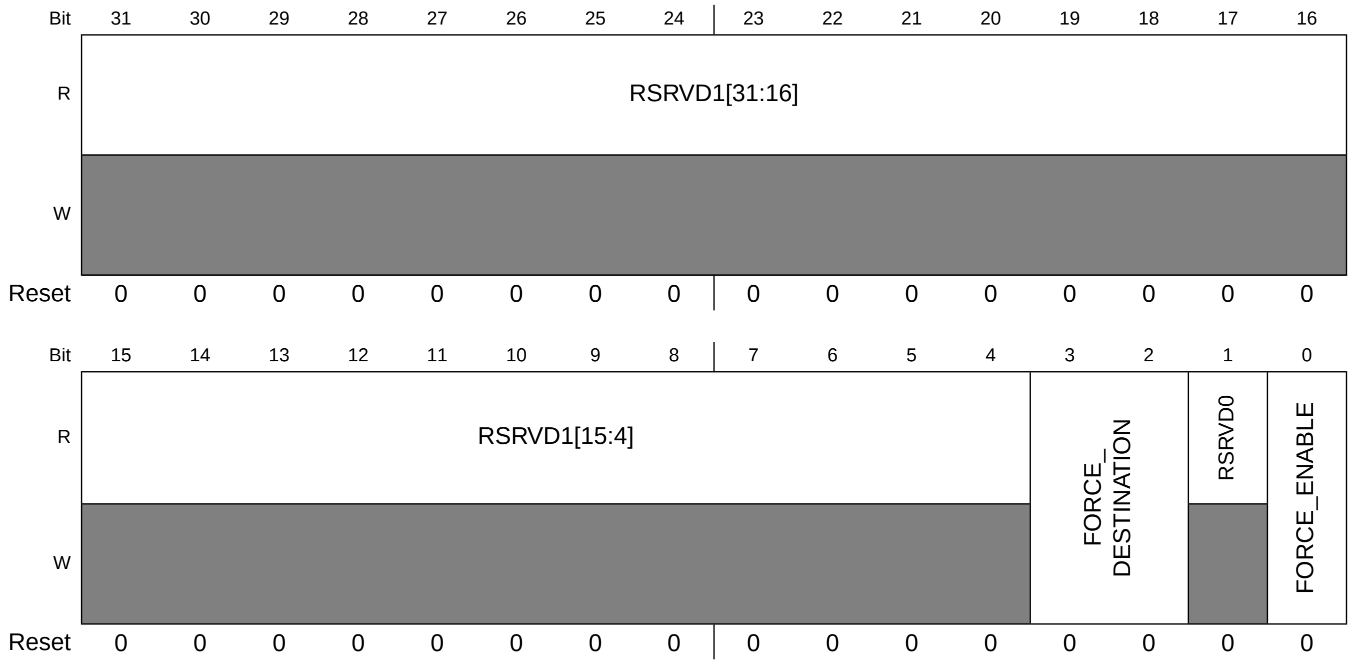
Field	Description
31–8 RSRVD0	Reserved bits. Write as 0.
7–0 QMGR_ MINCELLSP0	Mininum umber of free cells required.

29.9.33 ENET SWI Enable forced forwarding for a frame processed from port 0 (HW_ENET_SWI_FORCE_FWD_P0)

Enable forced forwarding for a frame processed from port 0. The forced forwarding, if enabled, directs a frame from the processor to output port(s) ignoring any lookup resolution.

Forced forwarding for Port 0 frames (i.e. frames transmitted from the DMA0 to the port 0 of the switch).

Address: HW_ENET_SWI_FORCE_FWD_P0 – 800F_8000h base + BCh offset = 800F_80BCh



HW_ENET_SWI_FORCE_FWD_P0 field descriptions

Field	Description
31–4 RSRVD1	Reserved bits. Write as 0.
3–2 FORCE_ DESTINATION	When force is enabled, the bits define if the frame should be forwarded to the MAC at port 1 (bit2=1) and/or the MAC at port 2 (bit(3=1)). Note, it is possible to forward to one or both MAC ports. If none of the bits is set, the force enable is ignored and the frame is processed normally. This can be used to implement a handshake, as the force enable bit will still be reset but no further action will happen.
1 RSRVD0	Reserved bits. Write as 0.
0 FORCE_ENABLE	When set (1) the next frame received from port 0 (the local DMA port) will be forwarded to the ports defined in the force destination port map. The bit is reset to 0 automatically when one frame from port 0 has been processed by the switch (i.e. has been read from the port 0 input buffer;). Therefore the bit must be set again as necessary.

29.9.34 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP1)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP1 – 800F_8000h base + C0h offset = 800F_80C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DESTINATION_PORT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_ SOURCE	COMPARE_ DEST	MODE		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP1 field descriptions

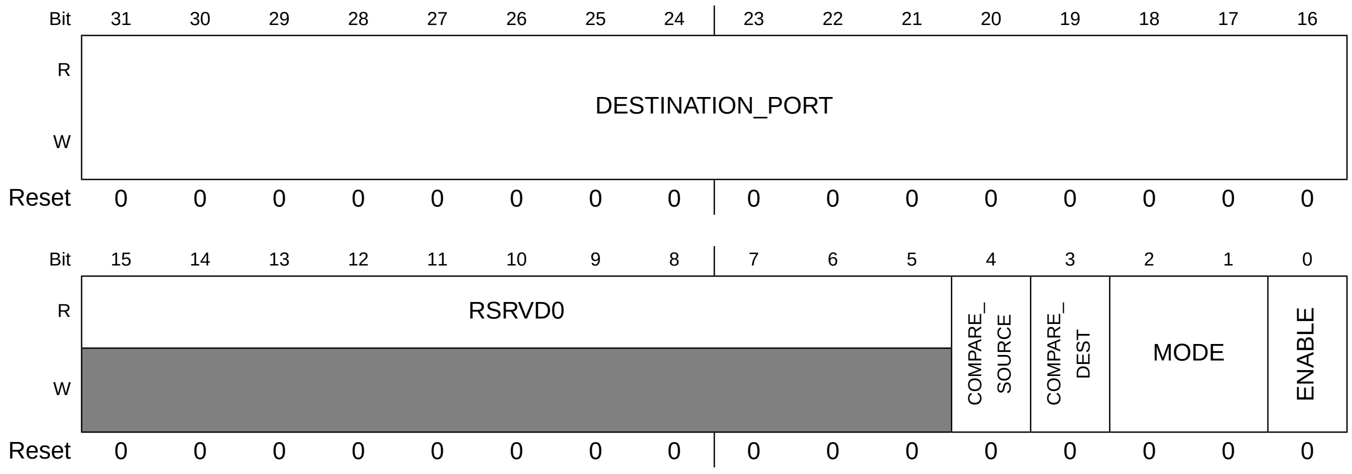
Field	Description
31–16 DESTINATION_ PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_ SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.35 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP2)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP2 – 800F_8000h base + C4h offset = 800F_80C4h



HW_ENET_SWI_PORTSNOOP2 field descriptions

Field	Description
31–16 DESTINATION_PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.

HW_ENET_SWI_PORTSNOOP2 field descriptions (continued)

Field	Description
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.36 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP3)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP3 – 800F_8000h base + C8h offset = 800F_80C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DESTINATION_PORT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_ SOURCE	COMPARE_ DEST	MODE		ENABLE
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP3 field descriptions

Field	Description
31–16 DESTINATION_ PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_ SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.

HW_ENET_SWI_PORTSNOOP3 field descriptions (continued)

Field	Description
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.37 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP4)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP4 – 800F_8000h base + CCh offset = 800F_80CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DESTINATION_PORT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_ SOURCE	COMPARE_ DEST	MODE		ENABLE
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP4 field descriptions

Field	Description
31–16 DESTINATION_ PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_ SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.38 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP5)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP5 – 800F_8000h base + D0h offset = 800F_80D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DESTINATION_PORT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_ SOURCE	COMPARE_ DEST	MODE		ENABLE
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP5 field descriptions

Field	Description
31–16 DESTINATION_ PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_ SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.39 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP6)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP6 – 800F_8000h base + D4h offset = 800F_80D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DESTINATION_PORT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_ SOURCE	COMPARE_ DEST	MODE		ENABLE
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP6 field descriptions

Field	Description
31–16 DESTINATION_ PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_ SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.40 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP7)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP7 – 800F_8000h base + D8h offset = 800F_80D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	DESTINATION_PORT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_SOURCE	COMPARE_DEST	MODE		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP7 field descriptions

Field	Description
31–16 DESTINATION_PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.
3 COMPARE_DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.

HW_ENET_SWI_PORTSNOOP7 field descriptions (continued)

Field	Description
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.41 ENET SWI Port Snooping function. Eight independent entries are available. (HW_ENET_SWI_PORTSNOOP8)

TCP/UDP Port number Snooping function configuration.

Note: it is possible to set both the compare source/dest bits. The result is OR'ed, meaning if any of the fields match the compare value, the frame is processed as defined by the mode bits.

Address: HW_ENET_SWI_PORTSNOOP8 – 800F_8000h base + DCh offset = 800F_80DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DESTINATION_PORT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0											COMPARE_SOURCE	COMPARE_DEST	MODE		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PORTSNOOP8 field descriptions

Field	Description
31–16 DESTINATION_PORT	The 16-bit port number to compare within the TCP or UDP header of a frame.
15–5 RSRVD0	Reserved bits. Write as 0.
4 COMPARE_SOURCE	When set, the TCP or UDP source port number field within the frame is compared with the compare value provided in 31:16.

HW_ENET_SWI_PORTSNOOP8 field descriptions (continued)

Field	Description
3 COMPARE_ DEST	When set, the TCP or UDP destination port number field within the frame is compared with the compare value provided in 31:16.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the TCP/UDP destination port value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the snooping function only if the enable bit is set. Otherwise the settings are ignored. When written with 0 will also force bits 3,4 to 0. Defaults to 0 upon reset.

29.9.42 ENET SWI IP Snooping function1 (HW_ENET_SWI_IPSNOOP1)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP1 – 800F_8000h base + E0h offset = 800F_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W	PROTOCOL								▨					MODE		ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP1 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP1 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.43 ENET SWI IP Snooping function2 (HW_ENET_SWI_IPSNOOP2)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP2 – 800F_8000h base + E4h offset = 800F_80E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP2 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP2 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.44 ENET SWI IP Snooping function3 (HW_ENET_SWI_IPSNOOP3)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP3 – 800F_8000h base + E8h offset = 800F_80E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP3 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

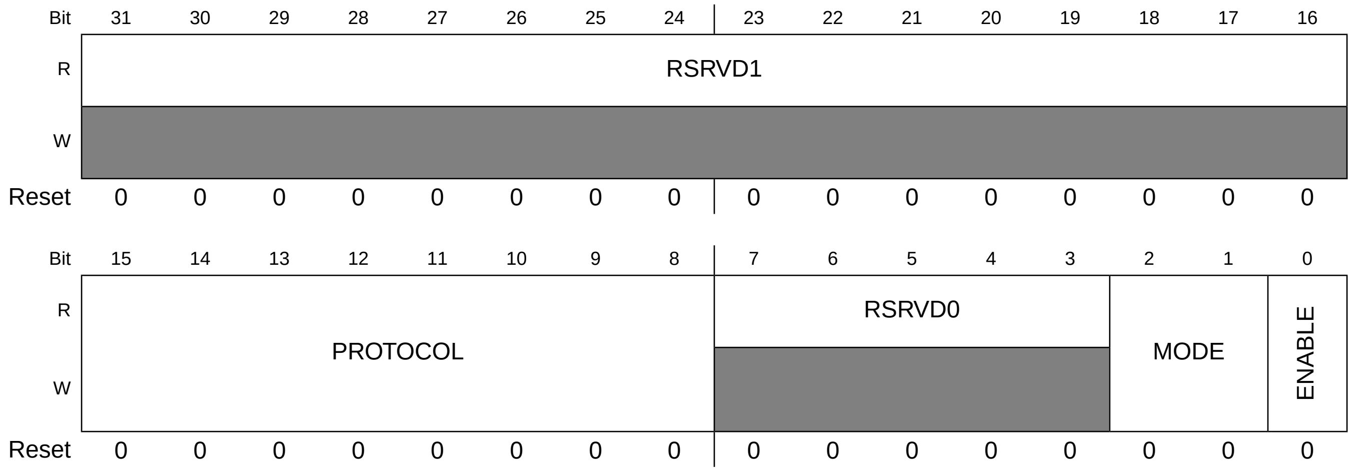
HW_ENET_SWI_IPSNOOP3 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.45 ENET SWI IP Snooping function4 (HW_ENET_SWI_IPSNOOP4)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP4 – 800F_8000h base + ECh offset = 800F_80ECh



HW_ENET_SWI_IPSNOOP4 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP4 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.46 ENET SWI IP Snooping function5 (HW_ENET_SWI_IPSNOOP5)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP5 – 800F_8000h base + F0h offset = 800F_80F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP5 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

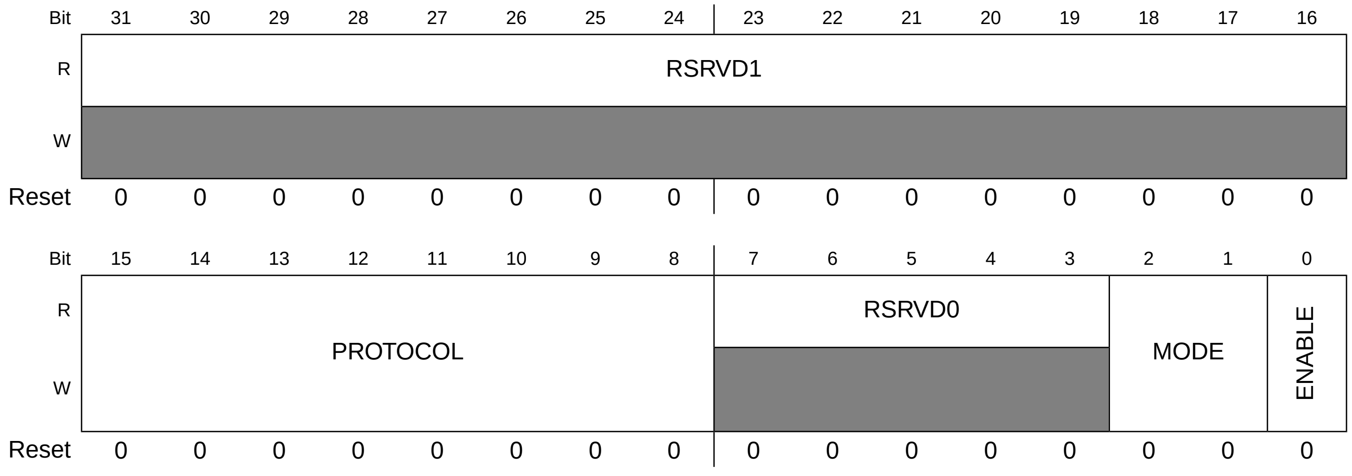
HW_ENET_SWI_IPSNOOP5 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.47 ENET SWI IP Snooping function6 (HW_ENET_SWI_IPSNOOP6)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP6 – 800F_8000h base + F4h offset = 800F_80F4h



HW_ENET_SWI_IPSNOOP6 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP6 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.48 ENET SWI IP Snooping function7 (HW_ENET_SWI_IPSNOOP7)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP7 – 800F_8000h base + F8h offset = 800F_80F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP7 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP7 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1:IP Snooping function 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.49 ENET SWI IP Snooping function8 (HW_ENET_SWI_IPSNOOP8)

IP Snooping function. Eight independent snooping entries are available.

Address: HW_ENET_SWI_IPSNOOP8 – 800F_8000h base + FCh offset = 800F_80FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PROTOCOL								RSRVD0					MODE		ENABLE
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IPSNOOP8 field descriptions

Field	Description
31–16 RSRVD1	Reserved bits. Write as 0.

HW_ENET_SWI_IPSNOOP8 field descriptions (continued)

Field	Description
15–8 PROTOCOL	The 8-bit protocol value to match with the incoming frame's IP header protocol field.
7–3 RSRVD0	Reserved bits. Write as 0.
2–1 MODE	Defines the forwarding that should occur, when an IP frame is received and the protocol field matches the protocol value (see bits 15:8). Bits 2:1: 00: forward frame to designated management port only 01: copy to management port and forward normally 10: discard Note: the management port is defined in register MGMT_CONFIG.
0 ENABLE	When set (1) the entry contains valid data and the function is active. If a match with the protocol value occurs, the frame is processed as defined by the mode setting. All other bits of the register are interpreted by the IP snooping function only if the enable bit is set. Otherwise the settings are ignored. Defaults to 0 upon reset.

29.9.50 ENET SWI Port 0 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY0)

Port 0 VLAN priority resolution map. The 3-bit to 3-bit priority map addressed by the priority bits of the VLAN tag

The VLAN_PRIORITY0 registers implement a 3-bit to 3-bit VLAN priority mapping capability. The current frame's 3-bit VLAN priority field is used as an index and the corresponding priority is taken from the respective position of the register giving the final classification for the frame.

Address: HW_ENET_SWI_VLAN_PRIORITY0 – 800F_8000h base + 100h offset = 800F_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0								P7			P6			P5			P4			P3			P2			P1			P0		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_PRIORITY0 field descriptions

Field	Description
31–24 RSRVD0	Reserved bits. Write as 0.

HW_ENET_SWI_VLAN_PRIORITY0 field descriptions (continued)

Field	Description
23–21 P7	Priority for input priority 7
20–18 P6	Priority for input priority 6
17–15 P5	Priority for input priority 5
14–12 P4	Priority for input priority 4
11–9 P3	Priority for input priority 3
8–6 P2	Priority for input priority 2
5–3 P1	Priority for input priority 1
2–0 P0	Priority for input priority 0

29.9.51 ENET SWI Port 1 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY1)

Port 1 VLAN priority resolution map. The 3-bit to 3-bit priority map addressed by the priority bits of the VLAN tag

The VLAN_PRIORITY1 registers implement a 3-bit to 3-bit VLAN priority mapping capability. The current frame's 3-bit VLAN priority field is used as an index and the corresponding priority is taken from the respective position of the register giving the final classification for the frame.

Address: HW_ENET_SWI_VLAN_PRIORITY1 – 800F_8000h base + 104h offset = 800F_8104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0								P7			P6			P5			P4			P3			P2			P1			P0		
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_PRIORITY1 field descriptions

Field	Description
31–24 RSRVD0	Reserved bits. Write as 0.

HW_ENET_SWI_VLAN_PRIORITY1 field descriptions (continued)

Field	Description
23–21 P7	Priority for input priority 7
20–18 P6	Priority for input priority 6
17–15 P5	Priority for input priority 5
14–12 P4	Priority for input priority 4
11–9 P3	Priority for input priority 3
8–6 P2	Priority for input priority 2
5–3 P1	Priority for input priority 1
2–0 P0	Priority for input priority 0

29.9.52 ENET SWI Port 2 VLAN priority resolution map (HW_ENET_SWI_VLAN_PRIORITY2)

Port 2 VLAN priority resolution map. The 3-bit to 3-bit priority map addressed by the priority bits of the VLAN tag

The VLAN_PRIORITY2 registers implement a 3-bit to 3-bit VLAN priority mapping capability. The current frame's 3-bit VLAN priority field is used as an index and the corresponding priority is taken from the respective position of the register giving the final classification for the frame.

Address: HW_ENET_SWI_VLAN_PRIORITY2 – 800F_8000h base + 108h offset = 800F_8108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0								P7			P6			P5			P4			P3			P2			P1			P0		
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_PRIORITY2 field descriptions

Field	Description
31–24 RSRVD0	Reserved bits. Write as 0.

HW_ENET_SWI_VLAN_PRIORITY2 field descriptions (continued)

Field	Description
23–21 P7	Priority for input priority 7
20–18 P6	Priority for input priority 6
17–15 P5	Priority for input priority 5
14–12 P4	Priority for input priority 4
11–9 P3	Priority for input priority 3
8–6 P2	Priority for input priority 2
5–3 P1	Priority for input priority 1
2–0 P0	Priority for input priority 0

29.9.53 ENET SWI IPv4 and IPv6 priority resolution table programming (HW_ENET_SWI_IP_PRIORITY)

IPv4 and IPv6 priority resolution table programming. One Register per port

Address: HW_ENET_SWI_IP_PRIORITY – 800F_8000h base + 140h offset = 800F_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	READ	RSRVD0[30:16]														
W		▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:15]	PRIORITY_PORT2		PRIORITY_PORT1		PRIORITY_PORT0		IPV4_SELECT	ADDRESS							
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IP_PRIORITY field descriptions

Field	Description
31 READ	Must be cleared to write values in the tables.

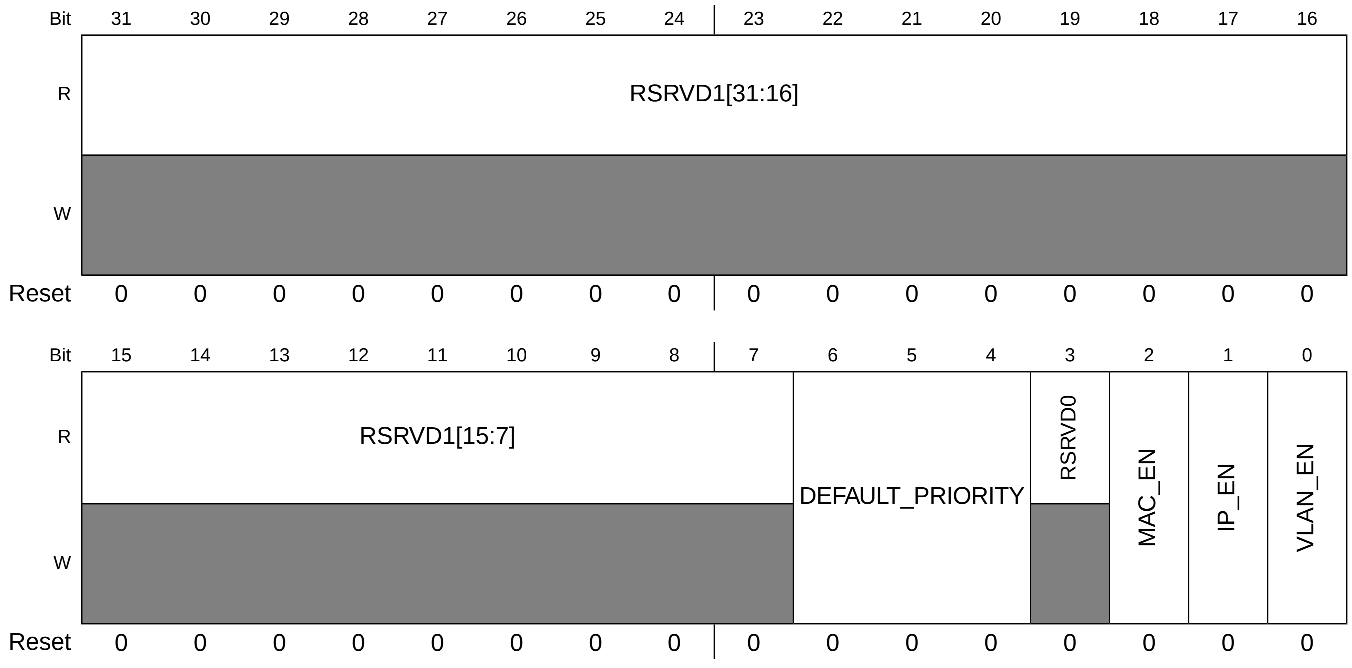
HW_ENET_SWI_IP_PRIORITY field descriptions (continued)

Field	Description
	When set during register writes, the IPv6 select and address bits are stored in the register only and the priority bits are ignored and not written into the addressed table. When the register is read, the priority bits represent the value read from the table always.
30–15 RSRVD0	Reserved bits. Write as 0.
14–13 PRIORITY_ PORT2	The priority information to write into the addressed table entry. These 2 bits represent the output priority selected when the frame is received on port 2. When reading from the register, the bits show the value from the addressed table entry (address from last write operation).
12–11 PRIORITY_ PORT1	The priority information to write into the addressed table entry. These 2 bits represent the output priority selected when the frame is received on port 1. When reading from the register, the bits show the value from the addressed table entry (address from last write operation).
10–9 PRIORITY_ PORT0	The priority information to write into the addressed table entry. These 2 bits represent the output priority selected when the frame is received on port 0. 00=priority 0 (will be forwarded to output queue 0) 01=priority 1 (output queue 1) 10=priority 2 (output queue 2) 11=priority 3 (output queue 3) When reading from the register, the bits show the value from the addressed table entry (address from last write operation).
8 IPV4_SELECT	If set during a write, the IPv4 table is accessed. Valid address values range from 0 to 63. If cleared, the IPv6 table is accessed. Valid address values range from 0 to 255.
7–0 ADDRESS	The address of the priority entry to read or write for a frame received on port n. The IPv4 priority table has 64 entries. The IPv6 table has 256 entries.

29.9.54 ENET SWI Port 0 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG0)

Port 0 Priority resolution configuration. Defines which priority information should be used for priority resolution

Address: HW_ENET_SWI_PRIORITY_CFG0 – 800F_8000h base + 180h offset = 800F_8180h



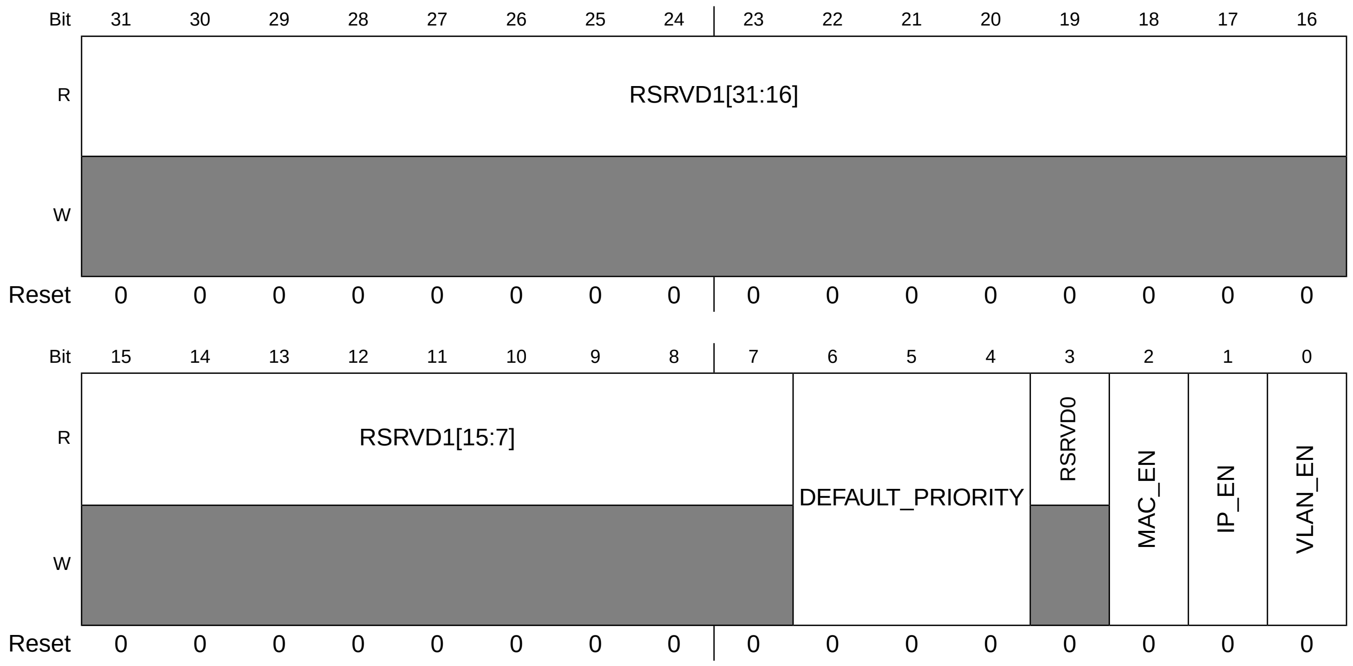
HW_ENET_SWI_PRIORITY_CFG0 field descriptions

Field	Description
31–7 RSRVD1	Reserved bits. Write as 0.
6–4 DEFAULT_ PRIORITY	The default priority of a frame received on port 0, if none of the priority resolutions could define a priority of the frame. Up to 3 bits can be implemented.
3 RSRVD0	Reserved bits. Write as 0.
2 MAC_EN	Enable MAC based priority resolution for frame received on port n. If set, the priority information found within the MAC address table is used.
1 IP_EN	Enable IP priority resolution for frame received on port n. If set, the IP DiffServ/COS field is used and priority is resolved according to the IP_PRIORITYn setting for the port. If cleared, IP Diffserv/COS fields are ignored.
0 VLAN_EN	Enable VLAN priority resolution for frame received on port n. If set, the VLAN tag field of a frame is inspected and priority is resolved according to the setting programmed in VLAN_PRIORITYn for the port on which the frame was received. If cleared, VLAN priority is ignored.

29.9.55 ENET SWI Port 1 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG1)

Port 1 Priority resolution configuration. Defines which priority information should be used for priority resolution

Address: HW_ENET_SWI_PRIORITY_CFG1 – 800F_8000h base + 184h offset = 800F_8184h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD1[31:16]
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD1[15:7]
DEFAULT_PRIORITY
RSRVD0
MAC_EN
IP_EN
VLAN_EN
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_ENET_SWI_PRIORITY_CFG1 field descriptions

Field	Description
31–7 RSRVD1	Reserved bits. Write as 0.
6–4 DEFAULT_ PRIORITY	The default priority of a frame received on port 1, if none of the priority resolutions could define a priority of the frame. Up to 3 bits can be implemented.
3 RSRVD0	Reserved bits. Write as 0.
2 MAC_EN	Enable MAC based priority resolution for frame received on port n. If set, the priority information found within the MAC address table is used.
1 IP_EN	Enable IP priority resolution for frame received on port n. If set, the IP DiffServ/COS field is used and priority is resolved according to the IP_PRIORITYn setting for the port. If cleared, IP Diffserv/COS fields are ignored.
0 VLAN_EN	Enable VLAN priority resolution for frame received on port n.

HW_ENET_SWI_PRIORITY_CFG1 field descriptions (continued)

Field	Description
	If set, the VLAN tag field of a frame is inspected and priority is resolved according to the setting programmed in VLAN_PRIORITYn for the port on which the frame was received. If cleared, VLAN priority is ignored.

29.9.56 ENET SWI Port 2 Priority resolution configuration (HW_ENET_SWI_PRIORITY_CFG2)

Port 2 Priority resolution configuration. Defines which priority information should be used for priority resolution

Address: HW_ENET_SWI_PRIORITY_CFG2 – 800F_8000h base + 188h offset = 800F_8188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:7]									DEFAULT_PRIORITY			RSRVD0	MAC_EN	IP_EN	VLAN_EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_PRIORITY_CFG2 field descriptions

Field	Description
31–7 RSRVD1	Reserved bits. Write as 0.
6–4 DEFAULT_ PRIORITY	The default priority of a frame received on port 2, if none of the priority resolutions could define a priority of the frame. Up to 3 bits can be implemented.
3 RSRVD0	Reserved bits. Write as 0.
2 MAC_EN	Enable MAC based priority resolution for frame received on port n.

HW_ENET_SWI_PRIORITY_CFG2 field descriptions (continued)

Field	Description
	If set, the priority information found within the MAC address table is used.
1 IP_EN	Enable IP priority resolution for frame received on port n. If set, the IP DiffServ/COS field is used and priority is resolved according to the IP_PRIORITYn setting for the port. If cleared, IP Diffserv/COS fields are ignored.
0 VLAN_EN	Enable VLAN priority resolution for frame received on port n. If set, the VLAN tag field of a frame is inspected and priority is resolved according to the setting programmed in VLAN_PRIORITYn for the port on which the frame was received. If cleared, VLAN priority is ignored.

29.9.57 ENET SWI Port 0 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO0)

Port 0 VLAN-ID field for VLAN input manipulation function of a port if it exists (synthesis option).

Address: HW_ENET_SWI_SYSTEM_TAGINFO0 – 800F_8000h base + 200h offset = 800F_ 8200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																SYSTEM_TAGINFO0															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_SYSTEM_TAGINFO0 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 SYSTEM_ TAGINFO0	VLAN information field.

29.9.58 ENET SWI Port 1 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO1)

Port 1 VLAN-ID field for VLAN input manipulation function of a port if it exists (synthesis option).

Address: HW_ENET_SWI_SYSTEM_TAGINFO1 – 800F_8000h base + 204h offset = 800F_8204h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																SYSTEM_TAGINFO0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_SYSTEM_TAGINFO1 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 SYSTEM_ TAGINFO0	VLAN information field.

29.9.59 ENET SWI Port 2 VLAN-ID field for VLAN input manipulation function (HW_ENET_SWI_SYSTEM_TAGINFO2)

Port 2 VLAN-ID field for VLAN input manipulation function of a port if it exists (synthesis option).

Address: HW_ENET_SWI_SYSTEM_TAGINFO2 – 800F_8000h base + 208h offset = 800F_8208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																SYSTEM_TAGINFO0															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_SYSTEM_TAGINFO2 field descriptions

Field	Description
31–16 RSRVD0	Reserved bits. Write as 0.
15–0 SYSTEM_ TAGINFO0	VLAN information field.

29.9.60 ENET SWI VLAN domain resolution entry 0. (HW_ENET_SWI_VLAN_RES_TABLE_0)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_0 – 800F_8000h base + 280h offset = 800F_8280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_0												PORT_2	PORT_1	PORT_0
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_0 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_0	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.61 ENET SWI VLAN domain resolution entry 1. (HW_ENET_SWI_VLAN_RES_TABLE_1)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_1 – 800F_8000h base + 284h offset = 800F_8284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_1												PORT_2	PORT_1	PORT_0
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_1 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_1	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.62 ENET SWI VLAN domain resolution entry 2. (HW_ENET_SWI_VLAN_RES_TABLE_2)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_2 – 800F_8000h base + 288h offset = 800F_8288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_2												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_2 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_2	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.63 ENET SWI VLAN domain resolution entry 3. (HW_ENET_SWI_VLAN_RES_TABLE_3)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_3 – 800F_8000h base + 28Ch offset = 800F_828Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_3												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_3 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_3	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.64 ENET SWI VLAN domain resolution entry 4. (HW_ENET_SWI_VLAN_RES_TABLE_4)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_4 – 800F_8000h base + 290h offset = 800F_8290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_4												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_4 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_4	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.65 ENET SWI VLAN domain resolution entry 5. (HW_ENET_SWI_VLAN_RES_TABLE_5)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_5 – 800F_8000h base + 294h offset = 800F_8294h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_5												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_5 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_5	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.66 ENET SWI VLAN domain resolution entry 6. (HW_ENET_SWI_VLAN_RES_TABLE_6)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_6 – 800F_8000h base + 298h offset = 800F_8298h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_6												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_6 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_6	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.67 ENET SWI VLAN domain resolution entry 7. (HW_ENET_SWI_VLAN_RES_TABLE_7)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_7 – 800F_8000h base + 29Ch offset = 800F_829Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_7												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_7 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_7	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.68 ENET SWI VLAN domain resolution entry 8. (HW_ENET_SWI_VLAN_RES_TABLE_8)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_8 – 800F_8000h base + 2A0h offset = 800F_82A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_8												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_8 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_8	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.69 ENET SWI VLAN domain resolution entry 9. (HW_ENET_SWI_VLAN_RES_TABLE_9)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_9 – 800F_8000h base + 2A4h offset = 800F_82A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_9												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_9 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_9	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.70 ENET SWI VLAN domain resolution entry 10. (HW_ENET_SWI_VLAN_RES_TABLE_10)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_10 – 800F_8000h base + 2A8h offset = 800F_82A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_10												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_10 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_10	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.71 ENET SWI VLAN domain resolution entry 11. (HW_ENET_SWI_VLAN_RES_TABLE_11)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_11 – 800F_8000h base + 2ACh offset = 800F_82ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_11												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_11 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_11	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.72 ENET SWI VLAN domain resolution entry 12. (HW_ENET_SWI_VLAN_RES_TABLE_12)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_12 – 800F_8000h base + 2B0h offset = 800F_82B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_12												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_12 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_12	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.73 ENET SWI VLAN domain resolution entry 13. (HW_ENET_SWI_VLAN_RES_TABLE_13)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_13 – 800F_8000h base + 2B4h offset = 800F_82B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_13												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_13 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_13	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.74 ENET SWI VLAN domain resolution entry 14. (HW_ENET_SWI_VLAN_RES_TABLE_14)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_14 – 800F_8000h base + 2B8h offset = 800F_82B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_14												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_14 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_14	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.75 ENET SWI VLAN domain resolution entry 15. (HW_ENET_SWI_VLAN_RES_TABLE_15)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_15 – 800F_8000h base + 2BCh offset = 800F_82BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_15												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_15 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_15	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.76 ENET SWI VLAN domain resolution entry 16. (HW_ENET_SWI_VLAN_RES_TABLE_16)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_16 – 800F_8000h base + 2C0h offset = 800F_82C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_16												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

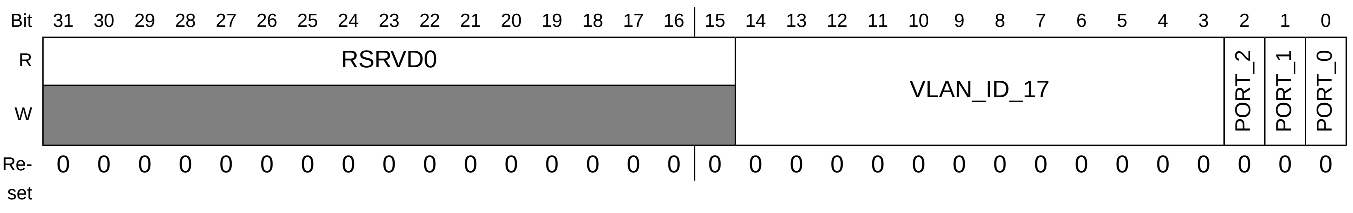
HW_ENET_SWI_VLAN_RES_TABLE_16 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_16	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.77 ENET SWI VLAN domain resolution entry 17. (HW_ENET_SWI_VLAN_RES_TABLE_17)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_17 – 800F_8000h base + 2C4h offset = 800F_82C4h



HW_ENET_SWI_VLAN_RES_TABLE_17 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_17	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.78 ENET SWI VLAN domain resolution entry 18. (HW_ENET_SWI_VLAN_RES_TABLE_18)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_18 – 800F_8000h base + 2C8h offset = 800F_82C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_18												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_18 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_18	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.79 ENET SWI VLAN domain resolution entry 19. (HW_ENET_SWI_VLAN_RES_TABLE_19)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_19 – 800F_8000h base + 2CCh offset = 800F_82CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_19												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_19 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_19	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.80 ENET SWI VLAN domain resolution entry 20. (HW_ENET_SWI_VLAN_RES_TABLE_20)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_20 – 800F_8000h base + 2D0h offset = 800F_82D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_20												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_20 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_20	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.81 ENET SWI VLAN domain resolution entry 21. (HW_ENET_SWI_VLAN_RES_TABLE_21)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_21 – 800F_8000h base + 2D4h offset = 800F_82D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_21												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_21 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_21	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.82 ENET SWI VLAN domain resolution entry 22. (HW_ENET_SWI_VLAN_RES_TABLE_22)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_22 – 800F_8000h base + 2D8h offset = 800F_82D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_22												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_22 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_22	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.83 ENET SWI VLAN domain resolution entry 23. (HW_ENET_SWI_VLAN_RES_TABLE_23)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_23 – 800F_8000h base + 2DCh offset = 800F_82DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_23												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_23 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_23	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.84 ENET SWI VLAN domain resolution entry 24. (HW_ENET_SWI_VLAN_RES_TABLE_24)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_24 – 800F_8000h base + 2E0h offset = 800F_82E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_24												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_24 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_24	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.85 ENET SWI VLAN domain resolution entry 25. (HW_ENET_SWI_VLAN_RES_TABLE_25)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_25 – 800F_8000h base + 2E4h offset = 800F_82E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_25												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_25 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_25	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.86 ENET SWI VLAN domain resolution entry 26. (HW_ENET_SWI_VLAN_RES_TABLE_26)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_26 – 800F_8000h base + 2E8h offset = 800F_82E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_26												PORT_2	PORT_1	PORT_0
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_26 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_26	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.87 ENET SWI VLAN domain resolution entry 27. (HW_ENET_SWI_VLAN_RES_TABLE_27)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_27 – 800F_8000h base + 2ECh offset = 800F_82ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_27												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_27 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_27	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.88 ENET SWI VLAN domain resolution entry 28. (HW_ENET_SWI_VLAN_RES_TABLE_28)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_28 – 800F_8000h base + 2F0h offset = 800F_82F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_28												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_28 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_28	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.89 ENET SWI VLAN domain resolution entry 29. (HW_ENET_SWI_VLAN_RES_TABLE_29)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_29 – 800F_8000h base + 2F4h offset = 800F_82F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_29												PORT_2	PORT_1	PORT_0
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_29 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_29	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.90 ENET SWI VLAN domain resolution entry 30. (HW_ENET_SWI_VLAN_RES_TABLE_30)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_30 – 800F_8000h base + 2F8h offset = 800F_82F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_30												PORT_2	PORT_1	PORT_0
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_30 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_30	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.91 ENET SWI VLAN domain resolution entry 31. (HW_ENET_SWI_VLAN_RES_TABLE_31)

Note: the VLAN table is always searched completely, hence there is no order in which the table entries need to be written.

Address: HW_ENET_SWI_VLAN_RES_TABLE_31 – 800F_8000h base + 2FCh offset = 800F_82FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																	VLAN_ID_31												PORT_2	PORT_1	PORT_0
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_VLAN_RES_TABLE_31 field descriptions

Field	Description
31–15 RSRVD0	Reserved bits. Write as 0.
14–3 VLAN_ID_31	12-bit VLAN identifier 0.
2 PORT_2	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 2.
1 PORT_1	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 1.
0 PORT_0	member of the VLAN identified with the 12-bit VLAN ID of the entry on port 0.

29.9.92 ENET SWI Total number of incoming frames processed (HW_ENET_SWI_TOTAL_DISC)

Total number of incoming frames processed but discarded in the switch.

Address: HW_ENET_SWI_TOTAL_DISC – 800F_8000h base + 300h offset = 800F_8300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOTAL_DISC																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_TOTAL_DISC field descriptions

Field	Description
31–0 TOTAL_DISC	ENET SWI Total number of incoming frames processed

29.9.93 ENET SWI Sum of bytes of frames counted in TOTAL_DISC (HW_ENET_SWI_TOTAL_BYT_DISC)

Sum of bytes of frames counted in TOTAL_DISC

Address: HW_ENET_SWI_TOTAL_BYT_DISC – 800F_8000h base + 304h offset = 800F_8304h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOTAL_BYT_DISC																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_TOTAL_BYT_DISC field descriptions

Field	Description
31–0 TOTAL_BYT_ DISC	ENET SWI Sum of bytes of frames counted in TOTAL_DISC

29.9.94 ENET SWI Total number of incoming frames processed (HW_ENET_SWI_TOTAL_FRM)

Total number of incoming frames processed and not discarded.

Address: HW_ENET_SWI_TOTAL_FRM – 800F_8000h base + 308h offset = 800F_8308h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOTAL_FRM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_TOTAL_FRM field descriptions

Field	Description
31–0 TOTAL_FRM	ENET SWI Total number of incoming frames processed

29.9.95 ENET SWI Sum of bytes of frames counted in TOTAL_FRM (HW_ENET_SWI_TOTAL_BYT_FRM)

Sum of bytes of frames counted in TOTAL_FRM

Address: HW_ENET_SWI_TOTAL_BYT_FRM – 800F_8000h base + 30Ch offset = 800F_830Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TOTAL_BYT_FRM																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_TOTAL_BYT_FRM field descriptions

Field	Description
31–0 TOTAL_BYT_FRM	ENET SWI Sum of bytes of frames counted in TOTAL_FRM

29.9.96 ENET SWI Port 0 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC0)

Port 0 Outgoing frames discarded due to output Queue congestion.

Address: HW_ENET_SWI_ODISC0 – 800F_8000h base + 310h offset = 800F_8310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ODISC0																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_ODISC0 field descriptions

Field	Description
31–0 ODISC0	ENET SWI Port 0 Outgoing frames discarded due to output Queue congestion

29.9.97 ENET SWI Port 0 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN0)

Port 0 incoming frames discarded due to mismatching or missing VLAN id while VLAN verification was enabled See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_VLAN0 – 800F_8000h base + 314h offset = 800F_8314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_VLAN0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_VLAN0 field descriptions

Field	Description
31–0 IDISC_VLAN0	ENET SWI Port 0 incoming frames discarded due to mismatching or missing VLAN id

29.9.98 ENET SWI Port 0 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED0)

Port 0 incoming frames discarded due to missing vlan tag. See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_UNTAGGED0 – 800F_8000h base + 318h offset = 800F_8318h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_UNTAGGED0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_UNTAGGED0 field descriptions

Field	Description
31–0 IDISC_ UNTAGGED0	ENET SWI Port 0 incoming frames discarded due to missing vlan tag

29.9.99 ENET SWI Port 0 incoming frames discarded (after learning) as port is configured in blocking mode (HW_ENET_SWI_IDISC_BLOCKED0)

Port 0 incoming frames discarded (after learning) as port is configured in blocking mode

Address: HW_ENET_SWI_IDISC_BLOCKED0 – 800F_8000h base + 31Ch offset = 800F_831Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_BLOCKED0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_BLOCKED0 field descriptions

Field	Description
31–0 IDISC_ BLOCKED0	ENET SWI Port 0 incoming frames discarded (after learning) as port is configured in blocking mode

29.9.100 ENET SWI Port 1 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC1)

Port 1 Outgoing frames discarded due to output Queue congestion.

Address: HW_ENET_SWI_ODISC1 – 800F_8000h base + 320h offset = 800F_8320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ODISC1																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_ODISC1 field descriptions

Field	Description
31–0 ODISC1	ENET SWI Port 1 Outgoing frames discarded due to output Queue congestio

29.9.101 ENET SWI Port 1 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN1)

Port 1 incoming frames discarded due to mismatching or missing VLAN id while VLAN verification was enabled See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_VLAN1 – 800F_8000h base + 324h offset = 800F_8324h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_VLAN1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_VLAN1 field descriptions

Field	Description
31–0 IDISC_VLAN1	ENET SWI Port 1 incoming frames discarded due to mismatching or missing VLAN id

29.9.102 ENET SWI Port 1 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED1)

Port 1 incoming frames discarded due to missing vlan tag. See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_UNTAGGED1 – 800F_8000h base + 328h offset = 800F_8328h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_UNTAGGED1																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_UNTAGGED1 field descriptions

Field	Description
31–0 IDISC_ UNTAGGED1	ENET SWI Port 1 incoming frames discarded due to missing vlan tag

29.9.103 ENET SWI Port 1 incoming frames discarded (after learning) as port is configured in blocking mode (HW_ENET_SWI_IDISC_BLOCKED1)

Port 1 incoming frames discarded (after learning) as port is configured in blocking mode

Address: HW_ENET_SWI_IDISC_BLOCKED1 – 800F_8000h base + 32Ch offset = 800F_832Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_BLOCKED1																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_BLOCKED1 field descriptions

Field	Description
31–0 IDISC_ BLOCKED1	ENET SWI Port 1 incoming frames discarded (after learning) as port is configured in blocking mode

29.9.104 ENET SWI Port 2 Outgoing frames discarded due to output Queue congestion (HW_ENET_SWI_ODISC2)

Port 2 Outgoing frames discarded due to output Queue congestion.

Address: HW_ENET_SWI_ODISC2 – 800F_8000h base + 330h offset = 800F_8330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ODISC2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_ODISC2 field descriptions

Field	Description
31–0 ODISC2	ENET SWI Port 2 Outgoing frames discarded due to output Queue congestion

29.9.105 ENET SWI Port 2 incoming frames discarded due to mismatching or missing VLAN id (HW_ENET_SWI_IDISC_VLAN2)

Port 2 incoming frames discarded due to mismatching or missing VLAN id while VLAN verification was enabled See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_VLAN2 – 800F_8000h base + 334h offset = 800F_8334h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_VLAN2																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_VLAN2 field descriptions

Field	Description
31–0 IDISC_VLAN2	ENET SWI Port 2 incoming frames discarded due to mismatching or missing VLAN id

29.9.106 ENET SWI Port 2 incoming frames discarded due to missing vlan tag (HW_ENET_SWI_IDISC_UNTAGGED2)

Port 2 incoming frames discarded due to missing vlan tag. See also Register VLAN_VERIFY.

Address: HW_ENET_SWI_IDISC_UNTAGGED2 – 800F_8000h base + 338h offset = 800F_8338h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_UNTAGGED2																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_UNTAGGED2 field descriptions

Field	Description
31–0 IDISC_ UNTAGGED2	ENET SWI Port 2 incoming frames discarded due to missing vlan tag

29.9.107 ENET SWI Port 2 incoming frames discarded (after learning) as port is configured in blocking mod (HW_ENET_SWI_IDISC_BLOCKED2)

Port 2 incoming frames discarded (after learning) as port is configured in blocking mode

Address: HW_ENET_SWI_IDISC_BLOCKED2 – 800F_8000h base + 33Ch offset = 800F_833Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IDISC_BLOCKED2																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_IDISC_BLOCKED2 field descriptions

Field	Description
31–0 IDISC_ BLOCKED2	ENET SWI Port 2 incoming frames discarded (after learning) as port is configured in blocking mod

29.9.108 ENET SWI Interrupt Event Register (HW_ENET_SWI_EIR)

The event bits are latched. To clear a bit it must be written with 1. The bit will stay set if the event condition persists.

Address: HW_ENET_SWI_EIR – 800F_8000h base + 400h offset = 800F_8400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																						LRN	OD2	OD1	OD0	QM	TXF	TXB	RXF	RXB	EBERR
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

HW_ENET_SWI_EIR field descriptions

Field	Description
31–10 RSRVD0	Reserved bits. Write as 0.
9 LRN	Learning Record available in registers LNR_REC_0 and LNR_REC_1 (Signal ipi_lrn_int asserted). Note: this interrupt can be very frequent on a heavy loaded network. It is not recommended to use this interrupt source as interrupt but rather implement a slow background task polling the bit to perform learning.
8 OD2	Outgoing frames discarded due to output Queue congestion on Port 2 or port is disabled (PORT_ENA). Asserts ipi_od2_int
7 OD1	Outgoing frames discarded due to output Queue congestion on Port 1or port is disabled (PORT_ENA). Asserts ipi_od1_int
6 OD0	Outgoing frames discarded due to output Queue congestion on Port 0 or port is disabled (PORT_ENA). Asserts ipi_od0_int
5 QM	Low Memory Threshold. Asserted if the memory became congested and number of free cells dropped below threshold QMGR_MINCELLS (Signal ipi_qm_int asserted).

HW_ENET_SWI_EIR field descriptions (continued)

Field	Description
	Note: will become asserted after reset immediately due to memory initialization.
4 TXF	Transmit frame interrupt. This bit indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated (Signal ipi_txf_int asserted).
3 TXB	Transmit buffer interrupt. This bit indicates a transmit buffer descriptor has been updated (Signal ipi_txb_int asserted).
2 RXF	Receive frame interrupt. This bit indicates a frame has been received and the last corresponding buffer descriptor has been updated (Signal ipi_rxf_int asserted).
1 RXB	Receive buffer interrupt. This bit indicates a receive buffer descriptor not the last in the frame has been updated (Signal ipi_rxb_int asserted).
0 EBERR	Ethernet bus error. This bit indicates a system bus error occurs when a DMA transaction is underway (Signal ipi_eberr_int asserted).

29.9.109 ENET SWI Interrupt Mask Register (HW_ENET_SWI_EIMR)

Interrupt mask. Each bit corresponds to an interrupt source defined by the EIR register. The corresponding EIMR bit determines whether an interrupt condition can generate an interrupt. At every processor clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR bit reflects the state of the interrupt signal even if the corresponding EIMR bit is set. 0 The corresponding interrupt source is masked. 1 The corresponding interrupt source is not masked and an interrupt can occur (asserting corresponding ipi_xxx signal).

Address: HW_ENET_SWI_EIMR – 800F_8000h base + 404h offset = 800F_8404h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0																						LRN	OD2	OD1	OD0	QM	TXF	TXB	RXF	RXB	EBERR
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_EIMR field descriptions

Field	Description
31–10 RSRVD0	Reserved bits. Write as 0.
9 LRN	0: interrupt masked1 1: interrupt enabled.
8 OD2	0: interrupt masked 1: interrupt enabled.

HW_ENET_SWI_EIMR field descriptions (continued)

Field	Description
7 OD1	0: interrupt masked 1: interrupt enabled.
6 OD0	0: interrupt masked 1: interrupt enabled.
5 QM	0: interrupt masked 1: interrupt enabled.
4 TXF	0: interrupt masked 1: interrupt enabled.
3 TXB	0: interrupt masked 1: interrupt enabled.
2 RXF	0: interrupt masked 1: interrupt enabled.
1 RXB	0: interrupt masked 1: interrupt enabled.
0 EBERR	0: interrupt masked 1: interrupt enabled.

29.9.110 ENET SWI Pointer to Receive Descriptor Ring (HW_ENET_SWI_ERDSR)

Pointer to Receive Descriptor Ring. Only Bits 31:2 are writeable. 1:0 always 0.

Address: HW_ENET_SWI_ERDSR – 800F_8000h base + 408h offset = 800F_8408h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERDSR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERDSR[15:2]														RSRVD0	
W															▨	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_ERDSR field descriptions

Field	Description
31–2 ERDSR	ERDSR
1–0 RSRVD0	Reserved bits. Write as 0.

29.9.111 ENET SWI Pointer to Transmit Descriptor Ring (HW_ENET_SWI_ETDSR)

Pointer to Transmit Descriptor Ring. Only Bits 31:2 are writeable. 1:0 always 0.

Address: HW_ENET_SWI_ETDSR – 800F_8000h base + 40Ch offset = 800F_840Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ETDSR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ETDSR[15:2]														RSRVD0	
W															▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_ETDSR field descriptions

Field	Description
31–2 ETDSR	ERDSR
1–0 RSRVD0	Reserved bits. Write as 0.

29.9.112 ENET SWI Maximum Receive Buffer Size (HW_ENET_SWI_EMRBR)

Maximum Receive Buffer Size.

Address: HW_ENET_SWI_EMRBR – 800F_8000h base + 410h offset = 800F_8410h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1																		EMRBR										RSRVD0			
W	▨																												▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_EMRBR field descriptions

Field	Description
31–14 RSRVD1	Reserved bits. Write as 0.
13–4 EMRBR	ENET SWI Maximum Receive Buffer Size
3–0 RSRVD0	Reserved bits. Write as 0.

29.9.113 ENET SWI Receive Descriptor Active Register (HW_ENET_SWI_RDAR)

Receive Descriptor Active Register

Address: HW_ENET_SWI_RDAR – 800F_8000h base + 414h offset = 800F_8414h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RDAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_RDAR field descriptions

Field	Description
31–0 RDAR	ENET SWI Receive Descriptor Active Register

29.9.114 ENET SWI Transmit Descriptor Active Register (HW_ENET_SWI_TDAR)

Transmit Descriptor Active Register

Address: HW_ENET_SWI_TDAR – 800F_8000h base + 418h offset = 800F_8418h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TDAR																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_TDAR field descriptions

Field	Description
31–0 TDAR	ENET SWI Transmit Descriptor Active Register

29.9.115 ENET SWI Learning Records A (0) and B (1) (HW_ENET_SWI_LRN_REC_0)

Learning Records A (0) and B (1).

Lower 32-Bit of the Frame MAC Address. 7:0 = first octet, 31:24=4th octet. Note: this register must be read first, before reading LRN_REC_1

Address: HW_ENET_SWI_LRN_REC_0 – 800F_8000h base + 500h offset = 800F_8500h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LRN_REC_0																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_LRN_REC_0 field descriptions

Field	Description
31–0 LRN_REC_0	ENET SWI Learning Records A (0) and B (1)

29.9.116 ENET SWI Learning Record B(1) (HW_ENET_SWI_LRN_REC_1)

Learning Record B(1).

Address: HW_ENET_SWI_LRN_REC_1 – 800F_8000h base + 504h offset = 800F_8504h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0						SW_PORT		HASH							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAC_ADDR1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_LRN_REC_1 field descriptions

Field	Description
31–26 RSRVD0	Reserved bits. Write as 0.
25–24 SW_PORT	Port number on which the Frame is received.
23–16 HASH	The 8-bit Hash value
15–0 MAC_ADDR1	Upper 16-Bit of the Frame MAC Address. 7:0=5th octet, 15:8=6th octet.

29.9.117 ENET SWI Learning data available status. (HW_ENET_SWI_LRN_STATUS)

Address: HW_ENET_SWI_LRN_STATUS – 800F_8000h base + 508h offset = 800F_8508h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:1]															LRN_ STATUS
W	▨															▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_LRN_STATUS field descriptions

Field	Description
31–1 RSRVD0	Reserved bits. Write as 0.
0 LRN_STATUS	1 indicates if the learning record is valid and can be read.

29.9.119 ENET SWI lookup MAC address memory end (HW_ENET_SWI_LOOKUP_MEMORY_END)

Address: HW_ENET_SWI_LOOKUP_MEMORY_END – 800F_8000h base + FFFCh offset = 8010_7FFCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MEMORY_DATA																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_ENET_SWI_LOOKUP_MEMORY_END field descriptions

Field	Description
31–0 MEMORY_DATA	Memroy cell.

Chapter 30
Application UART (AUART)

30.1 Application UART Overview

The Application UART:

- Performs serial-to-parallel conversion on data received from a peripheral device.
- Performs parallel-to-serial conversion on data transmitted to the peripheral device.
- Operates up to 3.25 Mb/s.

The CPU or DMA controller reads and writes data and control/status information through the APBX interface. The transmit and receive paths are buffered with internal FIFO memories, enabling up to 16 bytes to be stored independently in both transmit and receive modes.

The Application UART includes a programmable baud rate generator that generates a transmit and receive internal clock from the 24 MHz UART internal reference clock input UARTCLK. XCLK (apbclk) is not tied to the UARTCLK in the i.MX28. Automatic baud rate detection is supported up to 1.2 Mbps. One or two reference frames can be used in detecting the baud rate.

It offers similar functionality to the industry-standard 16C550 UART device and supports baud rates of up to 3.25 Mbits/s (in high-speed configuration with a minimum XCLK frequency of 1.5 MHz). Figure 30-1 shows a block diagram of the Application UART. The Application UART operation and baud rate values are controlled by the line control register (HW_UARTAPP_LINECTRL). The HW_UARTAPP_LINECTRL register controls both receive and transmit operations. However, when HW_UARTAPP_CTRL2_USE_LCR2 is set, then HW_UARTAPP_LINECTRL controls receive operations and HW_UARTAPP_LINECTRL2 controls transmit operations.

The Application UART can generate a single combined interrupt, so that the output is asserted if any of the individual interrupts are asserted and unmasked. Interrupt sources include the receive (including timeout), transmit, modem status, and error conditions.

Two DMA channels are supported, one for transmit and one for receive.

If a timeout condition occurs in the middle of a receive DMA block transfer, then the UART ends the DMA transfer and signals the end of the DMA block transfer. A receive DMA can be setup to get the status of the previous receive DMA block transfer. The status indicates the amount of valid data bytes in the previous receive DMA block transfer.

If a framing, parity, or break error occurs during reception, the appropriate error bit is set and stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately and FIFO data is prevented from being overwritten. You can program the FIFOs to be one-byte deep, providing a conventional double-buffered UART interface.

The modem status input signal Clear To Send (CTS) and output modem control line Request To Send (RTS) are supported. A programmable hardware flow control feature uses the nUARTCTS input and the nUARTRTS output to automatically control the serial data flow.

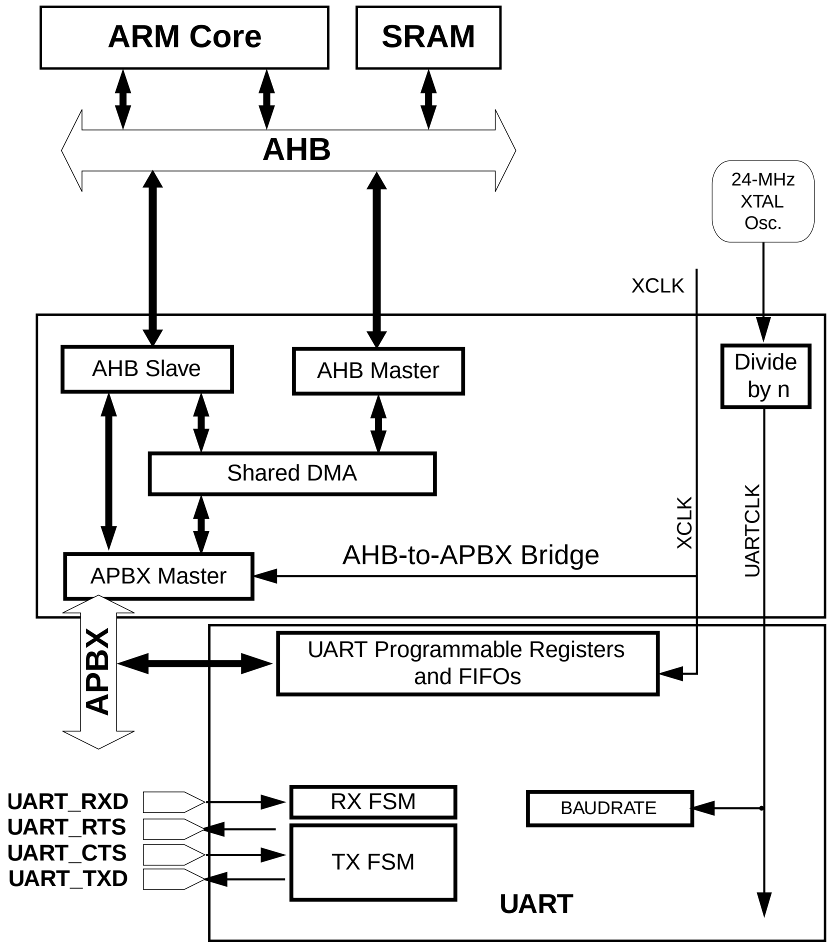


Figure 30-1. Application UART Block Diagram

30.2 Operation

Control data is written to the Application UART line control register. This register defines:

- Transmission parameters
- Word length
- Buffer mode
- Number of transmitted stop bits
- Parity mode
- Break generation
- Baud rate divisor

If USE_LCR2 is set, the Application UART Line Control Register applies to the receive operation, and similar control data written to the Application UART Line Control 2 Register applies to the transmit operation.

30.2.1 Fractional Baud Rate Divider

The baud rate divisor is calculated from the frequency of UARTCLK and the desired baud rate by using the following formula:

```
divisor = (UARTCLK * 32) / baud rate, rounded to the nearest integer
```

The divisor must be between 0x000000EC and 0x003FFFC0, inclusive. Program the lowest 6 bits of the divisor into BAUD_DIVFRAC, and the next 16 bits of the divisor into BAUD_DIVINT.

30.2.2 Automatic Baud Rate Detection

The receive baud rate can be automatically detected by the Application UART.

Some UART protocol are based on packets of frames with the first 1 or 2 frames being reference frames for baud detection. The UARTAPP detection logic is capable of starting in the middle of a RX packet and find the reference frame at the start of the next packet, then set the corresponding baud rate for the remainder of the packet. Note, the RXD is gated off to the UART Receiver module until the reference frames are detected. So, the reference frames are stripped from the packet and absent from the RXFIFO.

The UART firmware will tell the hardware when to detect the baud rate. The Autobaud Detection logic can be configured to detect RX baud rate each time a RX DMA is kicked off, or just each time the firmware writes the START_BAUD_DETECT bit in the AUTOBAUD register. Once the baud rate is found, it is loaded into the RX baud divisor register, and optionally loaded into the TX baud divisor register. Refer to the AutoBaud register for programming details.

30.2.3 UART Character Frame

Figure 30-2 illustrates the UART character frame.

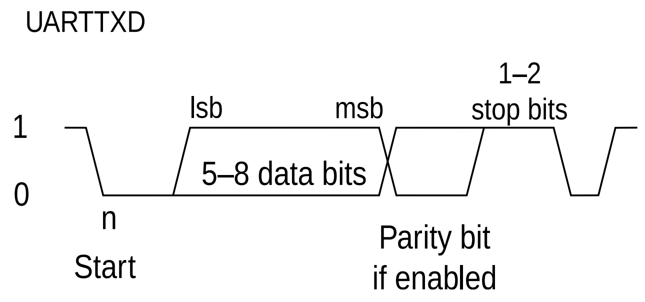


Figure 30-2. Application UART Character Frame

30.2.4 DMA Operation

The Application UART can generate a DMA request signal for interfacing with a Direct Memory Access (DMA) controller. Two DMA channels are supported, one for transmit and one for receive. Each channel has an associated 16-bit transfer counter for the number of bytes to transfer. Each DMA request is associated with one to four data bytes. For APBX DMA UART RX channel, the first PIO word in the DMA command is CTRL0. However, for APBX DMA UART TX channel, the first PIO word in a DMA command is CTRL1.

At the end of a receive DMA block transfer, the status register indicates any error conditions. If a timeout condition occurs in the middle of a receive DMA block transfer, then the UART sends dummy data to the DMA controller until the transfer counter is decremented to zero. A receive DMA can be setup to get the status of the previous receive DMA block transfer. The status indicates the amount of valid data bytes in the previous receive DMA block transfer.

30.2.5 Data Transmission or Reception

Data received or transmitted is stored in two 16-byte FIFOs, although the receive FIFO has an extra four bits per character for status information.

For transmission, data is written into the transmit FIFO. If the Application UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UARTAPP_LINECTRL or UARTAPP_LINECTRL2 (if USE_LCR2 is set). Data continues to be transmitted until there is no data left in the transmit FIFO.

The BUSY signal goes HIGH as soon as the data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. BUSY is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. BUSY can be asserted HIGH, even though the Application UART might no longer be enabled.

For each sample of data, three readings are taken and the majority value is kept. In the following paragraphs, the middle sampling point is defined, and one sample is taken on either side of it.

- When the receiver is idle (UARTRXD continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by BaudClk, begins running and data is sampled on the first cycle of that counter in normal UART mode to allow for the shorter logic 0 pulses (half way through a bit period).
- The start bit is valid if UARTRXD is still LOW on the first cycle of BaudClk, otherwise a false start bit is detected and it is ignored. If the start bit was valid, successive data bits are sampled on every second cycle of BaudClk (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.
- Lastly, a valid stop bit is confirmed if UARTRXD is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word (see Table 30-1).

30.2.6 Error Bits

Three error bits are stored in bits [10:8] of the receive FIFO and are associated with a particular character. An additional error indicating an overrun error is stored in bit 11 of the receive FIFO.

30.2.7 Overrun Bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full and the next character is completely received in the shift register. The data in the shift register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared. The table shows the bit functions of the receive FIFO.

Table 30-1. Receive FIFO Bit Functions

FIFO bit	Function
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

30.2.8 Disabling the FIFOs

FIFOs can be disabled. In this case, the transmit and receive sides of the Application UART have one-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read.

In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a one-byte register.

30.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

30.4 Programmable Registers

UARTAPP Hardware Register Format Summary

UARTAPP0 base address is 0x8006A000; UARTAPP1 base address is 0x8006C000; UARTAPP2 base address is 0x8006E000; UARTAPP3 base address is 0x80070000; UARTAPP4 base address is 0x80072000

HW_UARTAPP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8006_A000	UART Receive DMA Control Register (HW_UARTAPP_CTRL0)	32	R/W	C003_0000h	30.4.1/1943
8006_A010	UART Transmit DMA Control Register (HW_UARTAPP_CTRL1)	32	R/W	0000_0000h	30.4.2/1944
8006_A020	UART Control Register (HW_UARTAPP_CTRL2)	32	R/W	0022_0300h	30.4.3/1945
8006_A030	UART Line Control Register (HW_UARTAPP_LINECTRL)	32	R/W	0000_0000h	30.4.4/1948
8006_A040	UART Line Control 2 Register (HW_UARTAPP_LINECTRL2)	32	R/W	0000_0000h	30.4.5/1949
8006_A050	UART Interrupt Register (HW_UARTAPP_INTR)	32	R/W	0000_0000h	30.4.6/1950
8006_A060	UART Data Register (HW_UARTAPP_DATA)	32	R/W	0000_0000h	30.4.7/1952
8006_A070	UART Status Register (HW_UARTAPP_STAT)	32	R/W	C9F0_0000h	30.4.8/1954
8006_A080	UART Debug Register (HW_UARTAPP_DEBUG)	32	R	0000_0000h	30.4.9/1955
8006_A090	UART Version Register (HW_UARTAPP_VERSION)	32	R	0301_0000h	30.4.10/1957
8006_A0A0	UART AutoBaud Register (HW_UARTAPP_AUTOBAUD)	32	R/W	0000_0000h	30.4.11/1957

30.4.1 UART Receive DMA Control Register (HW_UARTAPP_CTRL0)

The UART Receive DMA Control Register contains the dynamic information associated with the receive command.

HW_UARTAPP_CTRL0: 0x000

HW_UARTAPP_CTRL0_SET: 0x004

HW_UARTAPP_CTRL0_CLR: 0x008

HW_UARTAPP_CTRL0_TOG: 0x00C

This register contains the main DMA controls for Receiving data.

Address: HW_UARTAPP_CTRL0 – 8006_A000h base + 0h offset = 8006_A000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	RUN	RX_SOURCE	RXTO_ ENABLE	RXTIMEOUT										
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XFER_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_CTRL0 field descriptions

Field	Description
31 SFTRST	Set to zero for normal operation. When this bit is set to one (default), then the entire block is held in its reset state.
30 CLKGATE	Set this bit zero for normal operation. Setting this bit to one (default), gates all of the block level clocks off for miniminizing AC energy consumption.
29 RUN	Tell the UART to execute the RX DMA Command. The UART will clear this bit at the end of receive execution.
28 RX_SOURCE	Source of Receive Data. If this bit is set to 1, the status register will be the source of the DMA, otherwise RX data will be the source.
27 RXTO_ENABLE	RXTIMEOUT Enable: If this bit is set to 0, the RX timeout will not affect receive DMA operation. If this bit is set to 1, a receive timeout will cause the receive DMA logic to terminate.
26–16 RXTIMEOUT	Receive Timeout Counter Value: number of 8-bit-time to wait before asserting timeout on the RX input. If the RXFIFO is not empty and the RX input is idle, then the watchdog counter will decrement each bit-time. Note 7-bit-time is added to the programmed value, so a value of zero will set the counter to 7-bit-time, a value of 0x1 gives 15-bit-time and so on. Also note that the counter is reloaded at the end of each frame, so if the frame is 10 bits long and the timeout counter value is zero, then timeout will occur (when FIFO is not empty) even if the RX input is not idle. The default value is 0x3 (31 bit-time).
15–0 XFER_COUNT	Number of bytes to receive. This must be a multiple of 4.

30.4.2 UART Transmit DMA Control Register (HW_UARTAPP_CTRL1)

The UART Transmit DMA Control Register contains the dynamic information associated with the transmit command.

HW_UARTAPP_CTRL1: 0x010

HW_UARTAPP_CTRL1_SET: 0x014

HW_UARTAPP_CTRL1_CLR: 0x018

HW_UARTAPP_CTRL1_TOG: 0x01C

This register contains the main DMA controls for Transmitting data.

Address: HW_UARTAPP_CTRL1 – 8006_A000h base + 10h offset = 8006_A010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2			RUN	RSVD1												XFER_COUNT															
W	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_CTRL1 field descriptions

Field	Description
31–29 RSVD2	Reserved, read as zero, do not modify.
28 RUN	Tell the UART to execute the TX DMA Command. The UART will clear this bit at the end of transmit execution.
27–16 RSVD1	Reserved, read as zero, do not modify.
15–0 XFER_COUNT	Number of bytes to transmit.

30.4.3 UART Control Register (HW_UARTAPP_CTRL2)

The UART Control Register contains configuration, including interrupt FIFO level select and the DMA control.

HW_UARTAPP_CTRL2: 0x020

HW_UARTAPP_CTRL2_SET: 0x024

HW_UARTAPP_CTRL2_CLR: 0x028

HW_UARTAPP_CTRL2_TOG: 0x02C

Use this register to define the FIFO level at which the UARTTXINTR and UARTRXINTR are triggered. The interrupts are generated based on a transition through a level rather than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level. The bits are reset so that the trigger level is when the FIFOs are at the half-way mark.

Address: HW_UARTAPP_CTRL2 – 8006_A000h base + 20h offset = 8006_A020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INVERT_RTS	INVERT_CTS	INVERT_TX	INVERT_RX	RTS_ SEMAPHORE	DMAONERR	TXDMAE	RXDMAE	RSVD2	RXIFLSEL			RSVD3	TXIFLSEL		
W									▨				▨			
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CTSEN	RTSEN	OUT2	OUT1	RTS	DTR	RXE	TXE	LBE	USE_ LCR2	RSVD4			SIRLP	SIREN	UARTEN
W											▨	▨	▨			
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

HW_UARTAPP_CTRL2 field descriptions

Field	Description
31 INVERT_RTS	Invert RTS signal. If this bit is set to 1, the RTS output is inverted before transmitted.
30 INVERT_CTS	Invert CTS signal. If this bit is set to 1, the CTS input is inverted before sampled.
29 INVERT_TX	Invert TX signal. If this bit is set to 1, the TX output is inverted before transmitted.
28 INVERT_RX	Invert RX signal. If this bit is set to 1, the RX input is inverted before sampled.
27 RTS_ SEMAPHORE	If this bit is set to 1, RTS is deasserted when the semaphore threshold is less than 2.
26 DMAONERR	DMA On Error. If this bit is set to 1, receive dma will terminate on error. (Cmd_end signal may not be asserted when this occurs.)
25 TXDMAE	Transmit DMA Enable. Data Register can be loaded with up to 4 bytes per write. TXFIFO must be enabled in TXDMA mode.
24 RXDMAE	Receive DMA Enable. Data Register can be contain up to 4 bytes per read. RXFIFO must be enabled in RXDMA mode.
23 RSVD2	Reserved, do not modify, read as zero.
22–20 RXIFLSEL	Receive Interrupt FIFO Level Select. The trigger points for the receive interrupt are as follows: 0x0 **NOT_EMPTY** — Trigger on FIFO containing at least 2 of 16 entries. 0x1 **ONE_QUARTER** — Trigger on FIFO full to at least 4 of 16 entries. 0x2 **ONE_HALF** — Trigger on FIFO full to at least 8 of 16 entries. 0x3 **THREE_QUARTERS** — Trigger on FIFO full to at least 12 of 16 entries. 0x4 **SEVEN_EIGHTHS** — Trigger on FIFO full to at least 14 of 16 entries.

HW_UARTAPP_CTRL2 field descriptions (continued)

Field	Description
	0x5 **INVALID5** — Reserved. 0x6 **INVALID6** — Reserved. 0x7 **INVALID7** — Reserved.
19 RSVD3	Reserved, do not modify, read as zero.
18–16 TXIFLSEL	Transmit Interrupt FIFO Level Select. The trigger points for the transmit interrupt are as follows: 0x0 **EMPTY** — Trigger on FIFO less than or equal to 2 of 16 entries. 0x1 **ONE_QUARTER** — Trigger on FIFO less than or equal to 4 of 16 entries. 0x2 **ONE_HALF** — Trigger on FIFO less than or equal to 8 of 16 entries. 0x3 **THREE_QUARTERS** — Trigger on FIFO less than or equal to 12 of 16 entries. 0x4 **SEVEN_EIGHTHS** — Trigger on FIFO less than or equal to 14 of 16 entries. 0x5 **INVALID5** — Reserved. 0x6 **INVALID6** — Reserved. 0x7 **INVALID7** — Reserved.
15 CTSEN	CTS Hardware Flow Control Enable. If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the nUARTCTS signal is asserted.
14 RTSEN	RTS Hardware Flow Control Enable. If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received. The FIFO space is controlled by RXIFLSEL value.
13 OUT2	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. This bit is not supported.
12 OUT1	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. This bit is not supported.
11 RTS	Request To Send. Software can manually control the nUARTRTS pin through this bit when RTSEN = 0. This bit is the complement of the UART request to send (nUARTRTS) modem status output. That is, when the bit is programmed to a 1, the output is 0.
10 DTR	Data Transmit Ready. This bit is the complement of the UART data transmit ready (nUARTDTR) modem status output. This bit is not supported.
9 RXE	Receive Enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for the UART signals. When the UART is disabled in the middle of reception, it completes the current character before stopping.
8 TXE	Transmit Enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for the UART signals. When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7 LBE	Loop Back Enable. This feature reduces the amount of external coupling required during system test. If this bit is set to 1, the UARTTXD path is fed through to the UARTRXD path. When this bit is set, the modem outputs are also fed through to the modem inputs.
6 USE_LCR2	=If this bit is set to 1, the Line Control 2 Register values are used.
5–3 RSVD4	Reserved, do not modify, read as zero.

HW_UARTAPP_CTRL2 field descriptions (continued)

Field	Description
2 SIRLP	IrDA SIR Low Power Mode. Unsupported.
1 SIREN	SIR Enable. Unsupported.
0 UARTEN	UART Enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for the UART signals. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

30.4.4 UART Line Control Register (HW_UARTAPP_LINECTRL)

HW_UARTAPP_LINECTRL: 0x030

HW_UARTAPP_LINECTRL_SET: 0x034

HW_UARTAPP_LINECTRL_CLR: 0x038

HW_UARTAPP_LINECTRL_TOG: 0x03C

The UART Line Control Register contains integer and fractional part of the baud rate divisor value. It also contains the line control bits.

Address: HW_UARTAPP_LINECTRL – 8006_A000h base + 30h offset = 8006_A030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BAUD_DIVINT																RSVD		BAUD_DIVFRAC						SPS	WLEN		FEN	STP2	EPS	PEN	BRK
W																	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_LINECTRL field descriptions

Field	Description
31–16 BAUD_DIVINT	Baud Rate Integer [15:0]. The integer baud rate divisor.
15–14 RSVD	Reserved, do not modify, read as zero.
13–8 BAUD_DIVFRAC	Baud Rate Fraction [5:0]. The fractional baud rate divisor.
7 SPS	Stick Parity Select. When bits 1, 2, and 7 of this register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled.
6–5 WLEN	Word length [1:0]. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits, 10 = 7 bits, 01 = 6 bits, 00 = 5 bits.

HW_UARTAPP_LINECTRL field descriptions (continued)

Field	Description
4 FEN	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, the FIFOs are disabled (character mode); that is, the FIFOs become 1-byte-deep holding registers.
3 STP2	Two Stop Bits Select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2 EPS	Even Parity Select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (PEN, bit 1) being cleared to 0.
1 PEN	Parity Enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.
0 BRK	Send Break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

30.4.5 UART Line Control 2 Register (HW_UARTAPP_LINECTRL2)

HW_UARTAPP_LINECTRL2: 0x040

HW_UARTAPP_LINECTRL2_SET: 0x044

HW_UARTAPP_LINECTRL2_CLR: 0x048

HW_UARTAPP_LINECTRL2_TOG: 0x04C

The UART Line Control 2 Register contains integer and fractional part of the baud rate divisor value. It also contains the line control bits.

Address: HW_UARTAPP_LINECTRL2 – 8006_A000h base + 40h offset = 8006_A040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BAUD_DIVINT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD		BAUD_DIVFRAC						SPS	WLEN		FEN	STP2	EPS	PEN	RSVD1
W	▨	▨														▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_LINECTRL2 field descriptions

Field	Description
31–16 BAUD_DIVINT	Baud Rate Integer [15:0]. The integer baud rate divisor.
15–14 RSVD	Reserved, do not modify, read as zero.
13–8 BAUD_DIVFRAC	Baud Rate Fraction [5:0]. The fractional baud rate divisor.
7 SPS	Stick Parity Select. When bits 1, 2, and 7 of this register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled.
6–5 WLEN	Word length [1:0]. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits, 10 = 7 bits, 01 = 6 bits, 00 = 5 bits.
4 FEN	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0, the FIFOs are disabled (character mode); that is, the FIFOs become 1-byte-deep holding registers.
3 STP2	Two Stop Bits Select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2 EPS	Even Parity Select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0, then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by Parity Enable (PEN, bit 1) being cleared to 0.
1 PEN	Parity Enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame.
0 RSVD1	Reserved, do not modify, read as zero.

30.4.6 UART Interrupt Register (HW_UARTAPP_INTR)

HW_UARTAPP_INTR: 0x050

HW_UARTAPP_INTR_SET: 0x054

HW_UARTAPP_INTR_CLR: 0x058

HW_UARTAPP_INTR_TOG: 0x05C

The UART Interrupt Register contains the interrupt enables and the interrupt status. The interrupt status bits report the unmasked state of the interrupts. To clear a particular interrupt status bit, write the bit-clear address with the particular bit set to 1. The enable bits control the UART interrupt output: a 1 will enable a particular interrupt to assert the UART interrupt output, while a 0 will disable the particular interrupt from affecting the interrupt output. All the bits, except for the modem status interrupt bits, are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Address: HW_UARTAPP_INTR – 8006_A000h base + 50h offset = 8006_A050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1				ABDIEN	OEIEN	BEIEN	PEIEN	FEIEN	RTIEN	TXIEN	RXIEN	DSRMIEN	DCDMIEN	CTSMIEN	RIMIEN
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				ABDIS	OEIS	BEIS	PEIS	FEIS	RTIS	TXIS	RXIS	DSRMIS	DCDMIS	CTSMIS	RIMIS
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_INTR field descriptions

Field	Description
31–28 RSVD1	Reserved, read as zero, do not modify.
27 ABDIEN	Automatic Baudrate Detected Interrupt Enable.
26 OEIEN	Overrun Error Interrupt Enable.
25 BEIEN	Break Error Interrupt Enable.
24 PEIEN	Parity Error Interrupt Enable.
23 FEIEN	Framing Error Interrupt Enable.
22 RTIEN	Receive Timeout Interrupt Enable.
21 TXIEN	Transmit Interrupt Enable.
20 RXIEN	Receive Interrupt Enable.
19 DSRMIEN	nUARTDSR Modem Interrupt Enable. This bit is not supported.

HW_UARTAPP_INTR field descriptions (continued)

Field	Description
18 DCDMIEN	nUARTDCD Modem Interrupt Enable. This bit is not supported.
17 CTSMIEN	nUARTCTS Modem Interrupt Enable.
16 RIMIEN	nUARTRI Modem Interrupt Enable. This bit is not supported.
15–12 RSVD2	Reserved, read as zero, do not modify.
11 ABDIS	Automatic Baudrate Detected Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
10 OEIS	Overrun Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
9 BEIS	Break Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
8 PEIS	Parity Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
7 FEIS	Framing Error Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
6 RTIS	Receive Timeout Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
5 TXIS	Transmit Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
4 RXIS	Receive Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
3 DSRMIS	nUARTDSR Modem Interrupt Status. This bit is not supported.
2 DCDMIS	nUARTDCD Modem Interrupt Status. This bit is not supported.
1 CTSMIS	nUARTCTS Modem Interrupt Status. To clear this bit, write the bit-clear address with the particular bit set to 1.
0 RIMIS	nUARTRI Modem Interrupt Status. This bit is not supported.

30.4.7 UART Data Register (HW_UARTAPP_DATA)

The UART Data Register is the receive and transmit data register. Receive (read) and transmit (write) up to four data characters per APB cycle.

For words to be transmitted: 1) If the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO; 2) If the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO). The write operation initiates transmission from the PrimeCell UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted. Note: With the use of APB byte-enables you can write 1, 2, or 4 valid bytes sumultaneously to the TXFIFO. The invalid bytes will also take up space in the TXFIFO. So every write cycle will consume 4 bytes in the TXFIFO. If TXFIFO is disabled, you must only write the LSByte of the DATA register. For received words: 1) If the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO; 2) if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO). The received data bytes (up to 4) are read by performing reads from the 32-bit DATA register. The status information can be read by a read of the UART Status register. The Overrun Error bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it. The Break Error bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received. When the Parity Error bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the LCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO. When the Framing Error bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.

Address: HW_UARTAPP_DATA – 8006_A000h base + 60h offset = 8006_A060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_DATA field descriptions

Field	Description
31–0 DATA	In DMA mode, up to 4 Received/Transmit characters can be accessed at a time. In PIO mode, only one character can be accessed at a time. The status register contains the receive data flags and valid bits.

30.4.8 UART Status Register (HW_UARTAPP_STAT)

The UART Status Register contains the various flags and receive status. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from the UART Data Register prior to reading the UART Status Register. The status information for overrun is set immediately when an overrun condition occurs.

Address: HW_UARTAPP_STAT – 8006_A000h base + 70h offset = 8006_A070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	HISPEED	BUSY	CTS	TXFE	RXFF	TXFF	RXFE	RXBYTE_INVALID				OERR	BERR	PERR	FERR
W	▨	▨	▨	▨	▨	▨	▨	▨					▨			
Reset	1	1	0	0	1	0	0	1	1	1	1	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXCOUNT															
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_STAT field descriptions

Field	Description
31 PRESENT	This read-only bit indicates that the Application UART function is present when it reads back a one. This Application UART function is not available on a device that returns a zero for this bit field. 0x0 **UNAVAILABLE** — UARTAPP is not present in this product. 0x1 **AVAILABLE** — UARTAPP is present in this product.
30 HISPEED	This read-only bit indicates that the high-speed function is present when it reads back a one. This high speed function is not available on a device that returns a zero for this bit field. 0x0 **UNAVAILABLE** — HISPEED is not present in this product. 0x1 **AVAILABLE** — HISPEED is present in this product.
29 BUSY	UART Busy.
28 CTS	Clear To Send.
27 TXFE	Transmit FIFO Empty. The meaning of this bit depends on the state of the FEN bit in the UART Line Control Register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.

HW_UARTAPP_STAT field descriptions (continued)

Field	Description
26 RXFF	Receive FIFO Full.
25 TXFF	Transmit FIFO Full.
24 RXFE	Receive FIFO Empty.
23–20 RXBYTE_ INVALID	The invalid state of the last read of Receive Data. Each bit corresponds to one byte of the RX data. (1 = invalid.)
19 OERR	Overrun Error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by any write to the Status Register. The FIFO contents remain valid since no further data is written when the FIFO is full; only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
18 BERR	Break Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Break Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.
17 PERR	Parity Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Parity Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.
16 FERR	Framing Error. For PIO mode, this is for the last character read from the data register. For DMA mode, it will be set to 1 if any received character for a particular RXDMA command had a Framing Error. To clear this bit, write a zero to it. Note that clearing this bit does not affect the interrupt status, which must be cleared by writing the interrupt register.
15–0 RXCOUNT	Number of bytes received during a Receive DMA command.

30.4.9 UART Debug Register (HW_UARTAPP_DEBUG)

The UART Debug Register contains the state of the DMA signals.

Address: HW_UARTAPP_DEBUG – 8006_A000h base + 80h offset = 8006_A080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXIBAUD_DIV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXFBAUD_DIV						RSVD1				TXDMARUN	RXDMARUN	TXCMDEND	RXCMDEND	TXDMARQ	RXDMARQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_DEBUG field descriptions

Field	Description
31–16 RXIBAUD_DIV	RX Integer Baud Divisor.
15–10 RXFBAUD_DIV	RX Fractional Baud Divisor.
9–6 RSVD1	Reserved, read as zero, do not modify.
5 TXDMARUN	DMA Command Run Status: This bit reflects the state of the toggle signal for TXDMARUN.
4 RXDMARUN	DMA Command Run Status: This bit reflects the state of the toggle signal for RXDMARUN.
3 TXCMDEND	DMA Command End Status: This bit reflects the state of the toggle signal for UART_TXCMDEND.
2 RXCMDEND	DMA Command End Status: This bit reflects the state of the toggle signal for UART_RXCMDEND.
1 TXDMARQ	DMA Request Status: This bit reflects the state of the toggle signal for UART_TXDMAREQ. Note that TX burst request is not supported.
0 RXDMARQ	DMA Request Status: This bit reflects the state of the toggle signal for UART_RXDMAREQ. Note that RX burst request is not supported.

30.4.10 UART Version Register (HW_UARTAPP_VERSION)

The UART version register can be used to read the version of the UARTAPP IP being used in this chip.

Address: HW_UARTAPP_VERSION – 8006_A000h base + 90h offset = 8006_A090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Re-set	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of RTL version.

30.4.11 UART AutoBaud Register (HW_UARTAPP_AUTOBAUD)

The UART AutoBaud register provides the reference characters and control info for the automatic baudrate detection logic.

Address: HW_UARTAPP_AUTOBAUD – 8006_A000h base + A0h offset = 8006_A0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	REFCHAR1								REFCHAR0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1											UPDATE_TX	TWO_REF_CHARS	START_WITH_RUNBIT	START_BAUD_DETECT	BAUD_DETECT_ENABLE
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_UARTAPP_AUTOBAUD field descriptions

Field	Description
31–24 REFCHAR1	Second reference character used in baud rate detection. During autobaud detection of the second reference frame is stripped from the incoming packet. So it will not appear in the RXFIFO. This field is ignored when TWO_REF_CHARS is 0.
23–16 REFCHAR0	First reference character used in baud rate detection. During autobaud detection of the first reference frame received is not available for reading. It is stripped from the incoming packet, so the UART receiver does not get this character and the RXFIFO is not updated.
15–5 RSVD1	Reserved, read as zero, do not modify.
4 UPDATE_TX	Set this bit to 1 will cause the TX baud rate divisor to be updated when the RX baud rate divisor is updated by the autobaud detection logic. This should not be set if it is possible that transmit function may be busy.
3 TWO_REF_ CHARS	Set this bit to 1 when using 2 reference characters for baud detection, and set to 0 for 1 reference character.
2 START_WITH_ RUNBIT	Set this bit to 1 will cause the assertion of HW_UARTAPP_CTRL0_RUN to start the autobaud detection logic. Set this bit to 0 will cause the assertion of START_BAUD_DETECT to start the autobaud detection logic.
1 START_BAUD_ DETECT	Set to 1 to start automatic baudrate detection. This bit is ignored when START_WITH_RUNBIT is set to 1. Each time a 1 is written to START_BAUD_DETECT it toggles the read value if START_WITH_RUNBIT is zero.
0 BAUD_DETECT_ ENABLE	Enable automatic baudrate detection.

Chapter 31
USB High-Speed On-the-Go Host Device Controller

31.1 USB High-Speed OTG-capable USB Controller Overview

The

i.MX28 has two USB high-speed controllers, the first one is a OTG-capable USB high speed controller(a.k.a USB0), and the second one is host-only high-speed USB controller(a.k.a USB1). This chapter describes the USB high-speed OTG-capable controller(USB0) included on the i.MX28. The host-only high-speed USB controller(USB1) is a subset of the OTG-capable controller(USB0). This chapter includes sections on the PIO, DMA, and UTMI interfaces, along with USB controller flowcharts.

The i.MX28 includes a Universal Serial Bus (USB) version 2.0 controller capable of operating as either a USB device or a USB host, as shown in Figure 31-1. In addition, USB0 contains supporting circuitry for USB On-the-Go (OTG). USB1 is configured as host-only controller. The USB controller is used to download digital music data or program code into external memory and to upload voice recordings from memory to the PC. Program updates can also be loaded into the flash memory area using the USB interface.

As a host controller, it can enumerate and control USB devices attached to it. The USB controller included on the i.MX28 supports eight endpoints: one control, one bulk-out, one bulk-in, and five flexible endpoints. Using the OTG features, the USB controller can negotiate with another OTG system to be either the host or the device in a peer connection.

The USB controller operates either in full-speed mode or high-speed mode.

Refer to the USB Implementer's Forum website www.usb.org for detailed specifications and information on the USB protocol, timing and electrical characteristics.

The USB 2.0 controller comprises both a programmed I/O (PIO) interface and a DMA interface. Both of these interfaces are designed to meet an ARM Ltd. AMBA Hardware Bus (AHB). The AHB is used by the USB controller as a slave (PIO register accesses) and as a master (DMA memory accesses).

The USB 2.0 PHY is fully integrated on-chip and is described in USB PHY Overview. The PHY is controlled over the APBX peripheral bus.

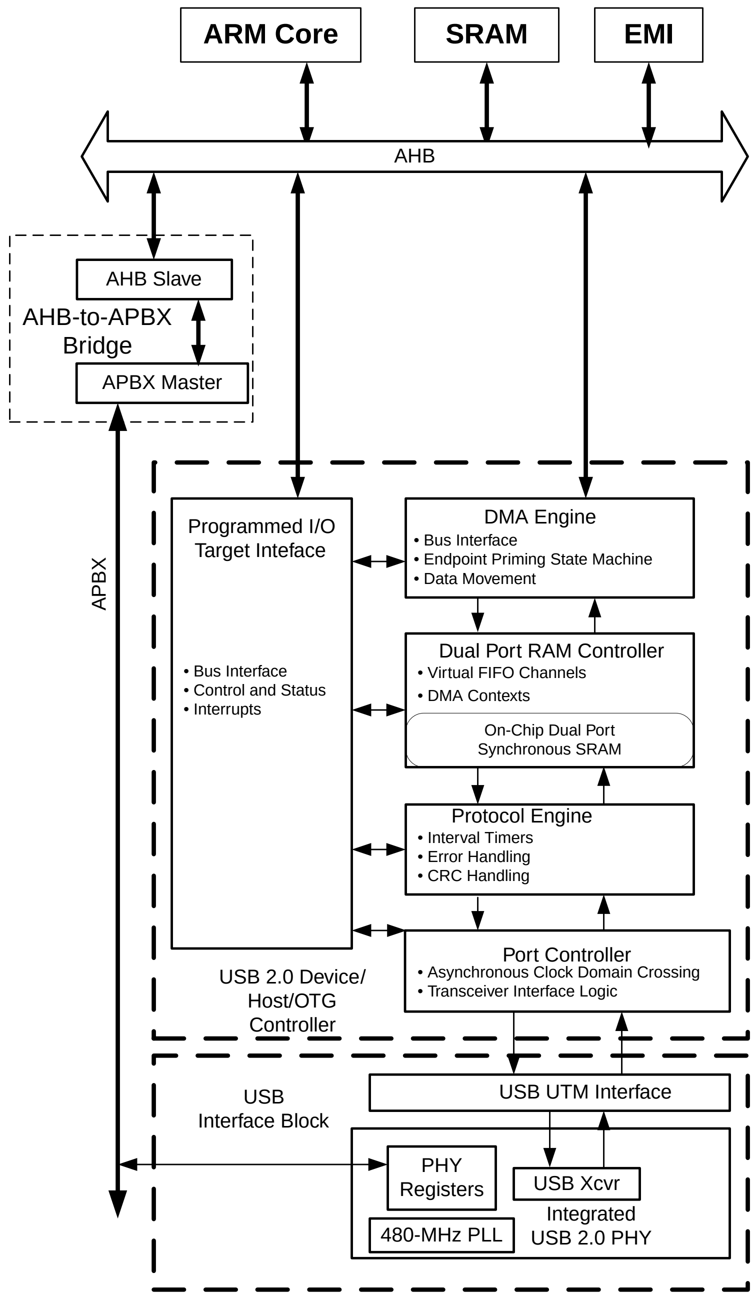


Figure 31-1. USB 2.0 Device Controller Block Diagram

31.2 Difference between USB0 and USB1

As mentioned above, USB0 is the High speed OTG-capable USB controller and USB1 is the host only high speed USB controller.

USB0 is capable of supplying VBUS, while USB1 is for connection to on-system (on-board) connectivity peripherals such as cellular modems and no support of VBUS.

31.3 USB Programmed I/O (PIO) Target Interface

The PIO interface is on an AHB slave of the USB controller. It allows the ARM processor to access the configuration, control, and status registers. There are identification registers for hardware configuration parameters and operational registers for control and status.

31.4 USB DMA Interface

The DMA is a master AHB interface that allows USB data to be transferred to/from the system memory. The data in memory is structured to implement a software framework supported by the controller. For a device controller, this structure is a linked-list interface that consists of queue heads and pointers that are transfer descriptors. The queue head is where transfers are managed. It has status information and location of the data buffers. The hardware controller's PIO registers enable the entire data structure, and once USB data is transferred between the host, the status of the transfer is updated in the queue head, with minimal latency to the system.

For a host controller, there is also a linked-list interface. It consists of a periodic frame list and pointers to transfer descriptors. The period frame list is a schedule of transfers. The frame list points to the data buffers through the transfer descriptors. The hardware controller's PIO registers enable the data structure and manage the transfers within a USB frame. The period frame list works as a sliding window of host transfers over time. As each transfer is completed, the status information is updated in the frame list.

31.5 USB UTM Interface

The USB UTM interface on the i.MX28 implements the specification that allows USB controllers to interface with the USB PHY. Please refer to the *USB 2.0 Transceiver Macrocell Interface (UTMI)* Specification, *Version 1.05*, for additional details: http://www.intel.com/technology/usb/spec.htm

31.5.1 Digital/Analog Loopback Test Mode

Since the UTM has to operate at high frequencies (480 MHz), it has a capacity to self-test. A pseudo-random number generator transmits data to the receive path, and data is compared for validity. In the digital loopback, the data transfer only resides in the UTM. It checks for sync, EOP, and bit-stuffing generation and data integrity. The analog loopback is the same as the digital loopback, but involves the analog PHY. This allows for checking of the high-speed (HS) and full-speed (FS) comparators and transmitters.

31.6 USB Controller Flowcharts

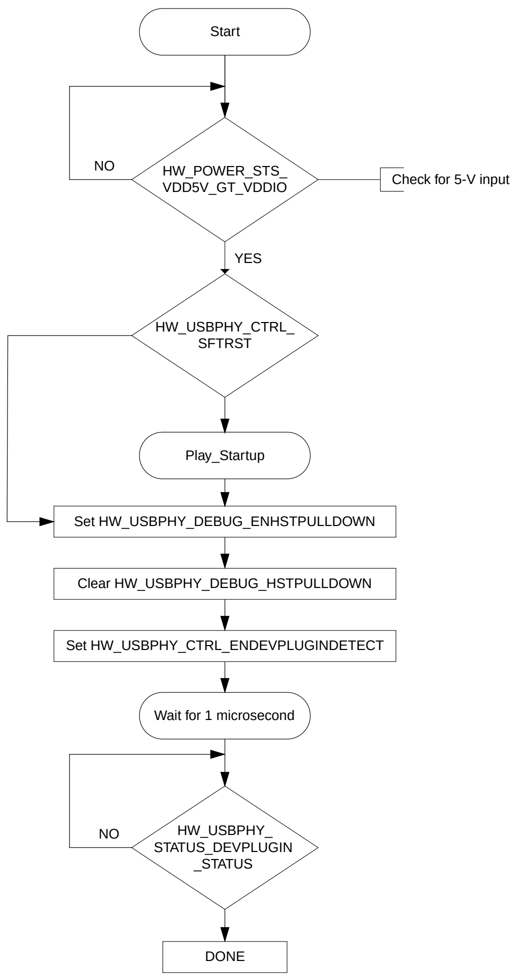


Figure 31-2. USB 2.0 Check_USB_Plugged_In Flowchart

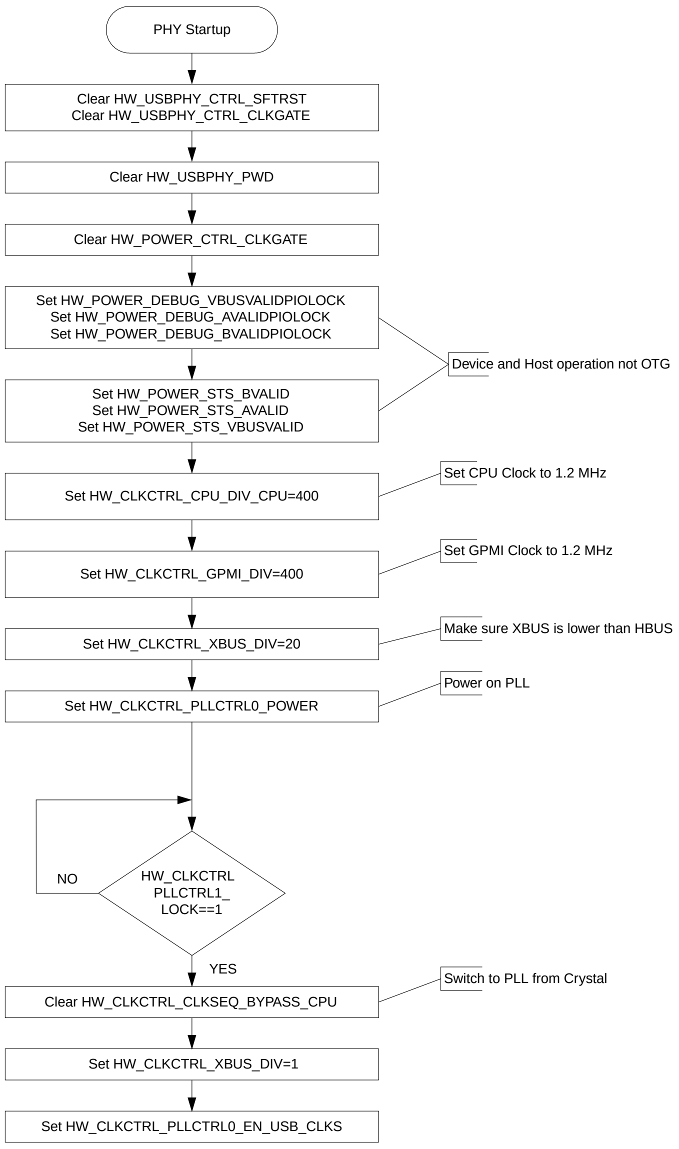

PHY Startup
Clear HW_USBPHY_CTRL_SFTRST
Clear HW_USBPHY_CTRL_CLKGATE
Clear HW_USBPHY_PWD
Clear HW_POWER_CTRL_CLKGATE
Set HW_POWER_DEBUG_VBUSVALIDPIOLOCK
Set HW_POWER_DEBUG_AVALIDPIOLOCK
Set HW_POWER_DEBUG_BVALIDPIOLOCK
Device and Host operation not OTG
Set HW_POWER_STS_BVALID
Set HW_POWER_STS_AVALID
Set HW_POWER_STS_VBUSVALID
Set HW_CLKCTRL_CPU_DIV_CPU=400
Set CPU Clock to 1.2 MHz
Set HW_CLKCTRL_GPMI_DIV=400
Set GPMI Clock to 1.2 MHz
Set HW_CLKCTRL_XBUS_DIV=20
Make sure XBUS is lower than HBUS
Set HW_CLKCTRL_PLLCTRL0_POWER
Power on PLL
NO
HW_CLKCTRL PLLCTRL1_ LOCK==1
YES
Clear HW_CLKCTRL_CLKSEQ_BYPASS_CPU
Switch to PLL from Crystal
Set HW_CLKCTRL_XBUS_DIV=1
Set HW_CLKCTRL_PLLCTRL0_EN_USB_CLKS

Figure 31-3. USB 2.0 USB PHY Startup Flowchart

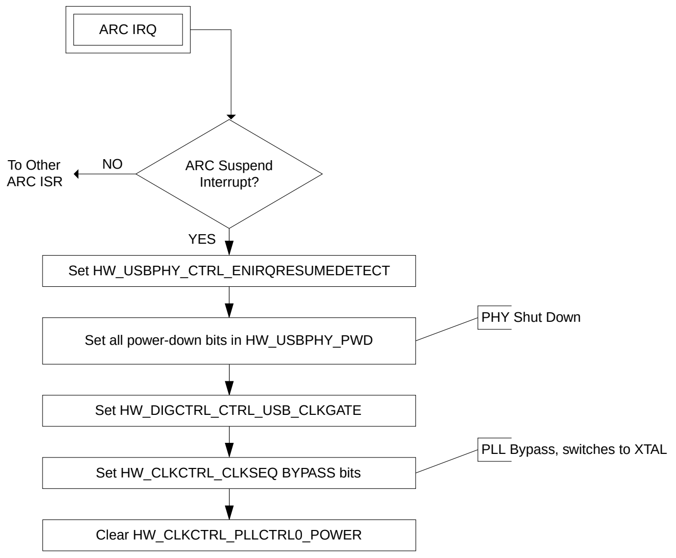

ARC IRQ
ARC Suspend Interrupt?
NO
To Other ARC ISR
YES
Set HW_USBPHY_CTRL_ENIRQRESUMEDETECT
Set all power-down bits in HW_USBPHY_PWD
PHY Shut Down
Set HW_DIGCTRL_CTRL_USB_CLKGATE
Set HW_CLKCTRL_CLKSEQ BYPASS bits
PLL Bypass, switches to XTAL
Clear HW_CLKCTRL_PLLCTRL0_POWER

Figure 31-4. USB 2.0 PHY PLL Suspend Flowchart

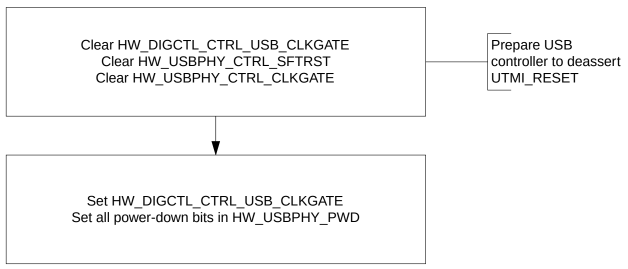

Clear HW_DIGCTL_CTRL_USB_CLKGATE
Clear HW_USBPHY_CTRL_SFTRST
Clear HW_USBPHY_CTRL_CLKGATE
Prepare USB controller to deassert UTMI_RESET
Set HW_DIGCTL_CTRL_USB_CLKGATE
Set all power-down bits in HW_USBPHY_PWD

Figure 31-5. UTMI Powerdown

31.6.1 References

- *USB 2.0 Transceiver Macrocell Interface (UTMI) Specification*, Version 1.05, March 2001, Jon Lueker, Steve McGowan (Editor) Ken Oliver, Dean Warren. http://www.intel.com
- *VSI Alliance Virtual Component Interface Standard*, Version 2 (OCB 2 2.0), April 2001, On-Chip Bus Development Working Group. http://www.vsi.org
- *Universal Serial Bus Specification*, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org
- *On-The-Go Supplement to the USB 2.0 Specification*, Revision 1.0, Dec 2001, On-The-Go Working Group of the USB-IF. http://www.usb.org
- *Enhanced Host Controller Interface Specification for Universal Serial Bus*, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com

- *Universal Serial Bus Specification*, Revision 1.1, September 1998, Compaq, Intel, Microsoft, NEC. http://www.usb.org
- *AMBA Specification*, Revision 2.0, May 1999, ARM Limited. http://www.arm.com
- *UTMI+ Low Pin Interface (ULPI) Specification*, Revision 1.0 February 2004, ULPI Specification Organization. http://www.ulpi.org

31.7 Programmable Registers

USBCTRL Hardware Register Format Summary

USB Controller0 base address is 0x80080000; USB Controller1 base address is 0x80090000

HW_USBCTRL memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8008_0000	Identification Register (HW_USBCTRL_ID)	32	R	E401_FA05h	31.7.1/1967
8008_0004	General Hardware Parameters Register (HW_USBCTRL_HWGENERAL)	32	R	0000_0015h	31.7.2/1968
8008_0008	Host Hardware Parameters Register (HW_USBCTRL_HWHOST)	32	R	1002_0001h	31.7.3/1969
8008_000C	Device Hardware Parameters Register (HW_USBCTRL_HWDEVICE)	32	R	0000_0011h	31.7.4/1969
8008_0010	TX Buffer Hardware Parameters Register (HW_USBCTRL_HWTXBUF)	32	R	8007_0A10h	31.7.5/1970
8008_0014	RX Buffer Hardware Parameters Register (HW_USBCTRL_HWRXBUF)	32	R	0000_0810h	31.7.6/1971
8008_0080	General-Purpose Timer 0 Load (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER0LD)	32	R/W	0000_0000h	31.7.7/1971
8008_0084	General-Purpose Timer 0 Control (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER0CTRL)	32	R/W	0000_0000h	31.7.8/1972
8008_0088	General-Purpose Timer 1 Load (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER1LD)	32	R/W	0000_0000h	31.7.9/1973
8008_008C	General-Purpose Timer 1 Control (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER1CTRL)	32	R/W	0000_0000h	31.7.10/1974
8008_0090	System Bus Configuration (Non-EHCI-Compliant) Register (HW_USBCTRL_SBUSCFG)	32	R/W	0000_0000h	31.7.11/1975
8008_0100	Capability Length and HCI Version (EHCI-Compliant) Register (HW_USBCTRL_CAPLENGTH)	32	R	0100_0040h	31.7.12/1976
8008_0104	Host Control Structural Parameters (EHCI-Compliant with Extensions) Register (HW_USBCTRL_HCSPARAMS)	32	R	0001_0011h	31.7.13/1976
8008_0108	Host Control Capability Parameters (EHCI-Compliant) Register (HW_USBCTRL_HCCPARAMS)	32	R	0000_0006h	31.7.14/1978

HW_USBCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8008_0120	Device Interface Version Number (Non-EHCI-Compliant) Register (HW_USBCTRL_DCIVERSION)	32	R/W	0000_0001h	31.7.15/1979
8008_0124	Device Control Capability Parameters (Non-EHCI-Compliant) Register (HW_USBCTRL_DCCPARAMS)	32	R	0000_0188h	31.7.16/1980
8008_0140	USB Command Register (HW_USBCTRL_USBCMD)	32	R/W	0008_0000h	31.7.17/1980
8008_0144	USB Status Register (HW_USBCTRL_USBSTS)	32	R/W	0000_0000h	31.7.18/1983
8008_0148	USB Interrupt Enable Register (HW_USBCTRL_USBINTR)	32	R/W	0000_0000h	31.7.19/1987
8008_014C	USB Frame Index Register (HW_USBCTRL_FRINDEX)	32	R/W	0000_0000h	31.7.20/1989
8008_0154	Frame List Base Address Register (Host Controller mode) (HW_USBCTRL_PERIODICLISTBASE)	32	R/W	0000_0000h	31.7.21/1990
8008_0154	USB Device Address Register (Device Controller mode) (HW_USBCTRL_DEVICEADDR)	32	R/W	0000_0000h	31.7.22/1991
8008_0158	Next Asynchronous Address Register (Host Controller mode) (HW_USBCTRL_ASYNCLISTADDR)	32	R/W	0000_0000h	31.7.23/1992
8008_0158	Endpoint List Address Register (Device Controller mode) (HW_USBCTRL_ENDPOINTLISTADDR)	32	R/W	0000_0000h	31.7.24/1993
8008_015C	Embedded TT Asynchronous Buffer Status and Control Register (Host Controller mode) (HW_USBCTRL_TTCTRL)	32	R/W	0000_0000h	31.7.25/1993
8008_0160	Programmable Burst Size Register (HW_USBCTRL_BURSTSIZE)	32	R/W	0000_1010h	31.7.26/1994
8008_0164	Host Transmit Pre-Buffer Packet Timing Register (HW_USBCTRL_TXFILLTUNING)	32	R/W	0000_0000h	31.7.27/1995
8008_016C	Inter-Chip Control Register (HW_USBCTRL_IC_USB)	32	R/W	0000_0000h	31.7.28/1997
8008_0170	ULPI Viewport Register (HW_USBCTRL_ULPI)	32	R/W	0000_0000h	31.7.29/1998
8008_0178	Endpoint NAK Register (HW_USBCTRL_ENDPTNAK)	32	R/W	0000_0000h	31.7.30/1999
8008_017C	Endpoint NAK Enable Register (HW_USBCTRL_ENDPTNAKEN)	32	R/W	0000_0000h	31.7.31/2000
8008_0184	Port Status and Control 1 Register (HW_USBCTRL_PORTSC1)	32	R/W	1000_0000h	31.7.32/2001
8008_01A4	OTG Status and Control Register (HW_USBCTRL_OTGSC)	32	R/W	0000_0120h	31.7.33/2007
8008_01A8	USB Device Mode Register (HW_USBCTRL_USBMODE)	32	R/W	0000_0000h	31.7.34/2010
8008_01AC	Endpoint Setup Status Register (HW_USBCTRL_ENDPTSETUPSTAT)	32	R/W	0000_0000h	31.7.35/2012
8008_01B0	Endpoint Initialization Register (HW_USBCTRL_ENDPTPRIME)	32	R/W	0000_0000h	31.7.36/2012
8008_01B4	Endpoint De-Initialize Register (HW_USBCTRL_ENDPTFLUSH)	32	R/W	0000_0000h	31.7.37/2014
8008_01B8	Endpoint Status Register (HW_USBCTRL_ENDPTSTAT)	32	R	0000_0000h	31.7.38/2015

HW_USBCTRL memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8008_01BC	Endpoint Complete Register (HW_USBCTRL_ENDPTCOMPLETE)	32	R/W	0000_0000h	31.7.39/2016
8008_01C0	Endpoint Control 0 Register (HW_USBCTRL_ENDPTCTRL0)	32	R/W	0080_0080h	31.7.40/2017
8008_01C4	Endpoint Control 1 Register (HW_USBCTRL_ENDPTCTRL1)	32	R/W	0000_0000h	31.7.41/2019
8008_01C8	Endpoint Control 2 Register (HW_USBCTRL_ENDPTCTRL2)	32	R/W	0000_0000h	31.7.42/2022
8008_01CC	Endpoint Control 3 Register (HW_USBCTRL_ENDPTCTRL3)	32	R/W	0000_0000h	31.7.43/2023
8008_01D0	Endpoint Control 4 Register (HW_USBCTRL_ENDPTCTRL4)	32	R/W	0000_0000h	31.7.44/2025
8008_01D4	Endpoint Control 5 Register (HW_USBCTRL_ENDPTCTRL5)	32	R/W	0000_0000h	31.7.45/2026
8008_01D8	Endpoint Control 6 Register (HW_USBCTRL_ENDPTCTRL6)	32	R/W	0000_0000h	31.7.46/2028
8008_01DC	Endpoint Control 7 Register (HW_USBCTRL_ENDPTCTRL7)	32	R/W	0000_0000h	31.7.47/2029

31.7.1 Identification Register (HW_USBCTRL_ID)

The Identification Register provides a simple way to determine if the USB-HS USB 2.0 core is provided in the system. The HW_USBCTRL_ID register identifies the USB-HS USB 2.0 core and its revision. The default value of this register is 0xE241FA05.

Address: HW_USBCTRL_ID – 8008_0000h base + 0h offset = 8008_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CIVERSION			VERSION				REVISION				TAG					RSVD1		NID						RSVD0		ID					
W																																
Reset	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	1	0	1

HW_USBCTRL_ID field descriptions

Field	Description
31–29 CIVERSION	Identifies the Chip Idea product version of the USB-HS USB 2.0 core.
28–25 VERSION	Identifies the version of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
24–21 REVISION	Identifies the revision of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
20–16 TAG	Identifies the tag of the USB-HS USB 2.0 core release. (<version>.<revision><tag>)
15–14 RSVD1	Reserved.
13–8 NID	One's complement version of ID[5:0].

HW_USBCTRL_ID field descriptions (continued)

Field	Description
7–6 RSVD0	Reserved.
5–0 ID	Configuration number. This number is set to 0x05 and indicates that the peripheral is the USB-HS USB 2.0 core.

31.7.2 General Hardware Parameters Register (HW_USBCTRL_HWGENERAL)

The default value of this register is 0x00000015.

General Hardware Parameters

Address: HW_USBCTRL_HWGENERAL – 8008_0000h base + 4h offset = 8008_0004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:11]					SM		PHYM			PHYW		BWT	CLKC		RT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1

HW_USBCTRL_HWGENERAL field descriptions

Field	Description
31–11 RSVD	Reserved.
10–9 SM	PHY Serial Engine Type. Always 0 = Serial engine not present.
8–6 PHYM	PHY Type. Always 1 = UTMI.
5–4 PHYW	Data Interface Width to PHY. Always 1 = 16 bits.
3 BWT	Reserved for internal testing. Always 0.

HW_USBCTRL_HWGENERAL field descriptions (continued)

Field	Description
2–1 CLKC	USB Controller Clocking Method. Always 2 = Mixed clocked.
0 RT	Reset Type. Always 1 = Synchronous

31.7.3 Host Hardware Parameters Register (HW_USBCTRL_HWHOST)

The default value of this register is 0x10020001.

Host hardware params as defined in sys-level/core-config

Address: HW_USBCTRL_HWHOST – 8008_0000h base + 8h offset = 8008_0008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TTPER								TTASY								RSVD												NPORT			HC
W																																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_USBCTRL_HWHOST field descriptions

Field	Description
31–24 TTPER	Periodic contexts for hub TT.
23–16 TTASY	Asynch contexts for hub TT.
15–4 RSVD	Reserved.
3–1 NPORT	Maximum downstream ports minus 1.
0 HC	Host Capable. Always 0x1.

31.7.4 Device Hardware Parameters Register (HW_USBCTRL_HWDEVICE)

The default value of this register is 0x0000000B.

device hardware params as defined in sys-level/core-config

Address: HW_USBCTRL_HWDEVICE – 8008_0000h base + Ch offset = 8008_000Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																										DEVEP					DC
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

HW_USBCTRL_HWDEVICE field descriptions

Field	Description
31–6 RSVD	Reserved.
5–1 DEVEP	Maximum number of endpoints, which is 8.
0 DC	Device Capable. Always 0x1.

31.7.5 TX Buffer Hardware Parameters Register (HW_USBCTRL_HWTXBUF)

The default value of this register is 0x40060910.

tx hardware buf params

Address: HW_USBCTRL_HWTXBUF – 8008_0000h base + 10h offset = 8008_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXLCR	RSVD							TXCHANADD							
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXADD								TXBURST							
W																
Reset	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0

HW_USBCTRL_HWTXBUF field descriptions

Field	Description
31 TXLCR	Always 0x1.

HW_USBCTRL_HWTXBUF field descriptions (continued)

Field	Description
30–24 RSVD	Reserved.
23–16 TXCHANADD	Number of address bits for the TX buffer.
15–8 TXADD	Always 0xa.
7–0 TXBURST	Burst size for memory-to-TX-buffer transfers.

31.7.6 RX Buffer Hardware Parameters Register (HW_USBCTRL_HWRXBUF)

The default value of this register is 0x00000710.

rx hardware buf params

Address: HW_USBCTRL_HWRXBUF – 8008_0000h base + 14h offset = 8008_0014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																RXADD								RXBURST							
W	▨																▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

HW_USBCTRL_HWRXBUF field descriptions

Field	Description
31–16 RSVD	Reserved.
15–8 RXADD	Always 0x08.
7–0 RXBURST	Burst size for RX buffer-to-memory transfers.

31.7.7 General-Purpose Timer 0 Load (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER0LD)

The host/device controller driver can measure time-related activities using this timer register. This register is not part of the standard EHCI controller. This register contains the timer duration or load value. See the GPTIMER0CTRL (Non-EHCI) for a description of the timer functions.

General Purpose Timer #0 Load Register

Address: HW_USBCTRL_GPTIMER0LD – 8008_0000h base + 80h offset = 8008_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								GPTLD																							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_GPTIMER0LD field descriptions

Field	Description
31–24 RSVD0	Reserved.
23–0 GPTLD	General-Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one-millisecond timer, load 1000 Note: Max value is 0xFFFFFF or 16.777215 seconds.

31.7.8 General-Purpose Timer 0 Control (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER0CTRL)

The host/device controller driver can measure time-related activities using this timer register. This register is not part of the standard EHCI controller. This register contains the control for the timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two modes supported by this timer, the first is a one-shot and the second is a looped count that is described in the register table below. When the timer counter value transitions to 0, an interrupt can be generated through the use of the timer interrupts in the USBTS and USBINTR registers.

Address: HW_USBCTRL_GPTIMER0CTRL – 8008_0000h base + 84h offset = 8008_0084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GPTRUN	▨	RSVD0					GPTMODE	GPTCNT[23:16]							
W		GPTRST	▨	▨	▨	▨	▨		▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPTCNT[15:0]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_GPTIMER0CTRL field descriptions

Field	Description
31 GPTRUN	General-Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT except if it stops or starts counting. 0 **STOP** — Timer stop. 1 **RUN** — Timer run.
30 GPTRST	General-Purpose Timer Reset. Writing a 1 to this bit will reload the GPTCNT with the value in GPTLD. 0 **NOACTION** — No action. 1 **LOADCOUNTER** — Load counter value.
29–25 RSVD0	Reserved.
24 GPTMODE	General-Purpose Timer Mode. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer will count down to 0, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to 0, generate an interrupt, and automatically reload the counter to begin again. 0 **ONESHOT** — One shot. 1 **REPEAT** — Repeat.
23–0 GPTCNT	General-Purpose Timer Counter. This field is the value of the running timer.

31.7.9 General-Purpose Timer 1 Load (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER1LD)

Same as GPTIMER0LD description.

Address: HW_USBCTRL_GPTIMER1LD – 8008_0000h base + 88h offset = 8008_0088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0								GPTLD																							
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_GPTIMER1LD field descriptions

Field	Description
31–24 RSVD0	Reserved.
23–0 GPTLD	General-Purpose Timer Load Value. This field is the value to be loaded into the GPTCNT countdown timer on a reset action. This value in this register represents the time in microseconds minus 1 for the timer duration. Example: for a one-millisecond timer, load 1000 Note: Max value is 0xFFFFFF or 16.777215 seconds.

31.7.10 General-Purpose Timer 1 Control (Non-EHCI-Compliant) Register (HW_USBCTRL_GPTIMER1CTRL)

Same as GPTIMER0CTRL description.

Address: HW_USBCTRL_GPTIMER1CTRL – 8008_0000h base + 8Ch offset = 8008_008Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	GPTRUN	▨	RSVD0					GPTMODE	GPTCNT[23:16]							
W		GPTRST	▨						▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPTCNT[15:0]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_GPTIMER1CTRL field descriptions

Field	Description
31 GPTRUN	General-Purpose Timer Run. This bit enables the general-purpose timer to run. Setting or clearing this bit will not have an effect on the GPTCNT except if it stops or starts counting. 0 **STOP** — Timer stop. 1 **RUN** — Timer run.
30 GPTRST	General-Purpose Timer Reset. Writing a 1 to this bit will reload the GPTCNT with the value in GPTLD. 0 **NOACTION** — No action. 1 **LOADCOUNTER** — Load counter value.

HW_USBCTRL_GPTIMER1CTRL field descriptions (continued)

Field	Description
29–25 RSVD0	Reserved.
24 GPTMODE	General-Purpose Timer Mode. This bit selects between a single timer countdown and a looped count down. In one-shot mode, the timer will count down to 0, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to 0, generate an interrupt, and automatically reload the counter to begin again. 0 **ONESHOT** — One shot. 1 **REPEAT** — Repeat.
23–0 GPTCNT	General-Purpose Timer Counter. This field is the value of the running timer.

31.7.11 System Bus Configuration (Non-EHCI-Compliant) Register (HW_USBCTRL_SBUSCFG)

This register controls the AMBA system bus Master/Slave interfaces.

Address: HW_USBCTRL_SBUSCFG – 8008_0000h base + 90h offset = 8008_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																													AHBBRST		
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_SBUSCFG field descriptions

Field	Description
31–3 RSVD	Reserved.
2–0 AHBBRST	AMBA AHB BURST. This field selects the following options for the m_hburst signal of the AMBA master interface: 0x0 **U_INCR** — INCR burst of unspecified length. 0x1 **S_INCR4** — INCR4, non-multiple transfers of INCR4 will be decomposed into singles. 0x2 **S_INCR8** — INCR8, non-multiple transfers of INCR8 will be decomposed into INCR4 or singles. 0x3 **S_INCR16** — INCR16, non-multiple transfers of INCR16 will be decomposed into INCR8, INCR4 or singles. 0x4 **RESERVED** — This value is reserved and should not be used. 0x5 **U_INCR4** — INCR4, non-multiple transfers of INCR4 will be decomposed into smaller unspecified length bursts. 0x6 **U_INCR8** — INCR8, non-multiple transfers of INCR8 will be decomposed into smaller unspecified length bursts. 0x7 **U_INCR16** — INCR16, non-multiple transfers of INCR16 will be decomposed into smaller unspecified length bursts.

31.7.12 Capability Length and HCI Version (EHCI-Compliant) Register (HW_USBCTRL_CAPLENGTH)

This register contains the Capability Length and HCI Version Register.

Address: HW_USBCTRL_CAPLENGTH – 8008_0000h base + 100h offset = 8008_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HCIVERSION																RSVD								CAPLENGTH							
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

HW_USBCTRL_CAPLENGTH field descriptions

Field	Description
31–16 HCIVERSION	Contains a BCD encoding of the EHCI revision number supported by the host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
15–8 RSVD	Reserved.
7–0 CAPLENGTH	Offset to add to register base address at beginning of the Operational Register.

31.7.13 Host Control Structural Parameters (EHCI-Compliant with Extensions) Register (HW_USBCTRL_HCSPARAMS)

Port-steering logic capabilities are described in this register. The default value of this register is 0x00010011.

Address: HW_USBCTRL_HCSPARAMS – 8008_0000h base + 104h offset = 8008_0104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2				N_TT				N_PTT				RSVD1			PI
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	N_CC				N_PCC				RSVD0			PPC	N_PORTS			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

HW_USBCTRL_HCSPARAMS field descriptions

Field	Description
31–28 RSVD2	Reserved.
27–24 N_TT	Number of Transaction Translators (N_TT). Indicates the number of embedded transaction translators associated with the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. This in a non-EHCI field to support embedded TT.
19–17 RSVD1	Reserved.
16 PI	Port Indicators (P INDICATOR). Indicates whether the ports support port indicator control. When set to 1, the port status and control registers include a read/writable field for controlling the state of the port indicator.
15–12 N_CC	Number of Companion Controller (N_CC). Indicates the number of companion controllers associated with this USB2.0 host controller. A 0 in this field indicates there are no internal Companion Controllers. Port-ownership hand-off is not supported. A value larger than 0 in this field indicates there are companion USB host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports.
11–8 N_PCC	Number of Ports per Companion Controller. Indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC.
7–5 RSVD0	Reserved.
4 PPC	Port Power Control. Indicates whether the host controller implementation includes port power control. A 1 indicates the ports have port power switches. A 0 indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register.
3–0 N_PORTS	Number of downstream ports. Specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 0x11-0xF. A 0 in this field is undefined.

31.7.14 Host Control Capability Parameters (EHCI-Compliant) Register (HW_USBCTRL_HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality) addressing capability. The default value of this register is 0x00000006.

Address: HW_USBCTRL_HCCPARAMS – 8008_0000h base + 108h offset = 8008_0108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EECP								IST				RSVD0	ASP	PFL	ADC
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

HW_USBCTRL_HCCPARAMS field descriptions

Field	Description
31–16 RSVD2	Reserved.
15–8 EECP	EHCI Extended Capabilities Pointer. Default = 0. This optional field indicates the existence of a capabilities list. A value of 0x00 indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 0x40 or greater if implemented to maintain the consistency of the PCI header defined for this class of device.
7–4 IST	Isochronous Scheduling Threshold. Indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit 7 is 0, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit 7 is a 1, then host software assumes the host controller may cache an isochronous data structure for an entire frame.
3 RSVD0	Reserved.
2 ASP	Asynchronous Schedule Park Capability. Default = 1.

HW_USBCTRL_HCCPARAMS field descriptions (continued)

Field	Description
	If this bit is set to a 1, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the Asynchronous Schedule Park Mode Enable and Asynchronous Schedule Park Mode Count fields in the USBCMD register.
1 PFL	Programmable Frame List Flag. If this bit is set to 0, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to 0. If set to a 1, then the system software can specify and use a smaller frame list and configure the host controller through the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K page boundary. This requirement ensures that the frame list is always physically contiguous.
0 ADC	64-bit Addressing Capability. No 64-bit addressing capability is supported.

31.7.15 Device Interface Version Number (Non-EHCI-Compliant) Register (HW_USBCTRL_DCIVERSION)

The device controller interface conforms to the two-byte BCD encoding of the interface version number contained in this register.

Address: HW_USBCTRL_DCIVERSION – 8008_0000h base + 120h offset = 8008_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																DCIVERSION															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_USBCTRL_DCIVERSION field descriptions

Field	Description
31–16 RSVD	Reserved.
15–0 DCIVERSION	Two-byte BCD encoding of the interface version number.

31.7.16 Device Control Capability Parameters (Non-EHCI-Compliant) Register (HW_USBCTRL_DCCPARAMS)

These fields describe the overall host/device capability of the controller. The default value of this register is 0x00000188.

Address: HW_USBCTRL_DCCPARAMS – 8008_0000h base + 124h offset = 8008_0124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1																							HC	DC	RSVD2		DEN				
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

HW_USBCTRL_DCCPARAMS field descriptions

Field	Description
31–9 RSVD1	Reserved.
8 HC	Host Capable. When this bit is 1, this controller is capable of operating as an EHCI-compatible USB 2.0 host controller.
7 DC	Device Capable. When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 RSVD2	Reserved.
4–0 DEN	Device Endpoint Number. This field indicates the number of endpoints built into the device controller, which is 8.

31.7.17 USB Command Register (HW_USBCTRL_USBCMD)

The serial bus host/device controller executes the command indicated in this register. * Default Value:0x00080B00 (Host mode), 0x00080000 (Device mode)

Address: HW_USBCTRL_USBCMD – 8008_0000h base + 140h offset = 8008_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3								ITC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FS2	ATDTW	SUTW	RSVD2	ASPE	RSVD1	ASP		LR	IAA	ASE	PSE	FS1	FS0	RST	RS
W				▨		▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_USBCMD field descriptions

Field	Description
31–24 RSVD3	Reserved.
23–16 ITC	Interrupt Threshold Control. Default 0x08. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are: 0x0 **IMM** — Immediate (no threshold). 0x1 **1_MICROFRAME** — 1_MICROFRAME. 0x2 **2_MICROFRAME** — 2_MICROFRAME. 0x4 **4_MICROFRAME** — 4_MICROFRAME. 0x8 **8_MICROFRAME** — 8_MICROFRAME. 0x10 **16_MICROFRAME** — 16_MICROFRAME. 0x20 **32_MICROFRAME** — 32_MICROFRAME. 0x40 **64_MICROFRAME** — 64_MICROFRAME.
15 FS2	Bit 2 of Frame List Size field. See definition of bit FS0 for the complete definition.
14 ATDTW	Add dTD TripWire (device mode only). This bit is used as a semaphore to ensure the proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit shall also be cleared by hardware when is state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
13 SUTW	Setup TripWire (device mode only). This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (See USBMODE) then there exists a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software and will be cleared by hardware when a hazard exists.
12 RSVD2	Reserved.
11 ASPE	Asynchronous Schedule Park Mode Enable (OPTIONAL). When S/W changes the USBMODE.CM to Host(11), this bit defaults to 0x1. Software uses this bit to enable or disable Park mode. When this bit is 1, Park mode is enabled. When this bit is a 0, Park mode is disabled. This field is set to 1 in host mode; 0 in device mode.

HW_USBCTRL_USBCMD field descriptions (continued)

Field	Description
10 RSVD1	Reserved.
9–8 ASP	Asynchronous Schedule Park Mode Count (OPTIONAL). When S/W changes the USBMODE.CM to Host(11), this field defaults to 0x3 and is R/W. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the Asynchronous schedule before continuing traversal of the Asynchronous schedule. See Section 4.10.3.2 of the EHCI specification for full operational details. Valid values are 0x1-0x3. Software must not write a 0 to this bit as this will result in undefined behavior. This field is set to 0x3 in host mode; 0x0 in device mode.
7 LR	Light Host/Device Controller Reset (OPTIONAL). Not Implemented. This field will always be 0.
6 IAA	Interrupt on Async Advance Doorbell. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is 1, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to 0 after it has set the Interrupt on Sync Advance status bit in the USBSTS register to 1. Software should not write a 1 to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a 1 to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable. Default 0. This bit controls whether the host controller skips processing the Asynchronous Schedule. 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule. Only the host controller uses this bit.
4 PSE	Periodic Schedule Enable. Default Ob. This bit controls whether the host controller skips processing the Periodic Schedule. 0 = Do not process the Periodic Schedule 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule. Only the host controller uses this bit.
3 FS1	Bit 1 of Frame List Size field. See definition of bit FS0 for the complete definition.
2 FS0	Bit 0 of Frame List Size field. The Frame List Size field (FS2, FS1, FS0) specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. Note that this field is made up from USBCMD bits 15, 3 and 2. Default is 000b. 000b = 1024 ELEMENTS (4096 bytes) Default value. 001b = 512_ELEMENTS (2048 bytes).

HW_USBCTRL_USBCMD field descriptions (continued)

Field	Description
	010b = 256_ELEMENTS (1024 bytes). 011b = 128_ELEMENTS (512 bytes). 100b = 64_ELEMENTS (256 bytes). 101b = 32_ELEMENTS (128 bytes). 110b = 16_ELEMENTS (64 bytes). 111b = 8_ELEMENTS (32 bytes). Only the host controller uses this field.
1 RST	Controller Reset (RESET). Software uses this bit to reset the controller. This bit is set to 0 by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a 0 to this register. Host Controller: When software writes a 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a 1 when the HCHalted bit in the USBSTS register is a 0. Attempting to reset an actively running host controller will result in undefined behavior. Device Controller: When software writes a 1 to this bit, the Device Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a 1 to this bit when the device is in the attached state is not recommended, since the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS). Default 0. 1 = Run. 0 = Stop. Host Controller: When set to a 1, the Host Controller proceeds with the execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Host Controller has finished the transaction and has entered the stopped state. Software should not write a 1 to this field unless the host controller is in the Halted state (i.e., HCHalted in the USBSTS register is a 1). Device Controller: Writing a 1 to this bit will cause the device controller to enable a pullup on D+ and initiate an attach event. This control bit is not directly connected to the pullup enable, as the pullup will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the device controller has been properly initialized. Writing a 0 to this will cause a detach event.

31.7.18 USB Status Register (HW_USBCTRL_USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus. Software clears certain bits in this register by writing a 1 to them. * Default Value:0x00001000 (Host mode), 0x00000000 (Device mode)

Address: HW_USBCTRL_USBSTS – 8008_0000h base + 144h offset = 8008_0144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD5						TI1	TI0	RSVD4				UPI	UAI	RSVD3	NAKI
W	▨	▨	▨	▨	▨	▨			▨	▨	▨	▨			▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AS	PS	RCL	HCH	RSVD2	ULPII	RSVD1	SLI	SRI	URI	AAI	SEI	FRI	PCI	UEI	UI
W	▨	▨	▨		▨		▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_USBSTS field descriptions

Field	Description
31–26 RSVD5	Reserved.
25 TI1	General-Purpose Timer Interrupt 1 (GPTINT1). This bit is set when the counter in the GPTIMER1CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
24 TI0	General-Purpose Timer Interrupt 0 (GPTINT0). This bit is set when the counter in the GPTIMER0CTRL (Non-EHCI) register transitions to 0. Writing a 1 to this bit will clear it.
23–20 RSVD4	Reserved.
19 UPI	USB Host Periodic Interrupt (USBHSTPERINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set and the TD was from the periodic schedule. This bit is also set by the Host Controller when a short packet is detected AND the packet is on the periodic schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be 0.
18 UAI	USB Host Asynchronous Interrupt (USBHSTASYNCINT). This bit is set by the Host Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set AND the TD was from the asynchronous schedule. This bit is also set by the Host when a short packet is detected AND the packet is on the asynchronous schedule. A short packet is when the actual number of bytes received was less than the expected number of bytes. This bit is not used by the device controller and will always be 0.
17 RSVD3	Reserved.

HW_USBCTRL_USBSTS field descriptions (continued)

Field	Description
16 NAKI	NAK Interrupt Bit. It is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and the corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when the all the enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status. This bit reports the current real status of the Asynchronous Schedule. When set to 0 the asynchronous schedule status is disabled and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used by the host controller.
14 PS	Periodic Schedule Status. 0 = Default. This bit reports the current real status of the Periodic Schedule. When set to 0 the periodic schedule is disabled, and if set to 1 the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used by the host controller.
13 RCL	Reclamation. 0 = Default. This is a read-only status bit used to detect an empty asynchronous schedule. Only used by the host controller; 0 in device mode.
12 HCH	HC Halted. 1 = Default. This bit is a 0 whenever the Run/Stop bit is a 1. The Host Controller sets this bit to 1 after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error). Only used by the host controller; 0 in device mode.
11 RSVD2	Reserved.
10 ULPII	Not present in this implementation.
9 RSVD1	Reserved.
8 SLI	DC Suspend. 0 = Default. When a device controller enters a suspend state from an active state, this bit will be set to a 1. The device controller clears the bit upon exiting from a suspend state. Only used by the device controller.

HW_USBCTRL_USBSTS field descriptions (continued)

Field	Description
7 SRI	SOF Received. 0 = Default. When the device controller detects a Start Of (micro) Frame, this bit will be set to a 1. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Since the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it. This is a non-EHCI status bit.
6 URI	USB Reset Received. 0 = Default. When the device controller detects a USB Reset and enters the default state, this bit will be set to a 1. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used by the device controller. NOTE: This bit should not normally be used to detect reset during suspend, as this block will normally be clock-gated during that time. Use HW_USBPHY_CTRL_RESUME_IRQ, instead.
5 AAI	Interrupt on Async Advance. 0 = Default. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a 1 to the Interrupt on Async Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. Only used by the host controller.
4 SEI	System Error. This bit is not used in this implementation and will always be set to 0.
3 FRI	Frame List Rollover. The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FHINDEX [12] toggles. Only used by the host controller.
2 PCI	Port Change Detect. The Host Controller sets this bit to a 1 when on any port a Connect Status occurs, a Port Enable/Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a 1 when the port controller enters the full or high-speed operational state. When the port controller exits the full or highspeed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively. This bit is not EHCI compatible.

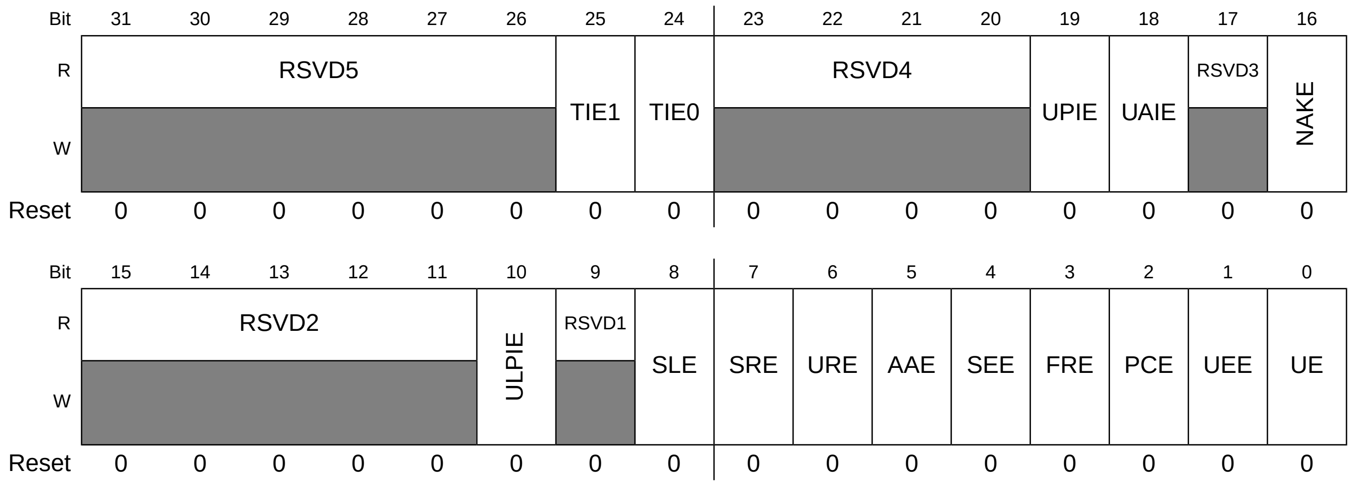
HW_USBCTRL_USBSTS field descriptions (continued)

Field	Description
1 UEI	USB Error Interrupt (USBERRINT). When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. See Section 4.15.1 in the EHCI specification for a complete list of host error interrupt conditions. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT). This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

31.7.19 USB Interrupt Enable Register (HW_USBCTRL_USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt is active. The USB Status register (USBSTS) still shows interrupt sources even if they are disabled by the USBINTR register, allowing polling of interrupt events by the software.

Address: HW_USBCTRL_USBINTR – 8008_0000h base + 148h offset = 8008_0148h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSVD5
TIE1
TIE0
RSVD4
UPIE
UAIE
RSVD3
NAKE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSVD2
ULPIE
RSVD1
SLE
SRE
URE
AAE
SEE
FRE
PCE
UEE
UE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_USBCTRL_USBINTR field descriptions

Field	Description
31–26 RSVD5	Reserved.
25 TIE1	General-Purpose Timer Interrupt Enable 1. When this bit is a 1, and the GPTINT1 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT1 bit.

HW_USBCTRL_USBINTR field descriptions (continued)

Field	Description
24 TIE0	General-Purpose Timer Interrupt Enable 0. When this bit is a 1, and the GPTINT0 bit in the USBSTS register is a 1, the controller will issue an interrupt. The interrupt is acknowledged by software clearing the GPTINT0 bit.
23–20 RSVD4	Reserved.
19 UPIE	USB Host Periodic Interrupt Enable. When this bit is a 1, and the USBHSTPERINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTPERINT bit.
18 UAIE	USB Host Asynchronous Interrupt Enable. RW 0x0 When this bit is a 1, and the USBHSTASYNCINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBHSTASYNCINT bit.
17 RSVD3	Reserved.
16 NAKE	NAK Interrupt Enable. This bit is set by software if it wants to enable the hardware interrupt for the NAK Interrupt bit. If both this bit and the corresponding NAK Interrupt bit are set, a hardware interrupt is generated.
15–11 RSVD2	Reserved.
10 ULPIE	ULPI Enable. Not used in this implementation.
9 RSVD1	Reserved.
8 SLE	Sleep Enable. When this bit is a 1, and the DCSuspend bit in the USBSTS register transitions, the device controller will issue an interrupt. The interrupt is acknowledged by software writing a 1 to the DCSuspend bit. Only used by the device controller.
7 SRE	SOF Received Enable. When this bit is a 1, and the SOF Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the SOF Received bit.
6 URE	USB Reset Enable. When this bit is a 1, and the USB Reset Received bit in the USBSTS register is a 1, the device controller will issue an interrupt. The interrupt is acknowledged by software clearing the USB Reset Received bit. Only used by the device controller.
5 AAE	Interrupt on Async Advance Enable. When this bit is a 1, and the Interrupt on Async Advance bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Async Advance bit. Only used by the host controller.

HW_USBCTRL_USBINTR field descriptions (continued)

Field	Description
4 SEE	System Error Enable. When this bit is a 1, and the System Error bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the System Error bit.
3 FRE	Frame List Rollover Enable. When this bit is a 1, and the Frame List Rollover bit in the USBSTS register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. Only used by the host controller.
2 PCE	Port Change Detect Enable. When this bit is a 1, and the Port Change Detect bit in the USBSTS register is a 1, the host/device controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit.
1 UEE	USB Error Interrupt Enable. When this bit is a 1, and the USBERRINT bit in the USBSTS register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBERRINT bit in the USBSTS register.
0 UE	USB Interrupt Enable. When this bit is a 1, and the USBINT bit in the USBSTS register is a 1, the host/device controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit.

31.7.20 USB Frame Index Register (HW_USBCTRL_FRINDEX)

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the USBCMD register. This register must be written as a DWord. Byte writes produce-undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop bit is set to a 1 produces undefined results. Writes to this register also affect the SOF value. In device mode this register is Read-Only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to 0 (i.e., SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be incremented (i.e., SOF for 125 us micro-frame.) * The default value of this register is undefined (free-running counter).

Address: HW_USBCTRL_FRINDEX – 8008_0000h base + 14Ch offset = 8008_014Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																		FRINDEX											UINDEX		
W	▨																		▨													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_FRINDEX field descriptions

Field	Description
31–14 RSVD	Reserved.
13–3 FRINDEX	Frame List Current Index. Read/write in host mode. Read in device mode. The value in this register increments at the end of each time frame (e.g., micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. In device mode, the value is the current frame number of the last frame transmitted. It is not used as an index. 12 **N_12** — FRAME LIST SIZE = 1024, USBCMD = 3'b000. 11 **N_11** — FRAME LIST SIZE = 512, USBCMD = 3'b001. 10 **N_10** — FRAME LIST SIZE = 256, USBCMD = 3'b010. 9 **N_9** — FRAME LIST SIZE = 128, USBCMD = 3'b011. 8 **N_8** — FRAME LIST SIZE = 64, USBCMD = 3'b100. 7 **N_7** — FRAME LIST SIZE = 32, USBCMD = 3'b101. 6 **N_6** — FRAME LIST SIZE = 16, USBCMD = 3'b110. 5 **N_5** — FRAME LIST SIZE = 8, USBCMD = 3'b111.
2–0 UINDEX	Current Microframe.

31.7.21 Frame List Base Address Register (Host Controller mode) (HW_USBCTRL_PERIODICLISTBASE)

In Host Controller mode, this 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence. This is a read/write register. Writes must be DWORD writes.

Address: HW_USBCTRL_PERIODICLISTBASE – 8008_0000h base + 154h offset = 8008_0154h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PERBASE																				RSVD											
W																					▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_PERIODICLISTBASE field descriptions

Field	Description
31–12 PERBASE	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
11–0 RSVD	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

31.7.22 USB Device Address Register (Device Controller mode) (HW_USBCTRL_DEVICEADDR)

In Device Controller mode, the upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor. The USBADRA is used to accelerate the SET_ADDRESS sequence by allowing the DCD to preset the USBADR register before the status phase of the SET_ADDRESS descriptor. This is a read/write register. Writes must be DWORD writes.

Address: HW_USBCTRL_DEVICEADDR – 8008_0000h base + 154h offset = 8008_0154h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	USBADR							USBADRA	RSVD[23:16]							
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:0]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address.
24 USBADRA	Device Address Advance. Default=0. When this bit is `0', any writes to USBADR are instantaneous. When this bit is written to a 1 at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1. IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2. OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3. Device Reset occurs (USBADR is reset to 0). Note: After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD cannot write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
23–0 RSVD	Reserved. Must be written as zeros. During runtime, the values of these bits are undefined.

31.7.23 Next Asynchronous Address Register (Host Controller mode) (HW_USBCTRL_ASYNCLISTADDR)

In Host Controller mode, this 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a 0 when read.

Address: HW_USBCTRL_ASYNCLISTADDR – 8008_0000h base + 158h offset = 8008_0158h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ASYBASE																											RSVD				
W																												▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (OH). Only used by the host controller.

HW_USBCTRL_ASYNCLISTADDR field descriptions (continued)

Field	Description
4–0 RSVD	Reserved. These bits are reserved and their value has no effect on operation.

31.7.24 Endpoint List Address Register (Device Controller mode) (HW_USBCTRL_ENDPOINTLISTADDR)

In Device Controller mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a 0 when read. The memory structure referenced by this physical memory pointer is assumed 64-byte. This is a read/write register. Writes must be DWORD writes.

Address: HW_USBCTRL_ENDPOINTLISTADDR – 8008_0000h base + 158h offset = 8008_0158h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EPBASE																					RSVD										
W																						▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPOINTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer (Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Heads (QH). (i.e., one queue head per endpoint and direction.)
10–0 RSVD	Reserved. These bits are reserved and their value has no effect on operation.

31.7.25 Embedded TT Asynchronous Buffer Status and Control Register (Host Controller mode) (HW_USBCTRL_TTCTRL)

This register contains parameters needed for internal TT operations. This register is not used in the device controller operation. This is a read/write register. Writes must be DWORD writes.

Address: HW_USBCTRL_TTCTRL – 8008_0000h base + 15Ch offset = 8008_015Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1	TTHA							RSVD2[23:16]							
W	▨	TTHA							▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2[15:0]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_TTCTRL field descriptions

Field	Description
31 RSVD1	Reserved. These bits are reserved and their value has no effect on operation.
30–24 TTHA	Internal TT Hub Address Representation. Default is 0 (Read/Write). This field is used to match against the Hub Address field in QH and siTD to determine if the packet is routed to the internal TT for directly attached FS/LS devices. If the Hub Address in the QH or siTD does not match this address then the packet will be broadcast on the High Speed ports destined for a downstream High Speed hub with the address in the QH/siTD.
23–0 RSVD2	Reserved. These bits are reserved and their value has no effect on operation.

31.7.26 Programmable Burst Size Register (HW_USBCTRL_BURSTSIZE)

This register is used to control dynamically change the burst size used during data movement on the initiator (master) interface. This is a read/write register. Writes must be DWORD writes. The default value is 0x00001010.

Address: HW_USBCTRL_BURSTSIZE – 8008_0000h base + 160h offset = 8008_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																TXPBURST								RXPBURST							
W	▨																TXPBURST								RXPBURST							
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0

HW_USBCTRL_BURSTSIZE field descriptions

Field	Description
31–16 RSVD	Reserved. These bits are reserved and their value has no effect on operation.
15–8 TXPBURST	Programmable TX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
7–0 RXPBURST	Programmable RX Burst Length. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

31.7.27 Host Transmit Pre-Buffer Packet Timing Register (HW_USBCTRL_TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system. Definitions: T0 = Standard packet overhead T1 = Time to send data payload Tff = Time to fetch packet into TX FIFO up to specified level. Ts = Total Packet Flight Time (send-only) packet Ts = T0 + T1 Tp = Total Packet Time (fetch and send) packet Tp = Tff + T0 + T1 Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure Tp remains before the end of the (micro)frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the (micro)frame is < Ts then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the TSCHHEALTH (Tff) described below. This is a read/write register. Writes must be DWORD writes. The default value of this register is 0x00000000.

Address: HW_USBCTRL_TXFILLTUNING – 8008_0000h base + 164h offset = 8008_0164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2										TXFIFOTHRES					
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1			TXSCHEALTH					RSVD0	TXSCHOH						
W	▨	▨	▨						▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_TXFILLTUNING field descriptions

Field	Description
31–22 RSVD2	Reserved. These bits are reserved and their value has no effect on operation.
21–16 TXFIFOTHRES	FIFO Burst Threshold. When S/W changes the USBMODE.CM to Host(11), the field defaults to 2. This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USBMODE register is set.
15–13 RSVD1	Reserved. These bits are reserved and their value has no effect on operation.
12–8 TXSCHEALTH	Scheduler Health Counter. This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31.
7 RSVD0	Reserved. This bit is reserved and its value has no effect on operation.
6–0 TXSCHOH	Scheduler Overhead. This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode for OTG & SPH.

HW_USBCTRL_TXFILLTUNING field descriptions (continued)

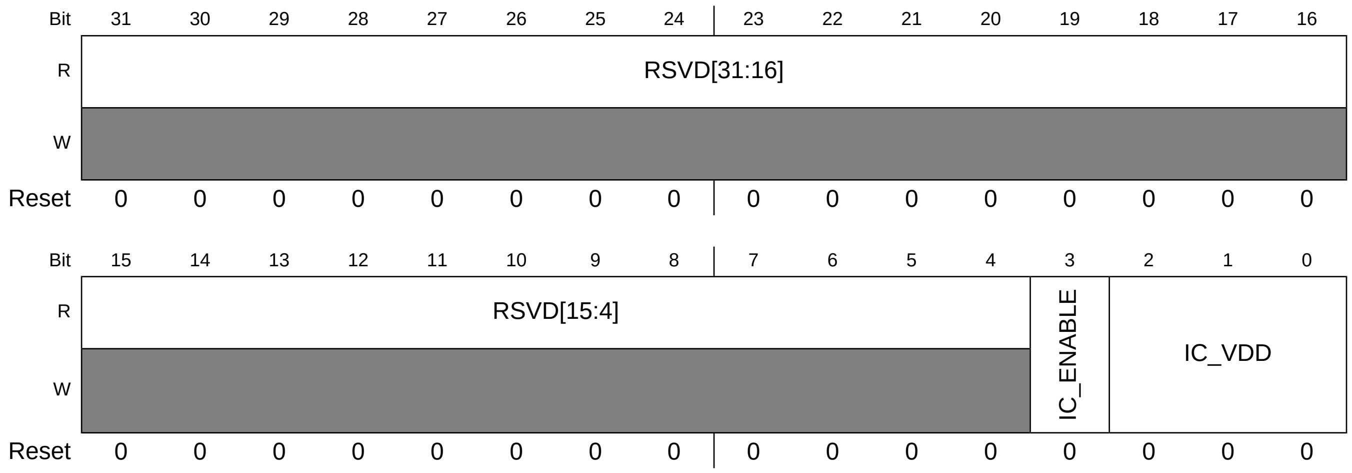
Field	Description
	The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode for OTG & SPH. The time unit represented in this register is always 1.267 in the MPH product.

31.7.28 Inter-Chip Control Register (HW_USBCTRL_IC_USB)

This register is present but not used in this implementation.

This register enables and controls the IC_USB FS/LS transceiver.

Address: HW_USBCTRL_IC_USB – 8008_0000h base + 16Ch offset = 8008_016Ch



HW_USBCTRL_IC_USB field descriptions

Field	Description
31–4 RSVD	Reserved.
3 IC_ENABLE	Inter-Chip Transceiver Enable. These bits enables the InterChip transceiver for each port (for the MPH case). To enable the interface, the bits PTS must be set to 0b11 in the PORTSCx. Writing a '1' to each bit selects the IC_USB interface for that port. If the Controller is not MultiPort, IC8 to IC2 will be '0' and Read-Only.
2–0 IC_VDD	Inter-Chip Transceiver Voltage. Selects the voltage being supplied to the peripheral through each port (MPH case). 0x0 **VOLTAGE_NONE** — . 0x1 **VOLTAGE_1_0** — . 0x2 **VOLTAGE_1_2** — . 0x3 **VOLTAGE_1_5** — . 0x4 **VOLTAGE_1_8** — . 0x5 **VOLTAGE_3_0** — . 0x6 **RESERVED0** — .

HW_USBCTRL_IC_USB field descriptions (continued)

Field	Description
	0x7 **RESERVED1** — .

31.7.29 ULPI Viewport Register (HW_USBCTRL_ULPI)

This register is present but not used in this implementation.

Address: HW_USBCTRL_ULPI – 8008_0000h base + 170h offset = 8008_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ULPIWU	ULPIRUN	ULPIRW	RSVD0	ULPISS	ULPIPORT			ULPIADDR							
W				▨	▨											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ULPIDATRD								ULPIDATWR							
W	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ULPI field descriptions

Field	Description
31 ULPIWU	Not used. Read as 0.
30 ULPIRUN	Not used. Read as 0.
29 ULPIRW	Not used. Read as 0.
28 RSVD0	Not used. Read as 0.
27 ULPISS	Not used. Read as 0.
26–24 ULPIPORT	Not used. Read as 0.
23–16 ULPIADDR	Not used. Read as 0.
15–8 ULPIDATRD	Not used. Read as 0.
7–0 ULPIDATWR	Not used. Read as 0.

31.7.30 Endpoint NAK Register (HW_USBCTRL_ENDPTNAK)

NAK-sent indicator

Address: HW_USBCTRL_ENDPTNAK – 8008_0000h base + 178h offset = 8008_0178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								EPTN								RSVD0								EPRN							
W	▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTNAK field descriptions

Field	Description
31–24 RSVD1	Reserved.

23–16
EPTN
	TX Endpoint NAK.
	Each TX endpoint has one bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint.
	EPTN[7] = Endpoint 7
	EPTN[6] = Endpoint 6
	EPTN[5] = Endpoint 5
	EPTN[4] = Endpoint 4
	EPTN[3] = Endpoint 3
	EPTN[2] = Endpoint 2
	EPTN[1] = Endpoint 1
	EPTN[0] = Endpoint 0

15–8 RSVD0	Reserved.

7–0
EPRN
	RX Endpoint NAK.
	Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint.
	EPRN[7] = Endpoint 7
	EPRN[6] = Endpoint 6
	EPRN[5] = Endpoint 5
	EPRN[4] = Endpoint 4
	EPRN[3] = Endpoint 3
	EPRN[2] = Endpoint 2
	EPRN[1] = Endpoint 1
	EPRN[0] = Endpoint 0

31.7.31 Endpoint NAK Enable Register (HW_USBCTRL_ENDPTNAKEN)

NAK-sent indicator enable

Address: HW_USBCTRL_ENDPTNAKEN – 8008_0000h base + 17Ch offset = 8008_017Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								EPTNE								RSVD0								EPRNE							
W	▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTNAKEN field descriptions

Field	Description
31–24 RSVD1	Reserved.

23–16
EPTNE
	TX Endpoint NAK Enable.
	Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set.
	EPTNE[7] = Endpoint 7
	EPTNE[6] = Endpoint 6
	EPTNE[5] = Endpoint 5
	EPTNE[4] = Endpoint 4
	EPTNE[3] = Endpoint 3
	EPTNE[2] = Endpoint 2
	EPTNE[1] = Endpoint 1
	EPTNE[0] = Endpoint 0

15–8 RSVD0	Reserved.

7–0
EPRNE
	RX Endpoint NAK Enable.
	Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set.
	EPRNE[7] = Endpoint 7
	EPRNE[6] = Endpoint 6
	EPRNE[5] = Endpoint 5
	EPRNE[4] = Endpoint 4
	EPRNE[3] = Endpoint 3
	EPRNE[2] = Endpoint 2
	EPRNE[1] = Endpoint 1
	EPRNE[0] = Endpoint 0

31.7.32 Port Status and Control 1 Register (HW_USBCTRL_PORTSC1)

Host Controller: A host controller must implement one to eight port registers. The number of port registers implemented by a particular instantiation of a host controller is documented in the HCSPARAMs register and is fixed at 1 in this implementation. Software uses this information as an input parameter to determine how many ports need service. This register is only reset when power is initially applied or in response to a controller reset. The initial conditions of a port are: - No device connected - Port disabled If the port has port power control, this state remains until software applies power to the port by setting port power to 1. Device Controller: A device controller must implement only port register 1 and it does not support power control. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock. * Default Value: 00010000000000000000XX0000000000b (Host mode) 00010000000000000001XX0000000100b (Device mode) X = Unknown

Address: HW_USBCTRL_PORTSC1 – 8008_0000h base + 184h offset = 8008_0184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PTS		STS	PTW	PSPD		PTS2	PFSC	PHCD	WKOC	WKDS	WKCN	PTC			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_PORTSC1 field descriptions

Field	Description
31–30 PTS	Parallel Transceiver Select. For this implementation, always set to 00b for UTMI. Note that this field is made up from PORTSCx bits 25, 30 and 31. 0 **UTMI** — UTMI/UTMI+. 1 **PHIL** — Phillips-Classic. 2 **ULPI** — ULPI. 3 **SERIAL** — Serial/1.1FS. 4 **HSIC** — UTMI for HSIC.
29 STS	Serial Transceiver Select. Always 0.
28 PTW	Parallel Transceiver Width. This bit is always 1, indicating an 16-bit (30-MHz) UTMI interface.
27–26 PSPD	Port Speed. This register field indicates the speed at which the port is operating. For high-speed mode operation in the host controller and high-speed/fullspeed operation in the device controller, the port routing steers data to the protocol engine.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
	This bit is not defined in the EHCI specification. 0 **FULL** — Full Speed. 1 **LOW** — Low Speed. 2 **HIGH** — High Speed.
25 PTS2	Parallel Transceiver Select,bit2. See PTS bits for details
24 PFSC	Port Force Full Speed Connect. Default = 0. Writing this bit to a 1 will force the port to only connect at Full Speed. It disables the chirp sequence that allows the port to identify itself as high-speed. This is useful for testing full-speed configurations with a high-speed host, hub or device. This bit is not defined in the EHCI specification. This bit is for debugging purposes.
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD). Default = 0. Writing this bit to a 1 will disable the PHY clock. Writing a 0 enables it. Reading this bit will indicate the status of the PHY clock. In Device Mode: The PHY can be put into Low Power Suspend running (USBCMD Run/Stop=0) or the host has signaled suspend (PORTSC SUSPEND=1). Lowpower suspend will be cleared automatically when the host has signaled resume. Before forcing a resume from the device, the device controller driver must clear this bit. In Host Mode: The PHY can be put into Low Power Suspend device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software. This bit is not defined in the EHCI specification.
22 WKOC	Wake on Over-current Enable (WKOC_E). Default = 0. Writing this bit to a 1 enables the port to be sensitive to over-current conditions as wake-up events. This field is 0 if Port Power (PP) is 0.
21 WKDS	Wake on Disconnect Enable (WKDSCNNT_E). Default=0. Writing this bit to a 1 enables the port to be sensitive to device disconnects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E). Default=0. Writing this bit to a 1 enables the port to be sensitive to device connects as wake-up events. This field is 0 if Port Power (PP) is 0 or in device mode.
19–16 PTC	Port Test Control. Default = 0000b.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
	Any other value than 0 indicates that the port is operating in test mode. Refer to Chapter 9 of the USB Specification Revision 2.0 for details on each test mode. The TEST_FORCE_ENABLE_FS and TEST_FORCE_ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the TEST_FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_DISABLE will allow the port state machines to progress normally from that point. Note: Low speed operations are not supported. 0 **TEST_DISABLE** — Disable. 1 **TEST_J_STATE** — J-State. 2 **TEST_K_STATE** — K-State. 3 **TEST_J_SE0_NAK** — Host:SE0/Dev:NAK. 4 **TEST_PACKET** — Test-Packet. 5 **TEST_FORCE_ENABLE_HS** — Force-Enable-HS. 6 **TEST_FORCE_ENABLE_FS** — Force-Enable-FS. 7 **TEST_FORCE_ENABLE_LS** — Force-Enable-LS.
15–14 PIC	Port Indicator Control. Default = 0. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. 0 **OFF** — OFF. 1 **AMBER** — Amber. 2 **GREEN** — Green. 3 **UNDEF** — undefined.
13 PO	Port Owner. Port owner handoff is not implemented in this design, therefore this bit will always read back as 0. The EHCI definition is include here for reference: Default = 0. This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is 0. System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a 1 to this bit when the attached device is not a high-speed device. A 1 in this bit means that an internal companion controller owns and controls the port.
12 PP	Port Power (PP). This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (i.e., PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an over-current condition is detected on a powered port, the PP bit in each affected port may be transitioned by the host controller driver from a 1 to a 0 (removing power from the port). This feature is implemented in the host/OTG controller (PPC = 1). In a device implementation, port power control is not necessary.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
11–10 LS	Line Status. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. The bit encodings are listed below. In Host Mode: The use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In Device Mode: The use of linestate by the device controller driver is not necessary. 0 **SE0** — SE0. 1 **K_STATE** — K. 2 **J_STATE** — J. 3 **UNDEF** — Undefined.
9 HSP	High-Speed Port. Default = 0. When the bit is 1, the host/device connected to the port is in high-speed mode and if set to 0, the host/device connected to the port is not in a high-speed mode. Note: HSP is redundant with PSPD(27:26) but will remain in the design for compatibility. This bit is not defined in the EHCI specification.
8 PR	Port Reset This field is 0 if Port Power (PP) is 0. In Host Mode: (Read/Write). 1 = Port is in Reset. 0 = Port is not in Reset. Default 0. When software writes a 1 to this bit, the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. This bit will automatically change to 0 after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the reset duration is timed in the driver. In Device Mode: This bit is a Read-Only status bit. Device reset from the USB bus is also indicated in the USBSTS register.
7 SUSP	Suspend In Host Mode: (Read/Write) 0 = Port not in suspend state. 1 = Port in suspend state. Default = 0. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits Port State 0x Disable 10 Enable 11 Suspend When in suspend state, the downstream propagation of data is blocked on this port, except for port reset.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
	The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to 0 when software sets the Force Port Resume bit to 0. The host controller ignores a write of 0 to this bit. If host software sets this bit to a 1 when the port is not enabled (i.e., Port enabled bit is a 0) the results are undefined. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: (Read-Only) 1 = Port in suspend state. 0 = Port not in suspend state. Default=0. In device mode, this bit is a Read-Only status bit.
6 FPR	Force Port Resume. 0 = No resume (K-state) detected/driven on port. 1 = Resume detected/driven on port. Default = 0. In Host Mode: Software sets this bit to 1 to drive resume signaling. The Host Controller sets this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a 1 because a J-to-K transition is detected, the Port Change Detect bit in the USBSTS register is also set to 1. This bit will automatically change to 0 after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a 0 after the resume duration is timed in the driver. Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a 1. This bit will remain a 1 until the port has switched to the high-speed idle. Writing a 0 has no effect because the port controller will time the resume operation and clear the bit when the port control state switches to HS or FS idle. This field is 0 if Port Power (PP) is 0 in host mode. This bit is not-EHCI compatible. In Device Mode: After the device has been in Suspend State for 5 ms or more, software must set this bit to 1 to drive resume signaling before clearing. The Device Controller will set this bit to 1 if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit transitions to a 1 because a J-to-K transition has been detected, the Port Change Detect bit in the USBSTS register is also set to 1.
5 OCC	Over-Current Change. 0 = Default. 1 = This bit gets set to 1 when there is a change to Over-Current Active. Software clears this bit by writing a 1 to this bit position.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
	For host/OTG implementations, the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations, this bit shall always be 0.
4 OCA	Over-Current Active. 0 = This port does not have an over-current condition. 1 = This port currently has an over-current condition. Default = 0. This bit will automatically transition from 1 to 0 when the over current condition is removed. For host/OTG implementations the user can provide over-current detection to the vbus_pwr_fault input for this condition. For device-only implementations this bit shall always be 0.
3 PEC	Port Enable/Disable Change. 0 = No change. 1 = Port enabled/disabled status has changed. Default = 0. In Host Mode: For the root hub, this bit gets set to a 1 only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a 1 to it. This field is 0 if Port Power (PP) is 0. In Device Mode: The device port is always enabled. (This bit will be 0)
2 PE	Port Enabled/Disabled. 0 = Disable. 1 = Enable. Default = 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a 1 to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0) downstream propagation of data is blocked except for reset. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: The device port is always enabled. (This bit will be 1)
1 CSC	Connect Status Change. 0 = No change. 1 = Change in Current Connect Status.

HW_USBCTRL_PORTSC1 field descriptions (continued)

Field	Description
	Default = 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (i.e., the bit will remain set). Software clears this bit by writing a 1 to it. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: This bit is undefined in device controller mode.
0 CCS	Current Connect Status. In Host Mode: 0 = No device is present. 1 = Device is present on port. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is 0 if Port Power (PP) is 0 in host mode. In Device Mode: 0 = Not Attached. 1 = Attached. Default = 0. A 1 indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A 0 indicates that the device did not attach successfully or was forcibly disconnected by the software writing a 0 to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

31.7.33 OTG Status and Control Register (HW_USBCTRL_OTGSC)

Host Controller: A host controller implements one On-The-Go (OTG) Status and Control register corresponding to Port 0 of the host controller. The OTGSC register has four sections: OTG Interrupt Enables (Read/Write) OTG Interrupt Status (Read/Write to Clear) OTG Status Inputs (Read-Only) OTG Controls (Read/Write) The status inputs are debounced using a 1-ms time constant. Values on the status inputs that do not persist for more than 1 ms will not cause an update of the status input register, or cause an OTG interrupt. See also USBMODE register. The default value of this register is 0x00000120.

Address: HW_USBCTRL_OTGSC – 8008_0000h base + 1A4h offset = 8008_01A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	DPIE	ONEMSE	BSEIE	BSVIE	ASVIE	AVVIE	IDIE	RSVD1	DPIS	ONEMSS	BSEIS	BSVIS	ASVIS	AVVIS	IDIS
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0	DPS	ONEMST	BSE	BSV	ASV	AVV	ID	HABA	HADP	IDPU	DP	OT	HAAR	VC	VD
W	▨			▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0

HW_USBCTRL_OTGSC field descriptions

Field	Description
31 RSVD2	Reserved.
30 DPIE	Data Pulse Interrupt Enable
29 ONEMSE	1 Millisecond Timer Interrupt Enable
28 BSEIE	B Session End Interrupt Enable. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable. Setting this bit enables the USB ID interrupt.
23 RSVD1	Reserved.
22 DPIS	Data Pulse Interrupt Status. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC(0).PortPower = Off (0). Software must write a 1 to clear this bit.
21 ONEMSS	1 Millisecond Timer Interrupt Status. This bit is set once every millisecond. Software must write a 1 to clear this bit.

HW_USBCTRL_OTGSC field descriptions (continued)

Field	Description
20 BSEIS	B Session End Interrupt Status. This bit is set when VBus has fallen below the B session end threshold. Software must write a 1 to clear this bit.
19 BSVIS	B Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the B session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
18 ASVIS	A Session Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the A session valid threshold (0.8 VDC). Software must write a 1 to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status. This bit is set when VBus has either risen above or fallen below the VBus valid threshold (4.4 VDC) on an A device. Software must write a 1 to clear this bit.
16 IDIS	USB ID Interrupt Status. This bit is set when a change on the ID input has been detected. Software must write a 1 to clear this bit.
15 RSVD0	Reserved.
14 DPS	Data Bus Pulsing Status. A 1 indicates data bus pulsing is being detected on the port.
13 ONEMST	1 Millisecond Timer Toggle. This bit toggles once per millisecond.
12 BSE	B Session End. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID. 0 = A device. 1 = B device.
7 HABA	Hardware Assist B-Disconnect to A-connect. 0 = Disabled. 1 = Enable automatic B-disconnect to A-connect sequence.

HW_USBCTRL_OTGSC field descriptions (continued)

Field	Description
6 HADP	Hardware Assist Data-Pulse 1 = Start Data Pulse Sequence.
5 IDPU	ID Pullup. This bit provide control over the ID pullup resister. 0 = off. 1 = on (default). When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 HAAR	Hardware Assist Auto-Reset. 0 = Disabled. 1 = Enable automatic reset after connect on host port.
1 VC	VBUS Charge. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge. Setting this bit causes VBus to discharge through a resistor.

31.7.34 USB Device Mode Register (HW_USBCTRL_USBMODE)

Default Value:0x00000000 (implementation OTGmode not selected).

Address: HW_USBCTRL_USBMODE – 8008_0000h base + 1A8h offset = 8008_01A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1																SRT	RSVD0									VBPS	SDIS	SLOM	ES	CM	
W																	SRT										VBPS	SDIS	SLOM	ES	CM	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_USBMODE field descriptions

Field	Description
31–16 RSVD1	Reserved.

HW_USBCTRL_USBMODE field descriptions (continued)

Field	Description
15 SRT	Short Reset Time.
14–6 RSVD0	Reserved
5 VBPS	Vbus Power Select 0 = Output is 0. 1 = Output is 1. This bit is connected to the vbus_pwr_select output and can be used for any generic control but is named to be used by logic that selects between an on-chip Vbus power source (charge pump) and an off-chip source in systems when both are available.
4 SDIS	Stream Disable Mode. 0 = Inactive (default). 1 = Active. In Device Mode: Setting to a 1 disables double priming on both RX and TX for low bandwidth systems. This mode, when enabled, ensures that the RX and TX buffers are sufficient to contain an entire packet, so that the usual double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received will be responded to with a NYET handshake when stream disable is active. In Host Mode: Setting to a 1 ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. Note: Time duration to pre-fill the FIFO becomes significative when stream disable is active. See TXFILLTUNING and TXTTFILLTUNING to characterize the adjustments needed for the scheduler when using this feature. Note: The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. 0 = Setup Lockouts On (default). 1 = Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USBCMD).
2 ES	Endian Select. This bit can change the byte ordering of the transfer buffers to match the host microprocessor bus architecture. The bit fields in the microprocessor interface and the DMA data structures (including the setup buffer within the device QH) are unaffected by the value of this bit, because they are based upon 32-bit words. 0 = Little Endian (default): First byte referenced in least significant byte of 32-bit word. 1 = Big Endian: First byte referenced in most significant byte of 32-bit word.
1–0 CM	Controller Mode.

HW_USBCTRL_USBMODE field descriptions (continued)

Field	Description
	Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller will default to an idle state and will need to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the RESET bit in the USBCMD register before reprogramming this register. 0x0 **IDLE** — IDLE. 0x2 **DEVICE** — DEVICE. 0x3 **HOST** — HOST.

31.7.35 Endpoint Setup Status Register (HW_USBCTRL_ENDPTSETUPSTAT)

endpoint setup status

Address: HW_USBCTRL_ENDPTSETUPSTAT – 8008_0000h base + 1ACh offset = 8008_01ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																								ENDPTSETUPSTAT							
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTSETUPSTAT field descriptions

Field	Description
31–8 RSVD	Reserved.
7–0 ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to 1. Software must clear or acknowledge the setup transfer by writing a 1 to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock out mechanism is engaged. This register is only used in device mode.

31.7.36 Endpoint Initialization Register (HW_USBCTRL_ENDPTPRIME)

This register is used in device mode only.

endpoint prime request

Address: HW_USBCTRL_ENDPTPRIME – 8008_0000h base + 1B0h offset = 8008_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PETB								RSVD0								PERB							
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTPRIME field descriptions

Field	Description
31–24 RSVD1	Reserved.

23–16
PETB
	Prime Endpoint Transmit Buffer.
	
	For each endpoint, a corresponding bit is used to request that a buffer be prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a 1 to the corresponding bit when posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.
	
	Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
	
	PETB[7] = Endpoint 7.
	
	PETB[6] = Endpoint 6.
	
	PETB[5] = Endpoint 5.
	
	PETB[4] = Endpoint 4.
	
	PETB[3] = Endpoint 3.
	
	PETB[2] = Endpoint 2.
	
	PETB[1] = Endpoint 1.
	
	PETB[0] = Endpoint 0.

15–8 RSVD0	Reserved.

7–0
PERB
	Prime Endpoint Receive Buffer.
	
	For each endpoint, a corresponding bit is used to request a buffer be prepared for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a 1 to the corresponding bit whenever posting a new transfer descriptor to an endpoint. Hardware will automatically use this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware will clear this bit when the associated endpoint(s) is (are) successfully primed.
	
	Note: These bits will be momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated.
	
	PERB[7] = Endpoint 7.
	
	PERB[6] = Endpoint 6.
	
	PERB[5] = Endpoint 5.
	
	PERB[4] = Endpoint 4.
	
	PERB[3] = Endpoint 3.
	
	PERB[2] = Endpoint 2.
	
	PERB[1] = Endpoint 1.

HW_USBCTRL_ENDPTPRIME field descriptions (continued)

Field	Description
	PERB[0] = Endpoint 0.

31.7.37 Endpoint De-Initialize Register (HW_USBCTRL_ENDPTFLUSH)

This register is used in device-mode only.

endpoint flush request

Address: HW_USBCTRL_ENDPTFLUSH – 8008_0000h base + 1B4h offset = 8008_01B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								FETB								RSVD0								FERB							
W	▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTFLUSH field descriptions

Field	Description
31–24 RSVD1	Reserved.

23–16
FETB
	Flush Endpoint Transmit Buffer.
	Writing a 1 to a bit(s) in this register will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for 1 of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful.
	FETB[7] = Endpoint 7.
	FETB[6] = Endpoint 6.
	FETB[5] = Endpoint 5.
	FETB[4] = Endpoint 4.
	FETB[3] = Endpoint 3.
	FETB[2] = Endpoint 2.
	FETB[1] = Endpoint 1.
	FETB[0] = Endpoint 0.

15–8 RSVD0	Reserved.

7–0
FERB
	Flush Endpoint Receive Buffer.
	Writing a 1 to a bit(s) will cause the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer will continue until completion. Hardware will clear this register after the endpoint flush operation is successful.
	FERB[7] = Endpoint 7.
	FERB[6] = Endpoint 6.

HW_USBCTRL_ENDPTFLUSH field descriptions (continued)

Field	Description

	FERB[5] = Endpoint 5.
	FERB[4] = Endpoint 4.
	FERB[3] = Endpoint 3.
	FERB[2] = Endpoint 2.
	FERB[1] = Endpoint 1.
	FERB[0] = Endpoint 0.

31.7.38 Endpoint Status Register (HW_USBCTRL_ENDPTSTAT)

This register is used in device mode only.

endpoint ready

Address: HW_USBCTRL_ENDPTSTAT – 8008_0000h base + 1B8h offset = 8008_01B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								ETBR								RSVD0								ERBR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTSTAT field descriptions

Field	Description
31–24 RSVD1	Reserved.

23–16
ETBR
	Endpoint Transmit Buffer Ready.
	One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.
	Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.
	ETBR[7] = Endpoint 7.
	ETBR[6] = Endpoint 6.
	ETBR[5] = Endpoint 5.
	ETBR[4] = Endpoint 4.
	ETBR[3] = Endpoint 3.
	ETBR[2] = Endpoint 2.
	ETBR[1] = Endpoint 1.

HW_USBCTRL_ENDPTSTAT field descriptions (continued)

Field	Description
	ETBR[0] = Endpoint 0.
15–8 RSVD0	Reserved.

7–0
ERBR
	Endpoint Receive Buffer Ready.
	
	One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a 1 by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There will always be a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPTPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register.
	
	Note: These bits will be momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated.
	
	ERBR[7] = Endpoint 7.
	
	ERBR[6] = Endpoint 6.
	
	ERBR[5] = Endpoint 5.
	
	ERBR[4] = Endpoint 4.
	
	ERBR[3] = Endpoint 3.
	
	ERBR[2] = Endpoint 2.
	
	ERBR[1] = Endpoint 1.
	
	ERBR[0] = Endpoint 0.

31.7.39 Endpoint Complete Register (HW_USBCTRL_ENDPTCOMPLETE)

This register is used in device-mode only.

endpoint complete

Address: HW_USBCTRL_ENDPTCOMPLETE – 8008_0000h base + 1BCh offset = 8008_01BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								ETCE								RSVD0								ERCE							
W	▨	▨	▨	▨	▨	▨	▨	▨									▨	▨	▨	▨	▨	▨	▨	▨								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCOMPLETE field descriptions

Field	Description
31–24 RSVD1	Reserved.

HW_USBCTRL_ENDPTCOMPLETE field descriptions (continued)

Field	Description

23–16
ETCE
	Endpoint Transmit Complete Event.
	
	Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register.
	
	ETCE[7] = Endpoint 7.
	
	ETCE[6] = Endpoint 6.
	
	ETCE[5] = Endpoint 5.
	
	ETCE[4] = Endpoint 4.
	
	ETCE[3] = Endpoint 3.
	
	ETCE[2] = Endpoint 2.
	
	ETCE[1] = Endpoint 1.
	
	ETCE[0] = Endpoint 0.

15–8 RSVD0	Reserved.

7–0
ERCE
	Endpoint Receive Complete Event.
	
	Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit will be set simultaneously with the USBINT. Writing a 1 will clear the corresponding bit in this register.
	
	ERCE[7] = Endpoint 7.
	
	ERCE[6] = Endpoint 6.
	
	ERCE[5] = Endpoint 5.
	
	ERCE[4] = Endpoint 4.
	
	ERCE[3] = Endpoint 3.
	
	ERCE[2] = Endpoint 2.
	
	ERCE[1] = Endpoint 1.
	
	ERCE[0] = Endpoint 0.

31.7.40 Endpoint Control 0 Register (HW_USBCTRL_ENDPTCTRL0)

Every Device will implement Endpoint0 as a control endpoint. The default value of this register is 0x00800080.

Address: HW_USBCTRL_ENDPTCTRL0 – 8008_0000h base + 1C0h offset = 8008_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	RSVD5			TXT		RSVD4	TXS
W	▨	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨			▨	
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RSVD2			RXT		RSVD1	RXS
W	▨	▨	▨	▨	▨	▨	▨	▨		▨	▨	▨			▨	
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL0 field descriptions

Field	Description
31–24 RSVD6	Reserved.
23 TXE	TX Endpoint Enable. 1 = Enabled. Endpoint0 is always enabled.
22–20 RSVD5	Reserved. Bit reserved and should be read as zeroes.
19–18 TXT	TX Endpoint Transmit Type. Endpoint0 is fixed as a Control endpoint. 0 **CONTROL** — Control.
17 RSVD4	Reserved.
16 TXS	Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.
15–8 RSVD3	Reserved.

HW_USBCTRL_ENDPTCTRL0 field descriptions (continued)

Field	Description
	Bit reserved and should be read as zeroes.
7 RXE	RX Endpoint Enable. 1 = Enabled. Endpoint0 is always enabled.
6–4 RSVD2	Reserved. Bit reserved and should be read as zeroes.
3–2 RXT	RX Endpoint Receive Type. Endpoint0 is fixed as a Control endpoint. 0 **CONTROL** — Control.
1 RSVD1	Reserved.
0 RXS	RX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. It will continue returning STALL until the bit is cleared by software or it will automatically be cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Note: There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

31.7.41 Endpoint Control 1 Register (HW_USBCTRL_ENDPTCTRL1)

Register HW_USBCTRL_ENDPTCTRL1 is the control register for endpoint 1 in a device. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL1 – 8008_0000h base + 1C4h offset = 8008_01C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨	▨	▨	▨	▨	▨	▨	▨				▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W	▨	▨	▨	▨	▨	▨	▨	▨				▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL1 field descriptions

Field	Description
31–24 RSVD6	Reserved.
23 TXE	TX Endpoint Enable. 0 = Disabled (default). 1 = Enabled. An endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset. Write 1 to reset PID sequence. Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.
21 TXI	TX Data Toggle Inhibit. 0 = PID Sequencing Enabled (default). 1 = PID Sequencing Disabled. This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 RSVD5	Reserved.
19–18 TXT	TX Endpoint Transmit Type. 0 **CONTROL** — Control. 1 **ISO** — Isochronous. 2 **BULK** — Bulk. 3 **INT** — Interrupt.
17 TXD	TX Endpoint Data Source. 0 = Dual Port Memory Buffer/DMA Engine (default). Should always be written as 0.

HW_USBCTRL_ENDPTCTRL1 field descriptions (continued)

Field	Description
16 TXS	Endpoint Stall. 0 = Endpoint OK. 1 = Endpoint Stalled. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. Note (control endpoint types only): There is a slight delay (50 clocks max.) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, Should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.
15–8 RSVD3	Reserved.
7 RXE	RX Endpoint Enable. 0 = Disabled (default). 1 = Enabled. An Endpoint should be enabled only after it has been configured.
6 RXR	Data Toggle Reset. Write 1 to reset PID Sequence. Whenever a configuration event is received for this endpoint, software must write a 1 to this bit in order to synchronize the data PIDs between the host and device.
5 RXI	RX Data Toggle Inhibit. 0 = Disabled (default). 1 = Enabled. This bit is only used for test and should always be written as 0. Writing a 1 to this bit will cause this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 RSVD2	Reserved.
3–2 RXT	RX Endpoint Receive Type. 0 **CONTROL** — Control. 1 **ISO** — Isochronous. 2 **BULK** — Bulk. 3 **INT** — Interrupt.
1 RXD	RX Endpoint Data Sink. 0 = Dual Port Memory Buffer/DMA Engine (default). Should always be written as 0.

HW_USBCTRL_ENDPTCTRL1 field descriptions (continued)

Field	Description
0 RXS	RX Endpoint Stall. 0 = Endpoint OK (default). 1 = Endpoint Stalled. This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a 1 to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. Note (control endpoint types only): There is a slight delay (50 clocks maximum) between the ENDPTSETUPSTAT being cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a 1 to it, then follow this procedure: Continually write this stall bit until it is set OR until a new SETUP has been received by checking the associated ENDPTSETUPSTAT bit.

31.7.42 Endpoint Control 2 Register (HW_USBCTRL_ENDPTCTRL2)

Register HW_USBCTRL_ENDPTCTRL2 is the control register for endpoint 2 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL2 – 8008_0000h base + 1C8h offset = 8008_01C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL2 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

31.7.43 Endpoint Control 3 Register (HW_USBCTRL_ENDPTCTRL3)

Register HW_USBCTRL_ENDPTCTRL3 is the control register for endpoint 3 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL3 – 8008_0000h base + 1CCh offset = 8008_01CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL3 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1

HW_USBCTRL_ENDPTCTRL3 field descriptions (continued)

Field	Description
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

31.7.44 Endpoint Control 4 Register (HW_USBCTRL_ENDPTCTRL4)

Register HW_USBCTRL_ENDPTCTRL4 is the control register for endpoint 4 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL4 – 8008_0000h base + 1D0h offset = 8008_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨	▨	▨	▨	▨	▨	▨	▨				▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W	▨	▨	▨	▨	▨	▨	▨	▨				▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL4 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1

HW_USBCTRL_ENDPTCTRL4 field descriptions (continued)

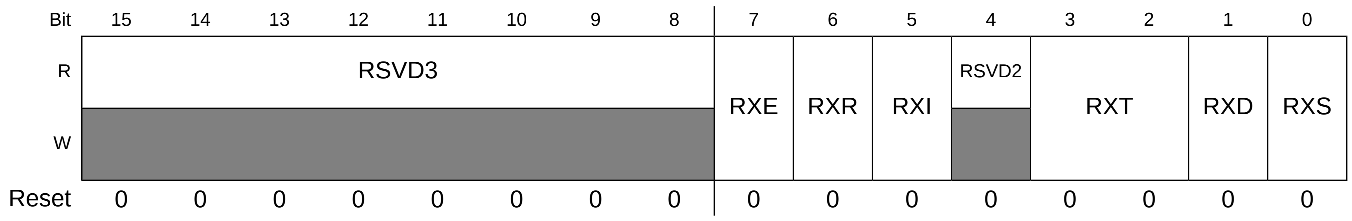
Field	Description
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

31.7.45 Endpoint Control 5 Register (HW_USBCTRL_ENDPTCTRL5)

Register HW_USBCTRL_ENDPTCTRL5 is the control register for endpoint 5 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL5 – 8008_0000h base + 1D4h offset = 8008_01D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨	▨	▨	▨	▨	▨	▨	▨				▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_USBCTRL_ENDPTCTRL5 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

31.7.46 Endpoint Control 6 Register (HW_USBCTRL_ENDPTCTRL6)

Register HW_USBCTRL_ENDPTCTRL6 is the control register for endpoint 6 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL6 – 8008_0000h base + 1D8h offset = 8008_01D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL6 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1

HW_USBCTRL_ENDPTCTRL6 field descriptions (continued)

Field	Description
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

31.7.47 Endpoint Control 7 Register (HW_USBCTRL_ENDPTCTRL7)

Register HW_USBCTRL_ENDPTCTRL7 is the control register for endpoint 7 in a device. See the bit field defintions and descriptions of register HW_USBCTRL_ENDPTCTRL1. CAUTION: If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (i.e., bulk type). Leaving an unconfigured endpoint control will cause undefined behavior for the data PID tracking on the active endpoint/direction.

Address: HW_USBCTRL_ENDPTCTRL7 – 8008_0000h base + 1DCh offset = 8008_01DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD6								TXE	TXR	TXI	RSVD5	TXT		TXD	TXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD3								RXE	RXR	RXI	RSVD2	RXT		RXD	RXS
W	▨											▨				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBCTRL_ENDPTCTRL7 field descriptions

Field	Description
31–24 RSVD6	same as HW_USBCTRL_ENDPTCTRL1
23 TXE	same as HW_USBCTRL_ENDPTCTRL1
22 TXR	same as HW_USBCTRL_ENDPTCTRL1
21 TXI	same as HW_USBCTRL_ENDPTCTRL1
20 RSVD5	same as HW_USBCTRL_ENDPTCTRL1
19–18 TXT	same as HW_USBCTRL_ENDPTCTRL1
17 TXD	same as HW_USBCTRL_ENDPTCTRL1
16 TXS	same as HW_USBCTRL_ENDPTCTRL1
15–8 RSVD3	same as HW_USBCTRL_ENDPTCTRL1
7 RXE	same as HW_USBCTRL_ENDPTCTRL1
6 RXR	same as HW_USBCTRL_ENDPTCTRL1
5 RXI	same as HW_USBCTRL_ENDPTCTRL1
4 RSVD2	same as HW_USBCTRL_ENDPTCTRL1
3–2 RXT	same as HW_USBCTRL_ENDPTCTRL1
1 RXD	same as HW_USBCTRL_ENDPTCTRL1
0 RXS	same as HW_USBCTRL_ENDPTCTRL1

Chapter 32
Integrated USB 2.0 PHY

32.1 USB PHY Overview

The i.MX28 contains an integrated USB 2.0 PHY macrocell capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s. The integrated PHY provides a standard UTM interface. The USB_DP and USB_DN pins connect directly to a USB connector.

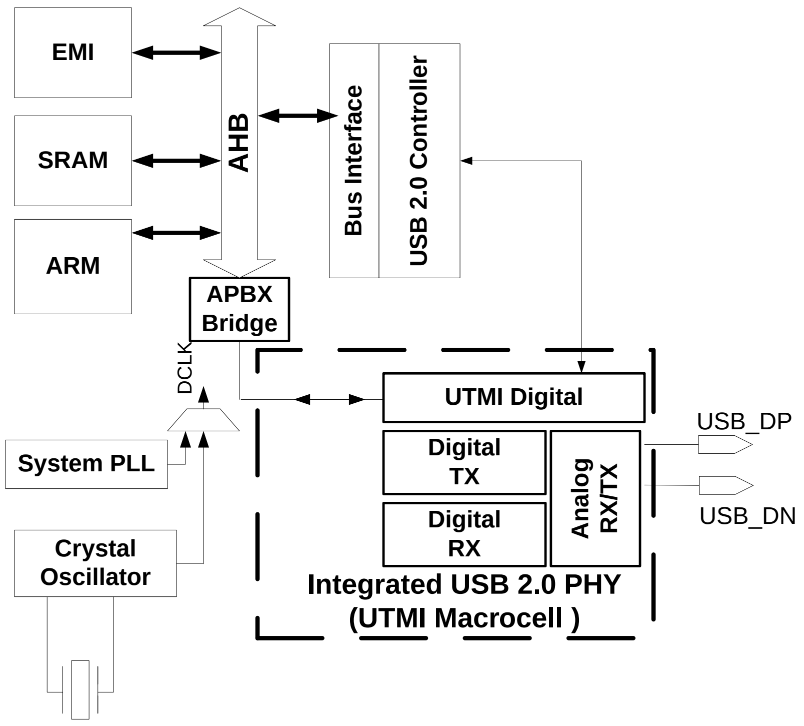


Figure 32-1. USB 2.0 PHY Block Diagram

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

32.2 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 32-2.

32.2.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection. The PLL supplies a 120 MHz signal to all of the digital logic. The UTMI block does a final divide-by-four to develop the 30 MHz clock used in the interface.

32.2.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake. In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

32.2.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and a 9X, 480 MHz sampled data from the high speed (HS) differential transceiver. As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

32.2.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module, as shown in the figure below and described further in this section.

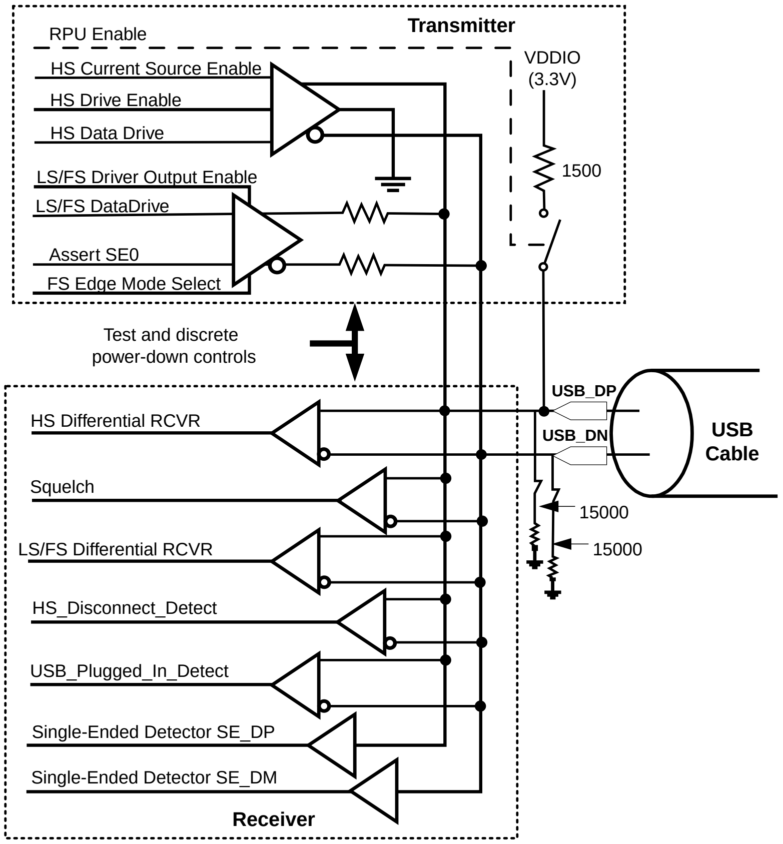


Figure 32-2. USB 2.0 PHY Analog Transceiver Block Diagram

32.2.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold. Otherwise, its output is 0. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each leg of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

32.2.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator. Its output is 1, if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is 0. Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

32.2.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator. The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_DP line is above the crossover point and the USB_DN line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

32.2.4.4 HS Disconnect Detector

This host-side function is not used in i.MX28 applications, but is included to make a complete UTMI macrocell. It is a differential analog receiver and threshold comparator. Its output is a 1 if the differential magnitude is greater than a nominal 575-mV threshold. Its output is 0, otherwise.

32.2.4.5 USB Plugged-In Detector

The USB plugged-in detector looks for both USB_DP and USB_DN to be high. There is a pair of large on-chip pullup resistors (200 KΩ) that hold both USB_DP and USB_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When in device mode, the host/hub interface that is upstream from the i.MX28 contains a 15 KΩ pulldown resistor that easily overrides the 200 KΩ pullup. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

32.2.4.6 Single-Ended USB_DP Receiver

The single-ended USB_DP receiver output is high whenever the USB_DP input is above its nominal 1.8 V threshold.

32.2.4.7 Single-Ended USB_DN Receiver

The single-ended USB_DN receiver output is high whenever the USB_DN input is above its nominal 1.8 V threshold.

32.2.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

32.2.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor. See Figure 32-2.

32.2.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90 W differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable. Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This is much too low of a load impedance for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure 32-3. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_DP signal.

32.2.5.2 Low-Speed/Full-Speed Differential Driver

The Low-Speed/full-speed differential drivers are essentially "open drain" low-impedance pulldown devices that are switched in a differential mode for low-speed or full-speed signaling, that is, either one or the other device is turned on to signal the "J" state or the "K" state. The tx_ls_en signal is used to select the USB_DP/USB_DN edge for low-speed or full-speed. Setting this bit to 1 selects the low-speed driver; otherwise the full-speed driver is selected. These drivers are both turned on, simultaneously, for high-speed signaling. This has the effect of switching in both 45 Ω terminating resistors. The tx_fs_hiz signal originates in the digital transmitter section. The hs_term signal that also controls these drivers comes from the UTMI.

32.2.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground. This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap circuit. The Iref, IBias, and V to I circuits are shared with the integrated battery charger.

32.2.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

The i.MX28 contains a switchable 1.5 KΩ pullup resistor on the USB_DP signal. This resistor is switched on to tell the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

32.2.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

The i.MX28 contains a switchable 15 KΩ pulldown resistor on both USB_DP and USB_DN signals. This is used in host mode to tell the device controller that a host is present.

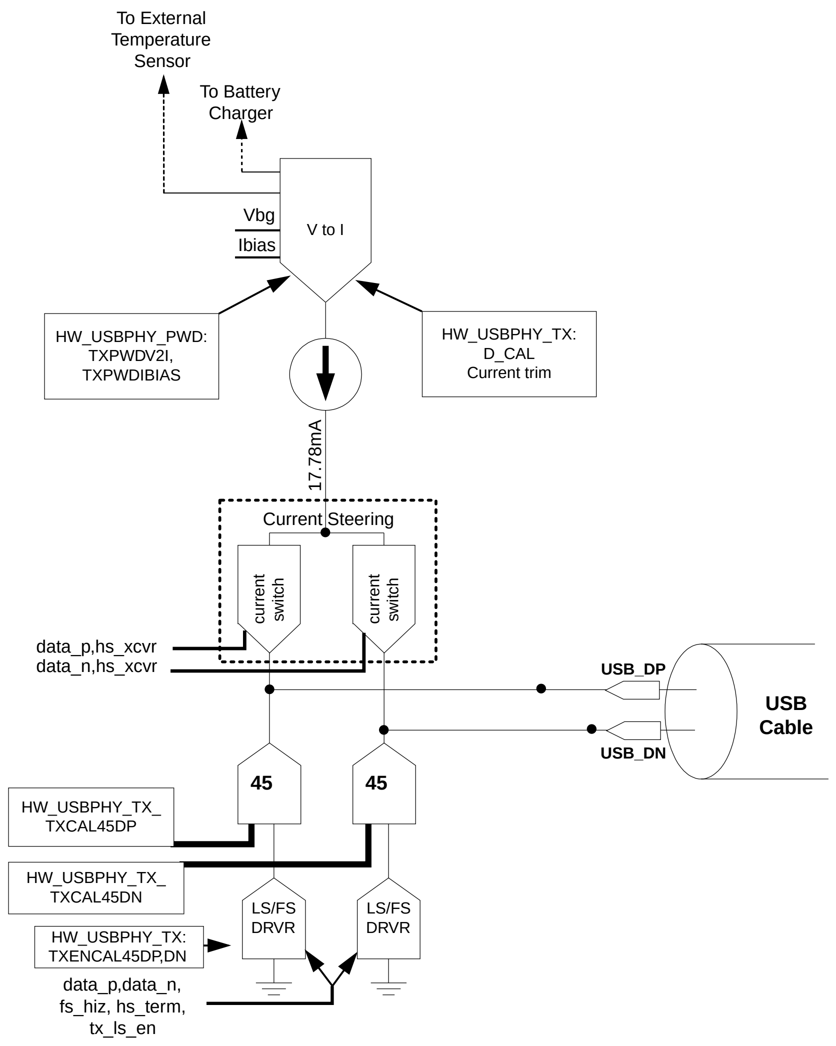


Figure 32-3. USB 2.0 PHY Transmitter Block Diagram

The following table summarizes the response of the PHY analog transmitter to various states of UTMI input and key transmit/receive state machine states.

Table 32-1. USB PHY Terminator States

UTMI OPMODE	UTMTERM	UTMXCVR	T/R	Function	45 Ω HI-Z	1500 Ω HI-Z	
00=Normal	0	00	X	HS	0	1	
	1	01/11	T	FS	0	0	
	1	10	T	LS	0	0	
	1	01/11	R	FS	1	0	SUSPEND
	1	10	R	LS	1	0	SUSPEND
	1	00	R	CHIRP	1	0	
	1	00	T	CHIRP	1	0	
	0	01	X	DISCONNECT	1	1	
01=NoDrive	0	00	T	HS	1	1	
	0	00	R	HS	1	1	
	1	01	X	FS	1	1	
	1	00	X	CHIRP	1	1	
	0	01	X	DISCONNECT	1	1	POR
10=NoNRZI NoBitStuff	0	00	X	HS	0	1	
	1	01	T	FS	0	0	
	1	01	R	FS	1	0	
	1	00	R	CHIRP	1	0	
	1	00	T	CHIRP	1	0	
	0	01	X	DISCONNECT	1	1	
11= Invalid	0	00	T	HS	1	1	
	0	00	R	HS	1	1	
	1	01	X	FS	1	1	
	1	00	X	CHIRP	1	1	
	0	01	X	DISCONNECT	1	1	

32.2.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

Note that HW_AUDIOOUT_REFCTRL_VDDXTAL_TO_VDDD is controlled by the SFTRST and CLKGATE bits in the AUDIOIN block.

32.3 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

32.4 Programmable Registers

USBPHY Hardware Register Format Summary

USBPHY0 base address is 0x8007C000; USBPHY1 base address is 0x8007E000

HW_USBPHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8007_C000	USB PHY Power-Down Register (HW_USBPHY_PWD)	32	R/W	001E_1C00h	32.4.1/2040
8007_C010	USB PHY Transmitter Control Register (HW_USBPHY_TX)	32	R/W	1006_0607h	32.4.2/2041
8007_C020	USB PHY Receiver Control Register (HW_USBPHY_RX)	32	R/W	0000_0000h	32.4.3/2043
8007_C030	USB PHY General Control Register (HW_USBPHY_CTRL)	32	R/W	C020_0000h	32.4.4/2044
8007_C040	USB PHY Status Register (HW_USBPHY_STATUS)	32	R/W	0000_0000h	32.4.5/2047
8007_C050	USB PHY Debug Register (HW_USBPHY_DEBUG)	32	R/W	7F18_0000h	32.4.6/2048
8007_C060	UTMI Debug Status Register 0 (HW_USBPHY_DEBUG0_STATUS)	32	R	0000_0000h	32.4.7/2050
8007_C070	UTMI Debug Status Register 1 (HW_USBPHY_DEBUG1)	32	R/W	0000_1000h	32.4.8/2051
8007_C080	UTMI RTL Version (HW_USBPHY_VERSION)	32	R	0402_0000h	32.4.9/2052

HW_USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8007_C090	USB PHY IP Block Register (HW_USBPHY_IP)	32	R/W	0000_0000h	32.4.10/2052

32.4.1 USB PHY Power-Down Register (HW_USBPHY_PWD)

The USB PHY Power-Down Register provides overall control of the PHY power state.

HW_USBPHY_PWD: 0x000

HW_USBPHY_PWD_SET: 0x004

HW_USBPHY_PWD_CLR: 0x008

HW_USBPHY_PWD_TOG: 0x00c

Address: HW_USBPHY_PWD – 8007_C000h base + 0h offset = 8007_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2											RXPWDRX	RXPWDDIFF	RXPWD1PT1	RXPWDENV	RSVD1 [16:16]
W	▨															▨
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:13]			TXPWDV2I	TXPWDIBIAS	TXPWDFS	RSVD0									
W	▨						▨									
Reset	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_PWD field descriptions

Field	Description
31–21 RSVD2	Reserved.
20 RXPWDRX	0 = Normal operation. 1 = Power-down the entire USB PHY receiver block except for the full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled.
19 RXPWDDIFF	0 = Normal operation. 1 = Power-down the USB high-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled.

HW_USBPHY_PWD field descriptions (continued)

Field	Description
18 RXPWD1PT1	0 = Normal operation. 1 = Power-down the USB full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled.
17 RXPWDENV	0 = Normal operation. 1 = Power-down the USB high-speed receiver envelope detector (squelch signal). Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled.
16–13 RSVD1	Reserved.
12 TXPWDV2I	0 = Normal operation. 1 = Power-down the USB PHY transmit V-to-I converter and the current mirror. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
11 TXPWDIBIAS	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10 TXPWDFS	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of HW_USBPHY_CTRL is enabled.
9–0 RSVD0	Reserved.

32.4.2 USB PHY Transmitter Control Register (HW_USBPHY_TX)

The USB PHY Transmitter Control Register handles the transmit controls.

HW_USBPHY_TX: 0x010

HW_USBPHY_TX_SET: 0x014

HW_USBPHY_TX_CLR: 0x018

HW_USBPHY_TX_TOG: 0x01c

Address: HW_USBPHY_TX – 8007_C000h base + 10h offset = 8007_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD5			USBPHY_TX_ EDGECTRL			USBPHY_TX_ SYNC_INVERT	USBPHY_TX_ SYNC_MUX	RSVD4		TXENCAL45DP	RSVD3	TXCAL45DP			
W	▨	▨	▨						▨	▨		▨				
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2		TXENCAL45DN	RSVD1	TXCAL45DN				RSVD0				D_CAL			
W	▨	▨		▨					▨	▨	▨	▨				
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1

HW_USBPHY_TX field descriptions

Field	Description
31–29 RSVD5	Reserved.
28–26 USBPHY_TX_ EDGECTRL	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.
25 USBPHY_TX_ SYNC_INVERT	Changes clock edge that sync mux will use when USBPHY_TX_SYNC_MUX is high. NOT FOR CUSTOMER USE.
24 USBPHY_TX_ SYNC_MUX	NOT FOR CUSTOMER USE. While in testmode enables clock jitter analysis by resyncing data to the USB_DP and USB_DM pins. 0 = No Sync, 1= Sync. When EMI clock is ungated the USB data is resynced with EMI clock (for EMI jitter analysis), otherwise the USB clock is used for resync.
23–22 RSVD4	Reserved.
21 TXENCAL45DP	This bit is not used and must remain cleared.
20 RSVD3	Reserved.
19–16 TXCAL45DP	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15–14 RSVD2	Reserved.

HW_USBPHY_TX field descriptions (continued)

Field	Description
13 TXENCAL45DN	This bit is not used and must remain cleared.
12 RSVD1	Reserved.
11–8 TXCAL45DN	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7–4 RSVD0	Reserved.
3–0 D_CAL	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

32.4.3 USB PHY Receiver Control Register (HW_USBPHY_RX)

The USB PHY Receiver Control Register handles receive path controls.

HW_USBPHY_RX: 0x020

HW_USBPHY_RX_SET: 0x024

HW_USBPHY_RX_CLR: 0x028

HW_USBPHY_RX_TOG: 0x02c

Address: HW_USBPHY_RX – 8007_C000h base + 20h offset = 8007_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2									RXDBYPASS	RSVD1[21:16]					
W	▨										▨					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1[15:7]									DISCONADJ			RSVD0	ENVADJ		
W	▨												▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_RX field descriptions

Field	Description
31–23 RSVD2	Reserved.
22 RXDBYPASS	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21–7 RSVD1	Reserved.
6–4 DISCONADJ	The DISCONADJ field adjusts the trip point for the disconnect detector: 0000 = Trip-Level Voltage is 0.57500 V 0001 = Trip-Level Voltage is 0.56875 V 0010 = Trip-Level Voltage is 0.58125 V 0011 = Trip-Level Voltage is 0.58750 V 01XX = Reserved 1XXX = Reserved
3 RSVD0	Reserved.
2–0 ENVADJ	The ENVADJ field adjusts the trip point for the envelope detector. 0000 = Trip-Level Voltage is 0.12500 V 0001 = Trip-Level Voltage is 0.10000 V 0010 = Trip-Level Voltage is 0.13750 V 0011 = Trip-Level Voltage is 0.15000 V 01XX = Reserved 1XXX = Reserved

32.4.4 USB PHY General Control Register (HW_USBPHY_CTRL)

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

HW_USBPHY_CTRL: 0x030

HW_USBPHY_CTRL_SET: 0x034

HW_USBPHY_CTRL_CLR: 0x038

HW_USBPHY_CTRL_TOG: 0x03c

Address: HW_USBPHY_CTRL – 8007_C000h base + 30h offset = 8007_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	UTMI_ SUSPENDM	HOST_FORCE_LS_SE0	RSVD3	ENAUTOSET_USBCLKS	ENAUTOCLR_ USBCLKGATE	FSDLL_RST_EN	ENVBUSCHG_WKUP	ENIDCHG_WKUP	ENDPDMCHG_WKUP	ENAUTOCLR_PHY_PWD	ENAUTOCLR_CLKGATE	ENAUTO_PWRON_PLL	WAKEUP_IRQ	ENIRQWAKEUP
W			▨		▨											
Reset	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUTMILEVEL3	ENUTMILEVEL2	DATA_ON_LRADC	DEVPLUGIN_IRQ	ENIRQDEVPLUGIN	RESUME_IRQ	ENIRQRESUMEDETECT	RESUMEIRQSTICKY	ENOTGIDDETECT	RSVD1	DEVPLUGIN_POLARITY	ENDEVPLUGINDETECT	HOSTDISCONDETECT_IRQ	ENIRQHOSTDISCON	ENHOSTDISCONDETECT	RSVD0
W										▨						▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_CTRL field descriptions

Field	Description
31 SFTRST	Writing a 1 to this bit will soft-reset the HW_USBPHY_PWD, HW_USBPHY_TX, HW_USBPHY_RX, and HW_USBPHY_CTRL registers.
30 CLKGATE	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wakeup event when USB is suspended while ENAUTOCLR_CLKGATE bit of HW_USBPHY_CTRL is enabled.
29 UTMI_SUSPENDM	Used by the PHY to indicate a powered-down state. If all the power-down bits in the HW_USBPHY_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28 HOST_FORCE_LS_SE0	Forces the next FS packet that is transmitted to have a EOP with low-speed timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the HW_USBPHY_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with HW_USBPHY_DEBUG_HOST_RESUME_DEBUG.
27 RSVD3	Reserved.
26 ENAUTOSET_USBCLKS	Enables the feature to auto-clear the EN_USB_CLKS register bits in HW_CLKCTRL_PLL1CTRL0/HW_CLKCTRL_PLL1CTRL1 if there is wakeup event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.

HW_USBPHY_CTRL field descriptions (continued)

Field	Description
25 ENAUTOCLR_ USBCLKGATE	Enables the feature to auto-clear the USB0_CLKGATE/USB1_CLKGATE register bit in HW_DIGCTL_CTRL if there is wakeup event on USB0/USB1 while USB0/USB1 is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
24 FSDLL_RST_EN	Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.
23 ENVBUSCHG_WKUP	Enables the feature to wakeup USB if VBUS is toggled when USB is suspended.
22 ENIDCHG_WKUP	Enables the feature to wakeup USB if ID is toggled when USB is suspended.
21 ENDPDMCHG_WKUP	Enables the feature to wakeup USB if DP/DM is toggled when USB is suspended. This bit is enabled by default.
20 ENAUTOCLR_PHY_PWD	Enables the feature to auto-clear the PWD register bits in HW_USBPHY_PWD if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
19 ENAUTOCLR_CLKGATE	Enables the feature to auto-clear the CLKGATE bit if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
18 ENAUTO_PWRON_PLL	Enables the feature to auto-enable the POWER bit of HW_CLKCTRL_PLLxCTRL0 if there is wakeup event if USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
17 WAKEUP_IRQ	Indicates that there is a wakeup event. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
16 ENIRQWAKEUP	Enables interrupt for the wakeup events.
15 ENUTMILEVEL3	Enables UTMI+ Level3. This should be enabled if needs to support external FS Hub with LS device connected
14 ENUTMILEVEL2	Enables UTMI+ Level2. This should be enabled if needs to support LS device
13 DATA_ON_LRADC	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12 DEVPLUGIN_IRQ	Indicates that the device is connected. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
11 ENIRQDEVPLUGIN	Enables interrupt for the detection of connectivity to the USB line.
10 RESUME_IRQ	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
9 ENIRQRESUMEDETECT	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8 RESUMEIRQSTICKY	Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.

HW_USBPHY_CTRL field descriptions (continued)

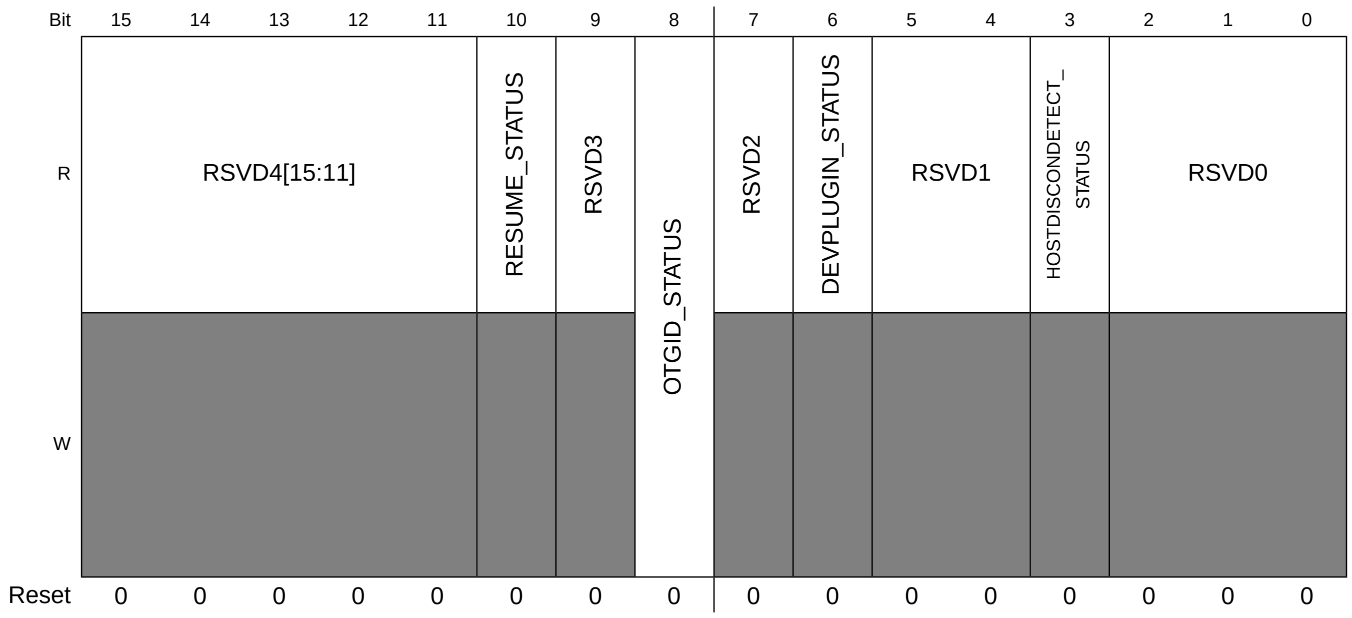
Field	Description
7 ENOTGIDDETECT	Enables circuit to detect resistance of MiniAB ID pin.
6 RSVD1	Reserved.
5 DEVPLUGIN_POLARITY	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4 ENDEVPLUGINDETECT	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3 HOSTDISCONDETECT_IRQ	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the SCT clear address space and not by a general write.
2 ENIRQHOSTDISCON	Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1 ENHOSTDISCONDETECT	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet.
0 RSVD0	Reserved.

32.4.5 USB PHY Status Register (HW_USBPHY_STATUS)

The USB PHY Status Register holds results of IRQ and other detects.

Address: HW_USBPHY_STATUS – 8007_C000h base + 40h offset = 8007_C040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



HW_USBPHY_STATUS field descriptions

Field	Description
31–11 RSVD4	Reserved.
10 RESUME_STATUS	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9 RSVD3	Reserved.
8 OTGID_STATUS	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, indicating device (B) side.
7 RSVD2	Reserved.
6 DEVPLUGIN_STATUS	Indicates that the device has been connected on the USB_DP and USB_DM lines.
5–4 RSVD1	Reserved.
3 HOSTDISCONDETECT_ STATUS	Indicates that the device has disconnected while in high-speed host mode.
2–0 RSVD0	Reserved.

32.4.6 USB PHY Debug Register (HW_USBPHY_DEBUG)

This register is used to debug the USB PHY.

HW_USBPHY_DEBUG: 0x050

HW_USBPHY_DEBUG_SET: 0x054

HW_USBPHY_DEBUG_CLR: 0x058

HW_USBPHY_DEBUG_TOG: 0x05c

Address: HW_USBPHY_DEBUG – 8007_C000h base + 50h offset = 8007_C050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD3	CLKGATE	HOST_RESUME_DEBUG	SQUELCHRESETLENGTH				ENSQUELCHRESET	RSVD2			SQUELCHRESETCOUNT				
W	▨								▨							
Reset	0	1	1	1	1	1	1	1	0	0	0	1	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1			ENTX2RXCOUNT	TX2RXCOUNT				RSVD0		ENHSTPULLDOWN		HSTPULLDOWN		DEBUG_INTERFACE_HOLD	OTGIDPIOLOCK
W	▨								▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_DEBUG field descriptions

Field	Description
31 RSVD3	Reserved.
30 CLKGATE	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29 HOST_RESUME_DEBUG	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28–25 SQUELCHRESETLENGTH	Duration of RESET in terms of the number of 480-MHz cycles.
24 ENSQUELCHRESET	Set bit to allow squelch to reset high-speed receive.
23–21 RSVD2	Reserved.

HW_USBPHY_DEBUG field descriptions (continued)

Field	Description
20–16 SQUELCHRESETCOUNT	Delay in between the detection of squelch to the reset of high-speed RX.
15–13 RSVD1	Reserved.
12 ENTX2RXCOUNT	Set this bit to allow a countdown to transition in between TX and RX.
11–8 TX2RXCOUNT	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7–6 RSVD0	Reserved.
5–4 ENHSTPULLDOWN	Set bit 5 to 1 to override the control of the USB_DP 15-KOhm pulldown. Set bit 4 to 1 to override the control of the USB_DM 15-KOhm pulldown. Clear to 0 to disable.
3–2 HSTPULLDOWN	Set bit 3 to 1 to pull down 15-KOhm on USB_DP line. Set bit 2 to 1 to pull down 15-KOhm on USB_DM line. Clear to 0 to disable.
1 DEBUG_INTERFACE_ HOLD	Use holding registers to assist in timing for external UTMI interface.
0 OTGIDPIOLOCK	Once OTG ID from HW_USBPHY_STATUS_OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

32.4.7 UTMI Debug Status Register 0 (HW_USBPHY_DEBUG0_STATUS)

The UTMI Debug Status Register 0 holds multiple views for counters and status of state machines. This is used in conjunction with the HW_USBPHY_DEBUG1.DBG_ADDRESS field to choose which function to view. The default is described in the bit fields below and is used to count errors.

Address: HW_USBPHY_DEBUG0_STATUS – 8007_C000h base + 60h offset = 8007_C060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SQUELCH_COUNT						UTMI_RXERROR_FAIL_COUNT										LOOP_BACK_FAIL_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_DEBUG0_STATUS field descriptions

Field	Description
31–26 SQUELCH_ COUNT	Running count of the squelch reset instead of normal end for HS RX.
25–16 UTMI_ RXERROR_ FAIL_COUNT	Running count of the UTMI_RXERROR.
15–0 LOOP_BACK_ FAIL_COUNT	Running count of the failed pseudo-random generator loopback. Each time entering testmode, counter goes to 900D and will count up for every detected packet failure in digital/analog loopback tests.

32.4.8 UTMI Debug Status Register 1 (HW_USBPHY_DEBUG1)

Chooses the muxing of the debug register to be shown in HW_USBPHY_DEBUG0_STATUS.

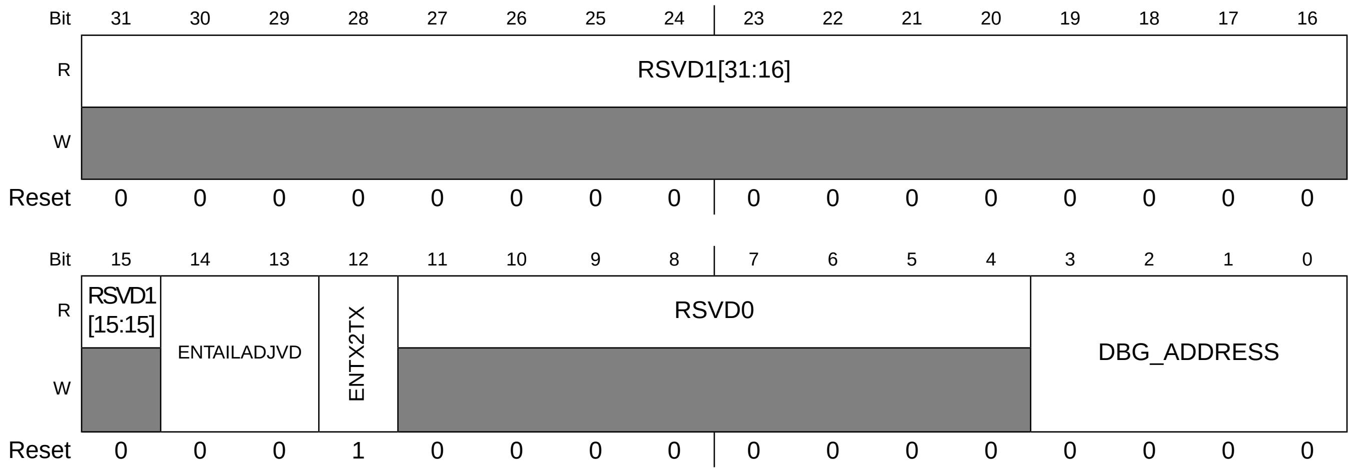
HW_USBPHY_DEBUG1: 0x070

HW_USBPHY_DEBUG1_SET: 0x074

HW_USBPHY_DEBUG1_CLR: 0x078

HW_USBPHY_DEBUG1_TOG: 0x07c

Address: HW_USBPHY_DEBUG1 – 8007_C000h base + 70h offset = 8007_C070h



HW_USBPHY_DEBUG1 field descriptions

Field	Description
31–15 RSVD1	Reserved.

HW_USBPHY_DEBUG1 field descriptions (continued)

Field	Description
14–13 ENTAILADJVD	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
12 ENTX2TX	This bit has no function in the STMP37xx.
11–4 RSVD0	Reserved.
3–0 DBG_ADDRESS	Chooses the multiplexing of the debug register to be shown in HW_USBPHY_DEBUG0_STATUS.

32.4.9 UTMI RTL Version (HW_USBPHY_VERSION)

Fields for RTL Version.

Address: HW_USBPHY_VERSION – 8007_C000h base + 80h offset = 8007_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

32.4.10 USB PHY IP Block Register (HW_USBPHY_IP)

The USB PHY IP Block Register IS FOR USE ONLY in non-Austin USB applications. It provides control of miscellaneous control bits found in other non-USB PIO control blocks that affects USB operations. - NOT FOR AUSTIN USE!!

HW_USBPHY_IP: 0x090

HW_USBPHY_IP_SET: 0x094

HW_USBPHY_IP_CLR: 0x098

HW_USBPHY_IP_TOG: 0x09c

Address: HW_USBPHY_IP – 8007_C000h base + 90h offset = 8007_C090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1							DIV_SEL		LFR_SEL		CP_SEL		TSTI_TX_DP	TSTI_TX_DM	ANALOG_ TESTMODE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD0													EN_ USB_ CLKS	PLL_ LOCKED	PLL_POWER
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_USBPHY_IP field descriptions

Field	Description
31–25 RSVD1	Reserved.
24–23 DIV_SEL	TEST MODE FOR FREESCALE USE ONLY. This field is currently NOT supported. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_div_sel). 0x0 **DEFAULT** — PLL frequency is 480 Mhz 0x1 **LOWER** — Lower the PLL fequency from 480MHz to 384Mhz 0x2 **LOWEST** — Lower the PLL fequency from 480MHz to 288MHz 0x3 **UNDEFINED** — Undefined
22–21 LFR_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts loop filter resistor. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_lfr_sel). 0x0 **DEFAULT** — Default loop filter resistor 0x1 **TIMES_2** — Doubles the loop filter resistor 0x2 **TIMES_05** — Halves the loop filter resistor 0x3 **UNDEFINED** — Undefined
20–19 CP_SEL	TEST MODE FOR FREESCALE USE ONLY. Adjusts charge pump current. These bits came from the clkctrl PIO control block (clkctrl_pllctrl0_cp_sel). 0x0 **DEFAULT** — Default charge pump current 0x1 **TIMES_2** — Doubles charge pump current 0x2 **TIMES_05** — Halves the charge pump current 0x3 **UNDEFINED** — Undefined
18 TSTI_TX_DP	Analog testmode bit. Drives value on the DP pad. Default value is 1'b0. This bit came from the test control module.

HW_USBPHY_IP field descriptions (continued)

Field	Description
17 TSTI_TX_DM	Analog testmode bit. Drives value on the DM pad. Default value is 1'b0. This bit came from the test control module.
16 ANALOG_ TESTMODE	Analog testmode bit. Set to 0 for normal operation. Set to 1 for engineering debug of analog PHY block. This bit came from the test control module.
15–3 RSVD0	Reserved.
2 EN_USB_CLKS	If set to 0, 9-phase PLL outputs for USB PHY are powered down. If set to 1, 9-phase PLL outputs for USB PHY are powered up. Additionally, the UTMICLK120_GATE and UTMICLK30_GATE must be deasserted in the UTMI phy to enable USB operation. This bit came from the clkctrl PIO control block (clkctrl_pllctrl0_en_usb_clks).
1 PLL_LOCKED	Software controlled bit to indicate when the USB PLL has locked. Software needs to wait 10 us after enabling the PLL POWER bit (0) before asserting this bit. If set to 0, tells the UTMI module that the USB PLL has not locked. If set to 1, tells the UTMI module that the USB PLL has locked. Software should clear this bit prior to turning off the USB PLL. This bit came from the clkctrl module.
0 PLL_POWER	USB PLL Power On (0 = PLL off; 1 = PLL On). Allow 10 us after turning the PLL on before using the PLL as a clock source. This is the time the PLL takes to lock to 480 MHz. This bit came from the clkctrl PIO control block (clkctrl_pllctrl0_power).

Chapter 33
LCD Interface (LCDIF)

33.1 LCD Interface (LCDIF) Overview

Many products based on the i.MX28 include an LCD panel with an integrated controller/driver. These smart LCDs are available in a range of sizes and capabilities, from simple text-only displays to QVGA, 16/18/24 bpp color TFT panels. Traditionally, many of these display controllers have had an asynchronous parallel MPU interface for command and data transfer to the frame buffer. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) or the VSYNC mode for high-speed data transfers. In addition to these displays, it is also common to provide support for digital video encoders that accept ITU-R BT.656 format 4:2:2 YCbCr digital component video and convert it to analog TV signals. The LCDIF block supports all of these different interfaces by providing fully programmable functionality and sharing register space, FIFOs, and ALU resources at the same time.

The high-level block diagram of the LCD interface is shown in Figure 33-1.

The block has several major features:

- High display resolutions up to 800x480 supported.
- AXI-based bus master mode for LCD writes and DMA operating modes for LCD reads requiring minimal CPU overhead.
- 8/16/18/24 bit LCD data bus support available depending on the package size.
- Programmable timing and parameters for system, MPU, VSYNC and DOTCLK LCD interfaces to support a wide variety of displays.
- ITU-R BT.656 mode (called Digital Video Interface or DVI mode here) including progressive-to-interlace feature and RGB to YCbCr 4:2:2 color space conversion to support 525/60 and 625/50 operation.

33.2 Operation

Bus Interface Mechanisms, through Initializing the LCDIF, describe the internal pipeline for the MPU write/read interface, VSYNC, DOTCLK, and DVI interfaces. Differences for each mode are then described in separate sections, as follows:

- MPU Interface
- VSYNC Interface
- DOTCLK Interface
- ITU-R BT.656 Digital Video Interface (DVI)

LCDIF pin usage by interface mode is described in LCDIF Pin Usage by Interface Mode.

The following figure shows the top-level block-diagram of LCDIF sub-system.

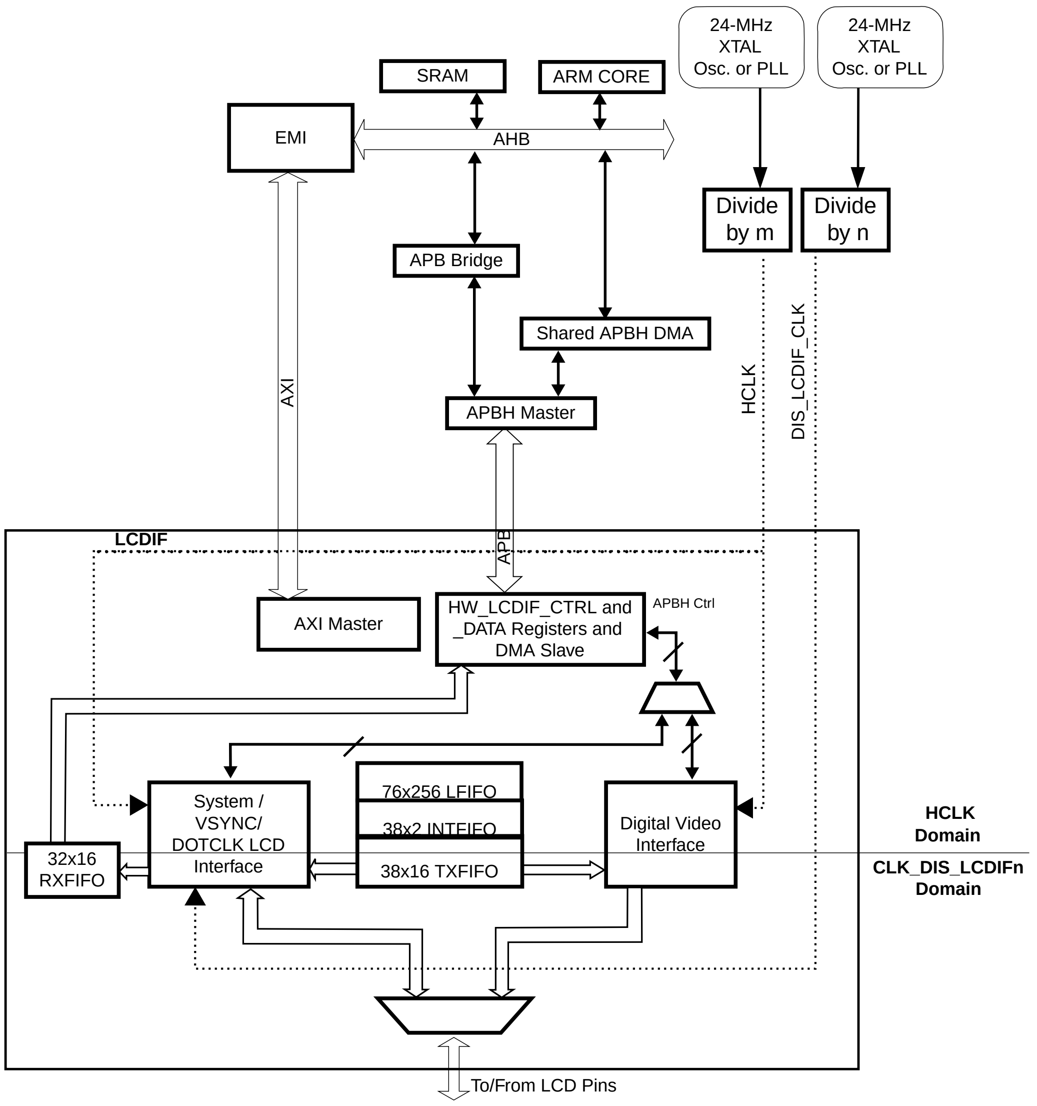


Figure 33-1. LCDIF Top Level Diagram

33.2.1 Bus Interface Mechanisms

The LCDIF block has memory-mapped control, data and status registers. It provides two efficient methods of transferring data to/from an external LCD controller or a digital video encoder, the APB DMA slave and AXI bus master. In either mode of operation, the bus interface portion of the block works off the HCLK domain, while the actual interface to external display/encoder works off the CLK_DIS_LCDIFn domain. There are three main FIFOs in the block datapath. The latency FIFO (LFIFO) in the write datapath is a 256 double words deep synchronous FIFO that offers buffering against system bandwidth and latency. Another FIFO in the write datapath, called TXFIFO, is a 16 words deep asynchronous FIFO that assists data crossing between the two clock domains. In the read datapath, there is a 16-deep asynchronous RXFIFO that is read by DMA operation. The following sub-sections describe the two system bus interface mechanisms.

33.2.1.1 Bus Master Operation in Write/Display Modes

In this mode, the LCDIF block acts as a master on AXI fabric shared with other blocks like PXP. This is a high performance mode and does not require DMA descriptor setup. The LCDIF_MASTER bit must be set to 1 to enable bus master operations. Software should program the CUR_BUF and NEXT_BUF registers to point to the base of the frame buffers that needs to be transferred out. In the MPU and VSYNC modes, once the frame buffer pointed to by the HW_LCDIF_CUR_BUF_ADDR is transferred out, LCDIF stops transmitting and turns off the RUN bit in CTRL register. It has to be setup and kicked off again for transmitting the next frame. In the DOTCLK and DVI modes, which are streaming modes in the true sense, software should start off with programming both the HW_LCDIF_CUR_BUF_ADDR and HW_LCDIF_NEXT_BUF_ADDR registers, and then, it only has to update the HW_LCDIF_NEXT_BUF_ADDR register in every frame. LCDIF will automatically update the HW_LCDIF_CUR_BUF_ADDR register with the value in HW_LCDIF_NEXT_BUF_ADDR at the end of current frame and start fetching the next frame from the new address. Thus, software has about one frame worth of time to update HW_LCDIF_NEXT_BUF_ADDR before it actually gets used. If for some reason, the HW_LCDIF_NEXT_BUF_ADDR register was not updated within a frame, LCDIF will keep transmitting the last frame until a new value is programmed into that register. The performance of the LCDIF block can be tweaked by changing the burst length and the number of outstanding requests that can be issued at a time depending on the bandwidth requirements.

LCDIF also provides the capability of interlacing a progressive frame by fetching odd lines in the first field and then fetching even lines in the second field. This feature can be used in the DVI mode and can be turned on by setting the INTERLACE_FIELDS bit in the HW_LCDIF_CTRL1 register.

33.2.1.2 DMA Operation in MPU Read Mode

Unlike previous generation SoCs, the DMA operation is now only used with the MPU read mode. None of the write modes can use the DMA for display purposes since APBH DMA will not be able to keep up with the high bandwidth requirements of the i.MX28. The LCDIF block resides on the AHB-APBH bridge DMA as a DMA slave. The AHB-APBH bridge DMA is used to move data from an external LCD controller, or any device that uses a 6800/8080 type interface, that needs to send data to the CPU through the LCDIF block. The software designer can take advantage of the DMA's linked descriptors that give substantial flexibility for setting up the frame buffers. The LCDIF provides a request signal to the central DMA if the LCDIF_MASTER bit in HW_LCDIF_CTRL register is 0. The request signal is asserted any time the LCDIF is enabled and there is at least one data to be read in its RXFIFO.The DMA request signal is also visible in the LCDIF control and status register. The DMA reads one word from the HW_LCDIF_DATA register every time it detects a toggle on the LCDIF request signal.

The CPU can also directly read commands or data by setting up the block in non-bus-master mode (HW_LCDIF_CTRL_LCDIF_MASTER = 0) and reading directly to the HW_LCDIF_DATA register without setting up the DMA descriptors. The FIFO status bits in the HW_LCDIF_STAT register indicate the full and empty states of the RXFIFO. When the RXFIFO is not empty, the data register can be safely read as required; doing otherwise will result in an incorrect operation.

33.2.2 Write Datapath

LCDIF expects all frame buffers to be arranged in the raster format since external displays and digital video encoders require the same. LCDIF directly fetches little-endian data from memory. This raw input data can be swizzled according to the INPUT_DATA_SWIZZLE field in the HW_LCDIF_CTRL register before any other operation is performed on the incoming data. The following four combinations are supported:

00 (0): No swizzle (little-endian)
01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
10 (2): Swap half-words
11 (3): Swap bytes within each half-word

The WORD_LENGTH field of HW_LCDIF_CTRL register indicates the input data/pixel format. HW_LCDIF_TRANSFER_COUNT register denotes how much data is contained in each frame. The H_COUNT field of this register indicates the number of pixels per line and V_COUNT indicates the total number of lines per frame. A special bit field in the HW_LCDIF_CTRL1 register, called the BYTE_PACKING_FORMAT, can be used to specify which bytes within the 32-bit word are going to be valid. For example, if the entire 32-bit word is valid, BYTE_PACKING_FORMAT should be set to 0xF, if only lower 3 bytes of each word in the frame buffer are valid, then BYTE_PACKING_FORMAT should be set to 0x7.

The LCD_DATABUS_WIDTH field in HW_LCDIF_CTRL register suggests the width of the bus going to the external display controller. If the LCD_DATABUS_WIDTH is not the same as WORD_LENGTH, LCDIF will do minor RGB to RGB color space conversion. For example, if the input frame has more bits per pixel than the display, for example, 16 bpp input frame going to 24 bpp LCD, LCDIF will pad the MSBs of each color to the LSBs of the same color for each pixel. If the input frame has lesser bits per pixel than the display, for example, 24 bpp input frame going to 16 bpp LCD, LCDIF will drop the LSBs of each color to go to the lower resolution. LCDIF also has the capability to support delta pixel displays by swizzling the R, G and B colors of each pixel in the odd and even lines of the frame separately by programming the ODD_LINE_PATTERN and the EVEN_LINE_PATTERN bitfields. This operation occurs after the RGB-to-RGB color space conversion operation.

LCDIF also supports RGB to YCbCr 4:2:2 color space conversion. This is useful in the DVI mode since the TV encoder requires input in YCbCr 4:2:2 format. The HW_LCDIF_CSC* registers have complete programmability over the CSC coefficients and offsets. The values must be written into these registers in the signed two's complement format.

The following list shows how the different input/output combinations can be obtained:

- WORD_LENGTH=1 indicates that the input is 8-bit data. This is most likely going to be used for sending commands in MPU interface, or maybe a grayscale image. Any combination of BYTE_PACKING_FORMAT [3:0] is permissible.

  Limitation: H_COUNT must be a multiple of the sum of BYTE_PACKING_FORMAT [3], BYTE_PACKING_FORMAT [2], BYTE_PACKING_FORMAT [1] and BYTE_PACKING_FORMAT [0]. LCD_DATABUS_WIDTH must be 1, indicating an 8-bit data bus.

- WORD_LENGTH=0 implies the input frame buffer is RGB 16 bits per pixel. DATA_FORMAT_16_BIT field determines the pixels are RGB 555 or RGB 565.

  Limitation: BYTE_PACKING_FORMAT [3:0] should be 0x3 or 0xC if there is only one pixel per word. If there are two pixels per word, it should be 0xF and H_COUNT will be restricted to be a multiple of 2 pixels.

- WORD_LENGTH=2 indicates that input frame buffer is RGB 18 bits per pixel, that is, RGB 666. The valid RGB values can be left-aligned or right-aligned within a 32-bit word. The alignment of the valid 18 bits within a word is indicated by the DATA_FORMAT_18_BIT bit.

  Limitation: BYTE_PACKING_FORMAT can be 0x7, 0xE or 0xF. Packed pixels are not supported in this case. H_COUNT can be any number.

- WORD_LENGTH=3 indicates that the input frame-buffer is RGB 24 bits per pixel (RGB 888). If BYTE_PACKING_FORMAT [3:0] is 0x7, it indicates that there is only one pixel per 32-bit word and there is no restriction on H_COUNT.

Limitation: If BYTE_PACKING_FORMAT [3:0] is 0xF, it indicates that the pixels are packed, that is, there are 4 pixels in 3 words or 12 bytes and H_COUNT must be a multiple of 4 pixels.

- YCBCR422_INPUT=1 implies that the input frame is in YCbCr 4:2:2 format. BYTE_PACKING_FORMAT must be 0xF.

  Limitation: LCD_DATABUS_WIDTH must be 8-bit and H_COUNT must be a multiple of 2 pixels.

- ODD_LINE_PATTERN and EVEN_LINE_PATTERN must be 0 when any of RGB_TO_YCBCR422_CSC or INTERLACE_FIELDS or YCBCR422_INPUT bits is 1.

After the RGB to RGB or RGB to YCbCr 4:2:2 color space conversions, there is one more opportunity to swizzle the data before sending it out to the display or the encoder. This can be done with the CSC_DATA_SWIZZLE field in the HW_LCDIF_CTRL register, and it provides the same options as the INPUT_DATA_SWIZZLE register.

Finally, there is an option to shift the output data before sending it out to the display. This is done based on the SHIFT_DIR and SHIFT_NUM_BITS fields in HW_LCDIF_CTRL register.

Figure 33-2 shows the general operations that occur in the write data path.

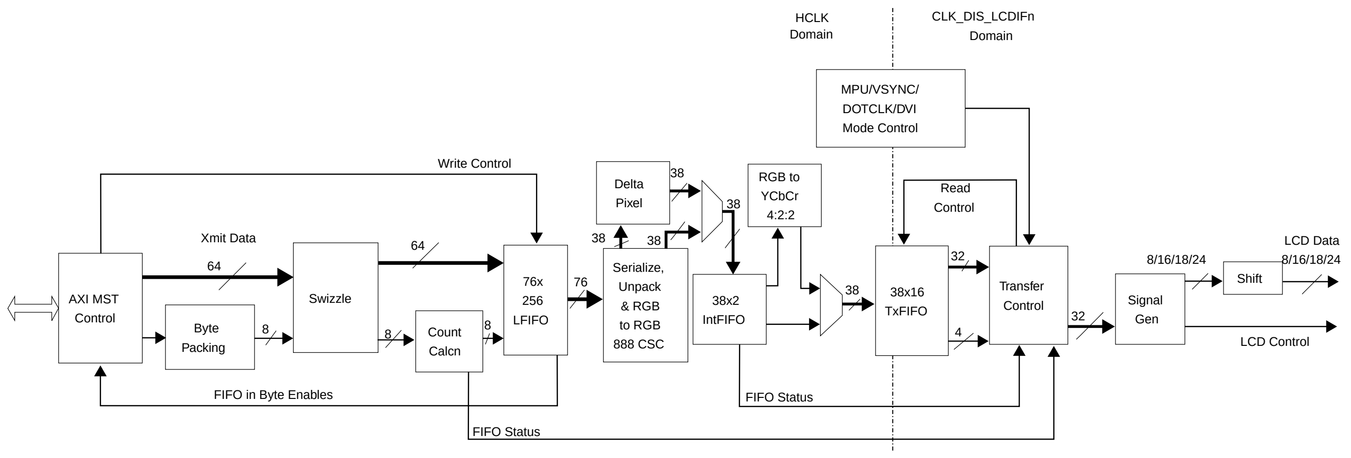


Figure 33-2. LCDIF Write Data Path

The examples in Figure 33-3 – Figure 33-6 illustrate some different combinations of register programming for write mode. Assume that the data written into the HW_LCDIF_DATA register is of the format {A7–A0, B7–B0, C7–C0, D7–D0} in 8-bit mode and {A15–A0, B15–B0} in 16-bit mode.

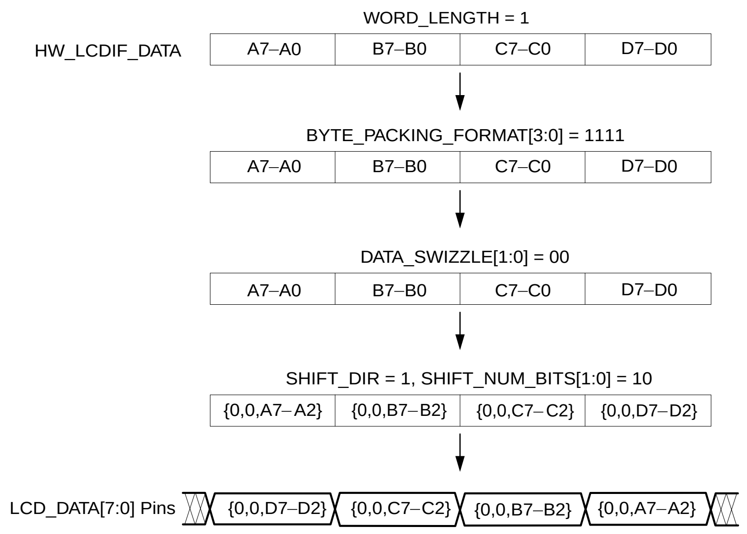


Figure 33-3. 8-Bit LCDIF Register Programming-Example A

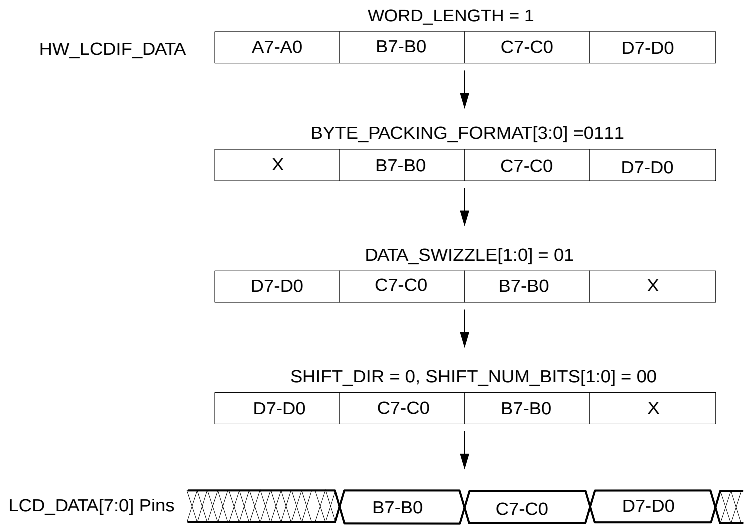


Figure 33-4. 8-Bit LCDIF Register Programming-Example B

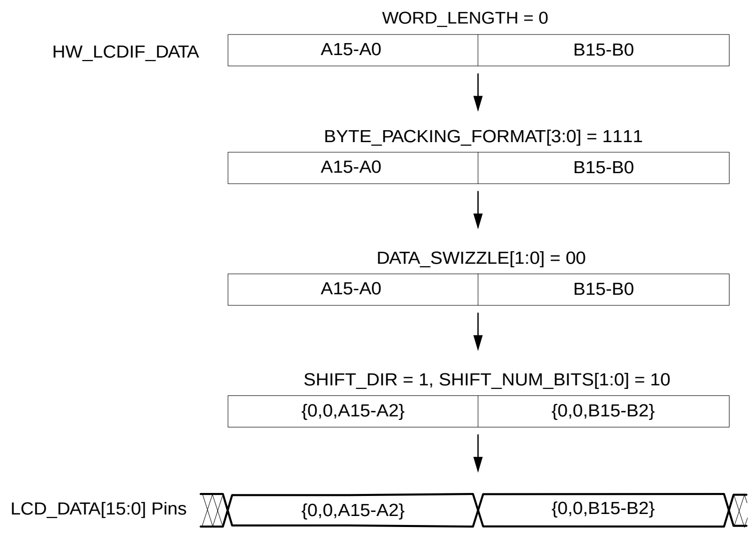


Figure 33-5. 16-Bit LCDIF Register Programming-Example A

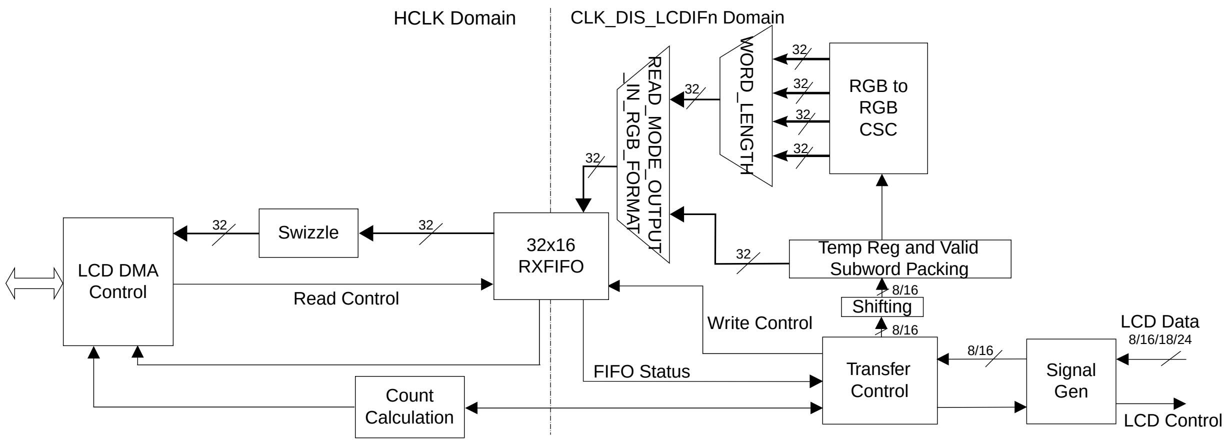


Figure 33-6. 16-Bit LCDIF Register Programming-Example B

33.2.3 Read Datapath

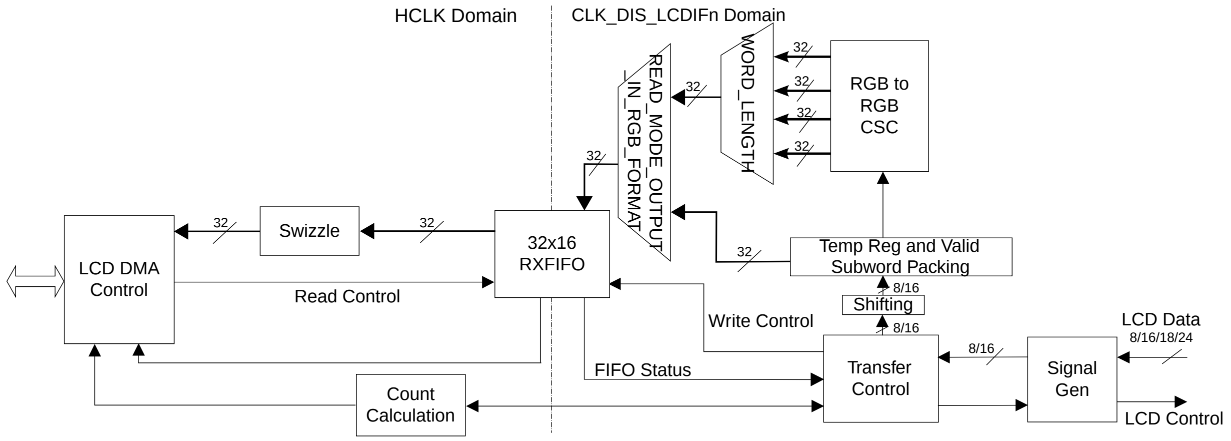


Figure 33-7. 16-Bit LCDIF Register Programming-Example B

The above figure shows the MPU read datapath in detail. LCDIF can read from an external LCD controller or any other device that follows the 6800/8080 MPU protocol. The size of the data read at a time on the interface is determined by the LCD_DATABUS_WIDTH bitfield. The data grabbed at every read strobe is called a subword and the number of subwords that can be packed in a 32-bit word is given by the READ_MODE_NUM_PACKED_SUBWORDS bitfield. Setting a non-zero value in the INITIAL_DUMMY_READ bitfield indicates to LCDIF to skip that many number of subwords before starting to record the read data. This feature is useful in the case of an LCD controller that returns the last written data the first time a read is issued, and then sends the correct data after that point. SHIFT_DIR and SHIFT_NUM_BITS bitfields indicate whether the data needs to be shifted before getting stored in the internal registers. For example, a value of 2 in READ_MODE_NUM_PACKED_SUBWORDS if lcd databus width is 8 bits indicates two bytes should be packed in a 32-bit word, while if the lcd databus width is 16 bits, it indicates that two half words (or 4 bytes) should be packed.

After the last subword within a word is reached, the block looks at the READ_PACK_DIR in the HW_LCDIF_CTRL2 register. If this bit is set, the block will swizzle the data, but only within the valid bytes, unlike in the write mode, where swizzle occurs across all 4 bytes. If the READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, LCDIF will convert the data obtained from the READ_PACK_DIR operation into 24-bit unpacked RGB and then re-convert it into 16/18/24 bpp RGB depending on the WORD_LENGTH field. The DATA_FORMAT_16/18/24_BIT bitfields are also considered while converting to 24-bit unpacked RGB format. For example, if DATA_FORMAT_18_BIT is 1, the RGB666 data will be packed in the upper bits [31:4]of a 32-bit word, and that bit is 0, the data will be packed in the lower bits [17:0]. After all these operations, the data gets written into the RXFIFO.

The following figures show some examples of how data is handled in different MPU read modes.

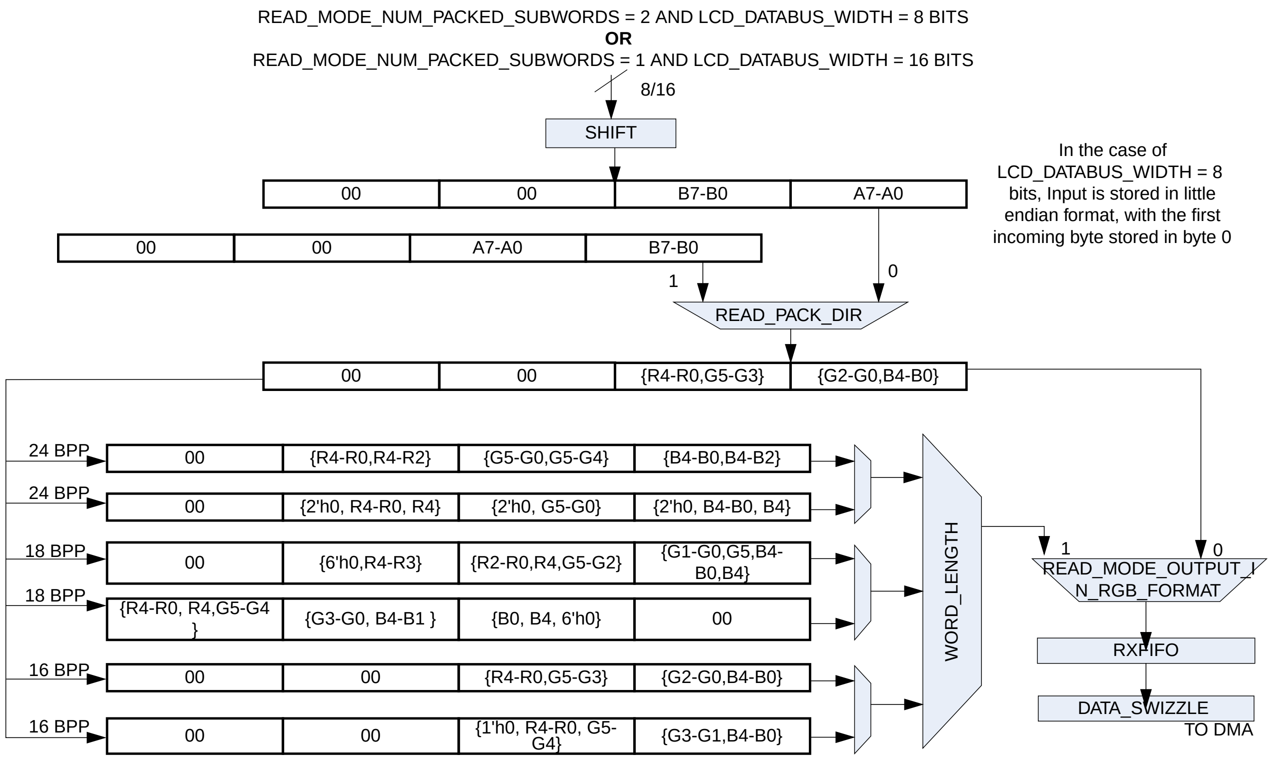


Figure 33-8. MPU Read Mode Operation - Page 1

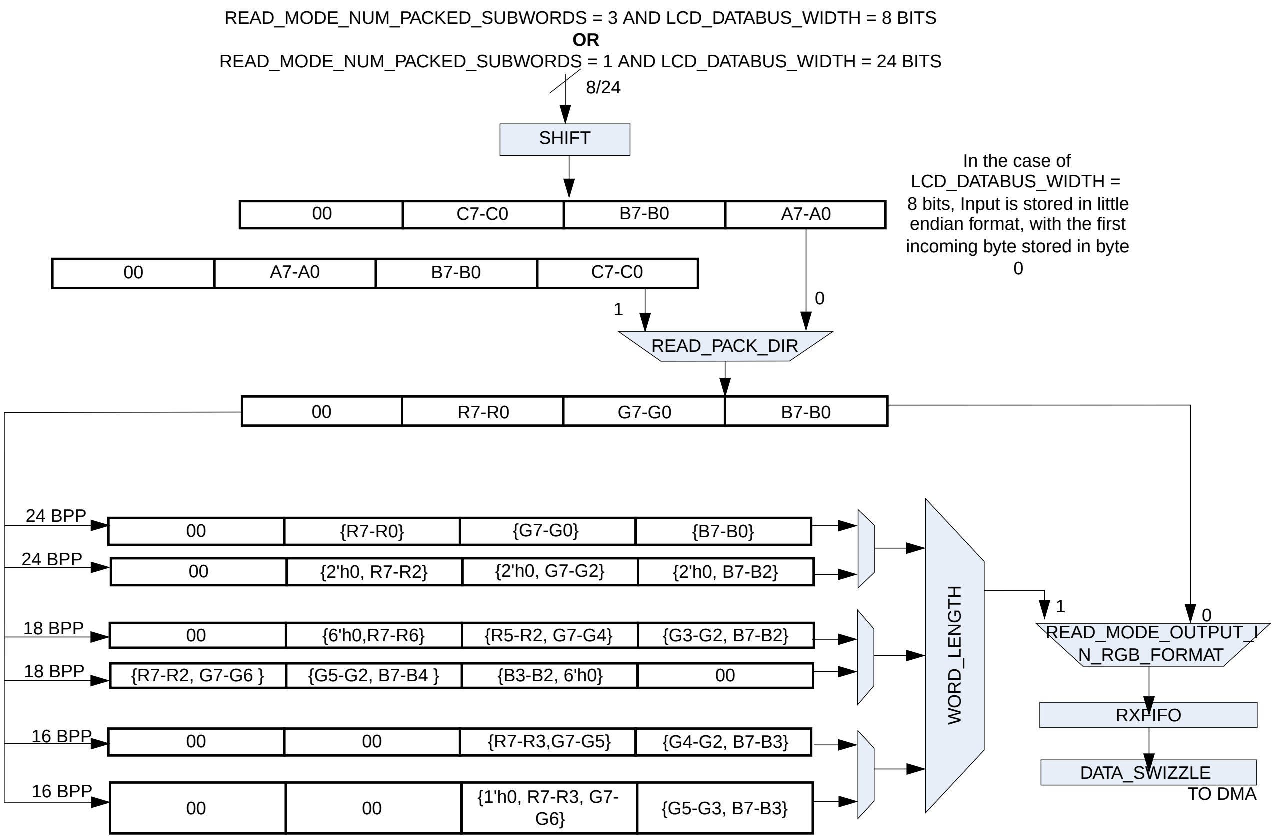

READ_MODE_NUM_PACKED_SUBWORDS = 3 AND LCD_DATABUS_WIDTH = 8 BITS
OR
READ_MODE_NUM_PACKED_SUBWORDS = 1 AND LCD_DATABUS_WIDTH = 24 BITS
8/24
SHIFT
In the case of LCD_DATABUS_WIDTH = 8 bits, Input is stored in little endian format, with the first incoming byte stored in byte 0
00 | C7-C0 | B7-B0 | A7-A0
00 | A7-A0 | B7-B0 | C7-C0
1
0
READ_PACK_DIR
00 | R7-R0 | G7-G0 | B7-B0
24 BPP
00 | {R7-R0} | {G7-G0} | {B7-B0}
24 BPP
00 | {2'h0, R7-R2} | {2'h0, G7-G2} | {2'h0, B7-B2}
18 BPP
00 | {6'h0,R7-R6} | {R5-R2, G7-G4} | {G3-G2, B7-B2}
18 BPP
{R7-R2, G7-G6 } | {G5-G2, B7-B4 } | {B3-B2, 6'h0} | 00
16 BPP
00 | 00 | {R7-R3,G7-G5} | {G4-G2, B7-B3}
16 BPP
00 | 00 | {1'h0, R7-R3, G7-G6} | {G5-G3, B7-B3}
WORD_LENGTH
1
0
READ_MODE_OUTPUT_IN_RGB_FORMAT
RXFIFO
DATA_SWIZZLE
TO DMA

Figure 33-9. MPU Read Mode Operation - Page 2

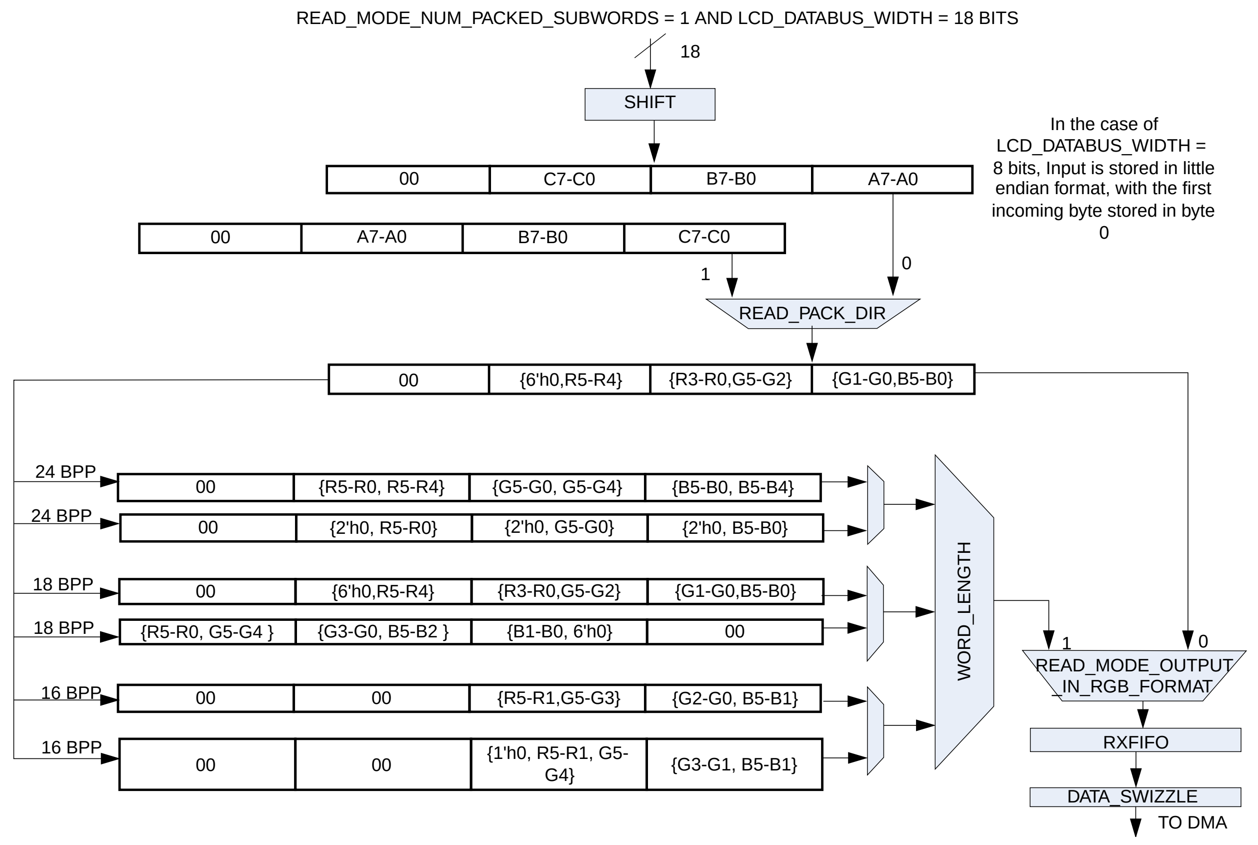


Figure 33-10. MPU Read Mode Operation - Page 3

Restrictions:

READ_PACK_DIR should only be used if it is required to swizzle the subwords before doing RGB to RGB CSC, otherwise the DATA_SWIZZLE field should be used to swizzle across bytes.

READ_PACK_DIR must be 0 if LCD_DATABUS_WIDTH is 8 bits and READ_MODE_NUM_PACKED_SUBWORDS =1

If READ_MODE_OUTPUT_IN_RGB_FORMAT bit is set, the following restrictions should be followed:

- If LCD_DATABUS_WIDTH = 8 bits, then READ_MODE_NUM_PACKED_SUBWORDS <= 3.
- If LCD_DATABUS_WIDTH = 16/18/24 bits, then READ_MODE_NUM_PACKED_SUBWORDS = 1.

33.2.4 LCDIF Interrupts

LCDIF supports a number of interrupts to aid controlling and status reporting of the block. All the interrupts have individual mask bits for enabling or disabling each of them. They all get funneled through a single interrupt line connected to the interrupt collector (ICOLL). The following list describes the different interrupts supported by LCDIF:

- Underflow interrupt is asserted when the clock domain crossing FIFO (TXFIFO) becomes empty but the block is in active display portion during that time. Software should take corrective action to make sure that this does not happen.
- In the bus master mode, the overflow interrupt will be asserted if the block has requested more data than it's FIFOs could hold. In the read mode, it will be asserted if the RxFIFO becomes full and the block reads more data.
- VSYNC edge interrupt will be asserted every time a leading VSYNC edge occurs.
- Cur_frame_done interrupt occurs at the end of every frame in all modes except DVI. In DVI mode, if IRQ_ON_ALTERNATE_FIELDS bit is set, it will occur at the end of every frame, otherwise it will occur at the end of every field.

33.2.5 Initializing the LCDIF

This section describes write modes and MPU read mode

33.2.5.1 Write Modes

The following initialization steps are common to all LCDIF write modes of operation before entering any particular mode.

Initialization steps:

1. To select the pins and their directions for talking to the LCD panel, set the appropriate bits in the HW_PINCTRL_MUXSELx registers in the PINCTRL block.
2. Start the CLK_DIS_LCDIFn clock and set the appropriate frequency by programming the registers in CLKCTRL.
3. Start the HCLK and set the appropriate frequency by programming the registers in CLKCTRL.
4. Bring the LCDIF out of soft reset and clock gate.

5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller. See Behavior During Reset for more information on Reset requirements.
6. Make sure READ_WRITEB bit in HW_LCDIF_CTRL register is 0.
7. Select AXI bus master mode by setting the LCDIF_MASTER bit in HW_LCDIF_CTRL register to 1.
8. Set the INPUT_DATA_SWIZZLE according to the endianness of the LCD controller. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24-bit input. Also, select the correct 16/18/24 bit data format with the corresponding fields in HW_LCDIF_CTRL register.
10. Set the BYTE_PACKING_FORMAT field in HW_LCDIF_CTRL1 according to the input frame.
11. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24-bit output.
12. Enable the necessary IRQs.

33.2.5.2 MPU Read Mode

The following initialization steps should be done to enter the MPU read mode of operation:

Initialization steps:

1. To select the pins and their directions for talking to the LCD panel, set the appropriate bits in the HW_PINCTRL_MUXSELx registers in the PINCTRL block.
2. Start the CLK_DIS_LCDIFn and set the appropriate frequency by programming the registers in CLKCTRL.
3. Start the HCLK and set the appropriate frequency by programming the registers in CLKCTRL.
4. Bring the LCDIF out of soft reset and clock gate.
5. Reset the LCD controller by setting LCDIF_CTRL1_RESET bit appropriately, being careful to observe the reset requirements of the controller.
6. Set the READ_WRITEB bit in HW_LCDIF_CTRL register to 1.

7. Select the DMA mode by making the LCDIF_MASTER bit in HW_LCDIF_CTRL register 0.
8. Also, set the DATA_SHIFT_DIR and SHIFT_NUM_BITS if it is required to shift the data left or right before it is output.
9. Indicate if the read data needs to color-space-converted and stored in a different RGB format by setting the READ_MODE_OUTPUT_IN_RGB_FORMAT field accordingly.
10. Set the WORD_LENGTH field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24-bit input if READ_MODE_OUTPUT_IN_RGB_FORMAT is required. Also, select the correct 16/18/24 bit data format with the corresponding fields in HW_LCDIF_CTRL register.
11. Set the READ_MODE_NUM_PACKED_SUBWORDS field in HW_LCDIF_CTRL2 according to the number of subwords per word required to be packed.
12. Set the READ_PACK_DIR to 1 if it is required to store the data in big-endian format.
13. Set the LCD_DATABUS_WIDTH appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24-bit output.
14. Enable the necessary IRQs.

33.2.6 MPU Interface

The MPU interface is used to transfer data and commands between the internal buffer of LCD controller/display and the MPU or vice versa at relatively lower speeds. LCDIF can support the 6800 as well as the 8080 MPU protocol. If DOTCLK_MODE, DVI_MODE and VSYNC_MODE bits in HW_LCDIF_CTRL registers are 0, it implies that the block is in MPU interface mode of operation. The LCDIF MPU mode has four basic timing parameters: Setup and Hold for the Command/Data register selection (TCS, TCH) and Setup and Hold for the Data bus (TDS, TDH). These parameters are expressed in CLK_DIS_LCDIFn cycles. The LCD_WR signal is used as the write strobe while LCD_RS signal is typically used to switch between command and data modes.

Figure 33-11 shows the timing-related information in the write mode of both 6800 and 8080 protocols.

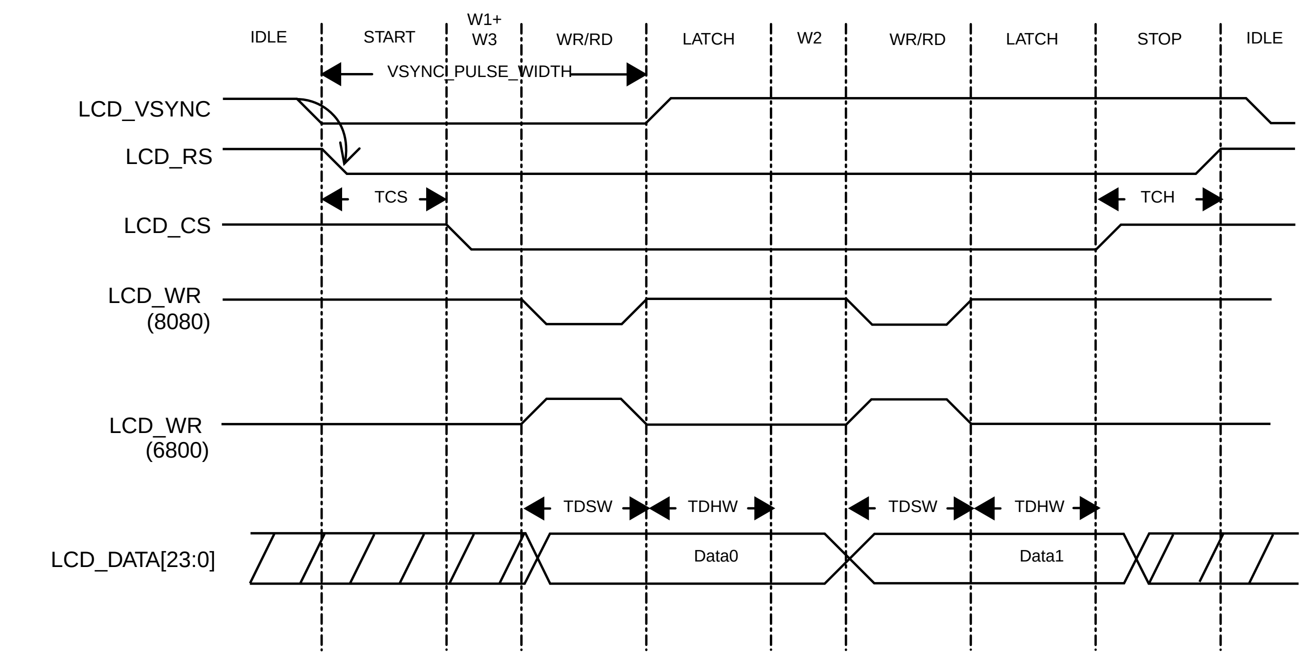


Figure 33-11. LCD Interface Signals in MPU Write Mode

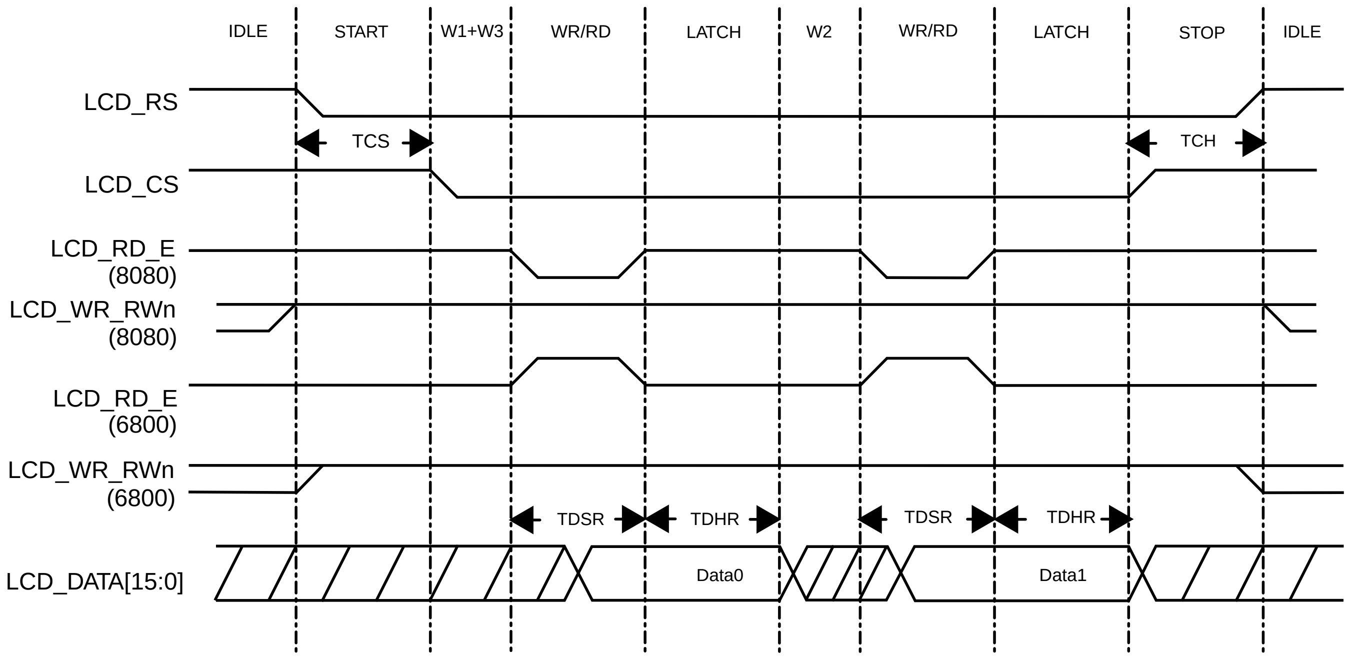


Figure 33-12. LCD Interface Signals in MPU Read Mode

The LCDIF has flexible pin and strobe timings which enable it to optimally support a wide range of LCDs. The minimum cycle time is two CLK_DIS_LCDIFn cycles (TDS=TDH=1). For example, this results in a maximum LCD data rate of 12 MB/s when CLK_DIS_LCDIFn is 24 MHz. TDS and TDH are 8-bit values, so the minimum LCDIF period is 510

CLK_DIS_LCDIFn cycles (47 KHz with a 24 MHz CLK_DIS_LCDIFn). The timings are not automatically adjusted if the CLK_DIS_LCDIFn frequency changes, so it may be necessary to adjust the timings if CLK_DIS_LCDIFn changes.

In the MPU interface mode, the HW_LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the LCDIF has received/transmitted all the data as per the HW_LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

33.2.6.1 Code Example to Initialize the LCDIF in MPU Write Mode

```
// Note: Common initialization steps in Initializing the LCDIF must also be
// executed along with the following code
BF_CS1(LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that the

                                    // idle state for LCD_RS signal is high, regardless of the

                                     // programming of the DATA_SELECT register.
BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, READ_WRITEB, 0);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 1);//Only if LCD controller implements a busy line
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values based

                             // on CLK_DIS_LCDIFn frequency and timing requirements of
controller.
                           // Note that these register must be non-zero for correct operation.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240); //For a 320 RGB x 240 display
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The LCDIF is now ready to receive data through DMA writes to the HW_LCDIF_DATA register or fetch data directly from memory as a bus master. Also, note that, while in soft DMA mode, the software will need to poll the FIFO STATUS bits to ensure that it does not overflow the LCDIF data buffers. When LCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

33.2.7 VSYNC Interface

The VSYNC interface uses the same protocol as the MPU interface, with an additional signal VSYNC at the frame rate of the display, as shown in Figure 33-11. It is used in the moving picture display mode where data has to be written to the internal LCD buffer at a speed higher than the display rate and displayed in synchronization with the VSYNC signal. This mode is selected by setting the VSYNC_MODE bit in HW_LCDIF_CTRL register. The VSYNC signal is programmable for period, polarity and direction. Many other programmable parameters are shared with the MPU interface. The VSYNC_OEB bit in HW_LCDIF_VDCTRL0 register indicates whether the display controller will send the VSYNC signal, or whether it should be generated by LCDIF. The timing of the VSYNC signal is based on the CLK_DIS_LCDIFn (make sure VSYNC_PULSE_WIDTH_UNIT = VSYNC_PERIOD_UNIT = 0 and VSYNC_ONLY = 1) and it is determined by the VSYNC_PERIOD, VSYNC_PULSE_WIDTH and VSYNC_POL fields in HW_LCDIF_VDCTRL0-4 registers. The SYNC_SIGNALS_ON bit in HW_LCDIF_VDCTRL4 register must be set if the target requires the VSYNC signal to be generated by LCDIF. If the WAIT_FOR_VSYNC_EDGE bit in HW_LCDIF_CTRL register is set, it indicates that the hardware should wait until it sees the leading VSYNC edge before starting the data transfer. The VERITCAL_WAIT_CNT indicates the number of CLK_DIS_LCDIFn cycles from the leading VSYNC edge after which data transfer will be started on the interface.

In the VSYNC interface mode, the HW_LCDIF_CTRL_ BYPASS_COUNT bit must be 0. The RUN bit is cleared automatically once the LCDIF has received/transmitted all the data as per the HW_LCDIF_TRANSFER_COUNT register and has completed the transfer to the panel. The current transfer can be cancelled/aborted if the RUN bit is manually made 0.

33.2.7.1 Code Example to initialize LCDIF in VSYNC mode

```
// Note: Common initialization steps in Initializing the LCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DATA_SELECT, 1); // 0 if sending command, 1 if sending data. Note that
//the idle state for LCD_RS signal is high, regardless of the programming of the DATA_SELECT
//register.
BF_CS1 (LCDIF_CTRL, MODE86, 8080_MODE);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 0); //Must be 0 in MPU mode
BF_CS1 (LCDIF_CTRL1, BUSY_ENABLE, 0);
BF_CS4 (LCDIF_TIMING, CMD_HOLD, 2, CMD_SETUP, 2, DATA_HOLD, 2, DATA_SETUP, 2);  //Values
//based on CLK_DIS_LCDIFn frequency and timing requirements of controller. Note that these
register
//must be non-zero for the MPU and VSYNC modes.
BF_CS2 (LCDIF_TRANSFER_COUNT, H_COUNT, 320, V_COUNT, 240);//For a 320 RGB x 240 display
//The following section indicates setting up the VSYNC signal timing when VSYNC is an output
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Making VSYNC signal an output
BF_CS1 (LCDIF_VDCTRL4, VSYNC_ONLY, 1); //Only need to generate VSYNC signal
BF_CS1 (VDCTRL0, VSYNC_POL, 0); //Setting the polarity of VSYNC signal to be low during
//VSYNC_PULSE_WIDTH time
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 0, VSYNC_PULSE_WIDTH_UNIT, 0);
BF_CS2 (LCDIF_VDCTRL1, VSYNC_PERIOD, 400000, VSYNC_PULSE_WIDTH, 100);//Frame display rate in
//terms of number of CLK_DIS_LCDIFns.
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 0, HSYNC_PERIOD, 0);
BF_CS1 (LCDIF_VDCTRL3, VERTICAL_WAIT_CNT, 50);
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS2 (LCDIF_CTRL, VSYNC_MODE, 1, WAIT_FOR_VSYNC_EDGE, 1); //set WAIT_FOR_VSYNC_EDGE if
//software wishes to transfer the next frame after the VSYNC edge occurs.
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

The LCDIF is now ready to receive data through DMA writes to the HW_LCDIF_DATA register or fetch data directly from memory as a bus master. When LCDIF is done transmitting H_COUNT x V_COUNT pixels, it will stop, turn off the RUN bit and assert the cur_frame_done interrupt.

33.2.8 DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays. It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

Figure 33-13 shows the DOTCLK protocol with its programmable parameters.

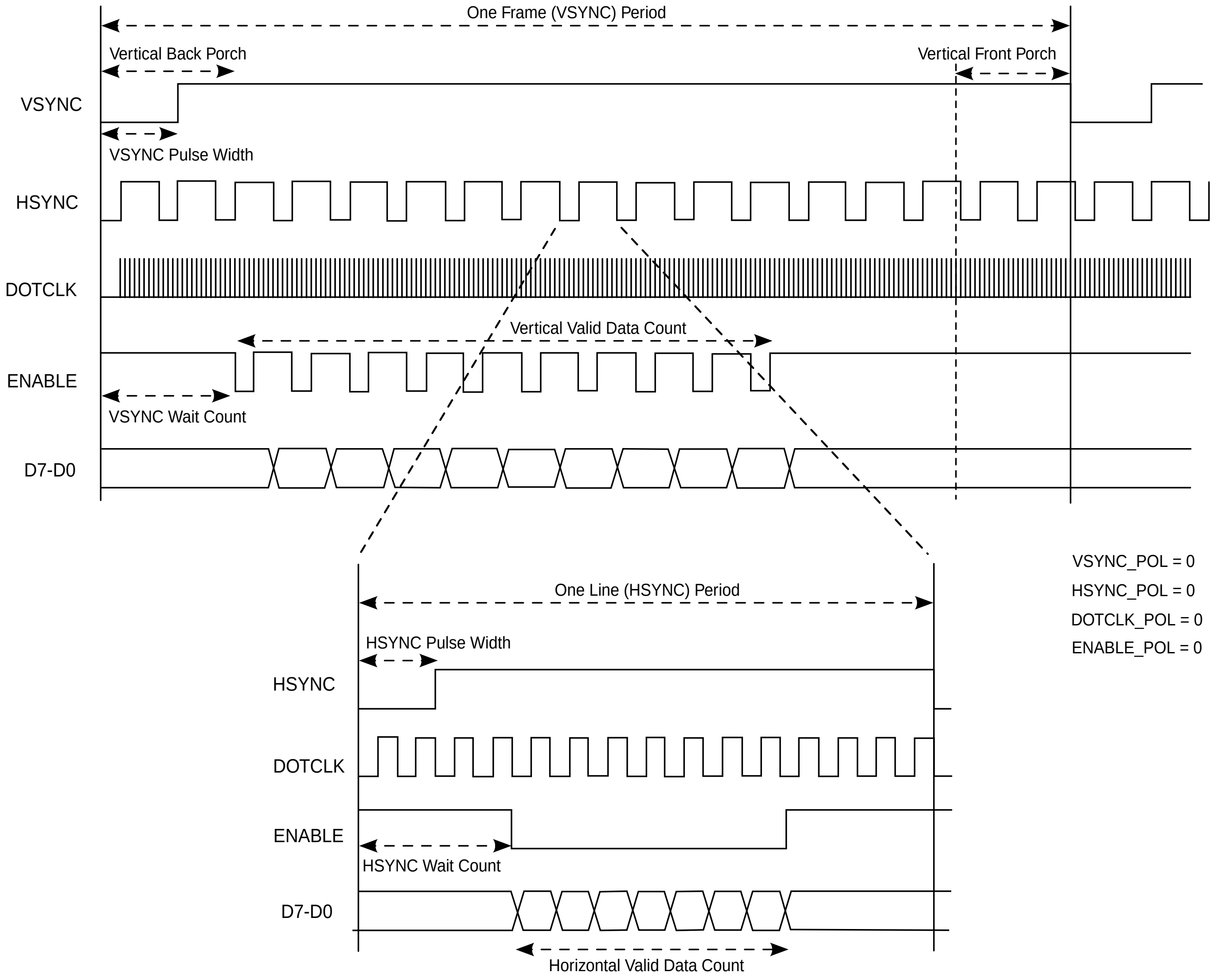


Figure 33-13. LCD Interface Signals in DOTCLK Mode

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC, ENABLE and DOTCLK signals. The polarities, periods and pulse-widths of the sync signals are programmable using the HW_LCDIF_VDCTRL0–4 registers. The units for the VSYNC signal must be number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and

VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The CLK_DIS_LCDIFn is controlled using the HW_CLKCTRL_PIX, HW_CLKCTRL_FRAC, and HW_CLKCTRL_CLKSEQ registers in the CLCKTRL block.

In DOTCLK mode, HW_LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DOTCLK_MODE bit 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and issue the cur_frame_done interrupt.

33.2.8.1 Code Example

The following code shows an example for programming a 320x240 display. Note that setting up the display must be done through the MPU mode or through SPI.

```
// Note: Common initialization steps in Initializing the LCDIF must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_PULSE_WIDTH, 2);
BF_CS1 (LCDIF_VDCTRL1, VSYNC_PERIOD, 280);
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 360); //Assuming LCD_DATABUS_WIDTH

                                                                   //   is 24bit
BF_CS2 (LCDIF_VDCTRL3, VSYNC_ONLY, 0);
BF_CS2 (LCDIF_VDCTRL3, HORIZONTAL_WAIT_CNT, 20, VERTICAL_WAIT_CNT, 20);
BF_CS1 (LCDIF_VDCTRL4, DOTCLK_H_VALID_DATA_CNT, 320);//Note that DOTCLK_V_VALID_DATA_CNT is
                                          //implicitly assumed to be
HW_LCDIF_TRANSFER_COUNT_V_COUNT
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit whatever it had in its FIFO, turn off the RUN bit and toggle the dma_end_cmd signal indicating to the DMA that it is done with the transfer.

33.2.9 ITU-R BT.656 Digital Video Interface (DVI)

ITU-R BT.656 Digital Video Interface shown below transmits 4:2:2 YCbCr digital component video to a digital video encoder that can translate it into 525/60 or 625/50 analog TV signal. Unique timing codes (timing reference signals) are embedded within the video stream to indicate the different timing events that would have been otherwise indicated by VSYNC, HSYNC and BLANK signals. The hardware supports 8-bit data transfers; the pins are shared with the lower 8 bits of LCD data bus. The LCD_RS pin is shared with the clock signal of the interface (called CCIRCLK here for uniqueness). CCIRCLK also can be obtained on the LCD_DOTCK pin. The mode shares the write FIFO with the LCD interface and the associated pipeline. The programmable parameters in registers HW_LCDIF_DVICTRL0-3 allow setting the total number of horizontal lines per frame, vertical and horizontal blanking interval, odd and even field start and end positions, and so on. In short, these parameters are provided to ensure that the hardware has enough flexibility to generate the right 525/60 or 625/50 data streams. Most of the initialization steps in Initializing the LCDIF such as data shifting, swizzle, and so on, are applicable to DVI mode also. The register descriptions in the programmable registers section at the end of this chapter include example code for programming the DVICTRL0-3 registers.

In DVI mode, HW_LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DVI_MODE bit the value 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and assert the cur_frame_done interrupt.

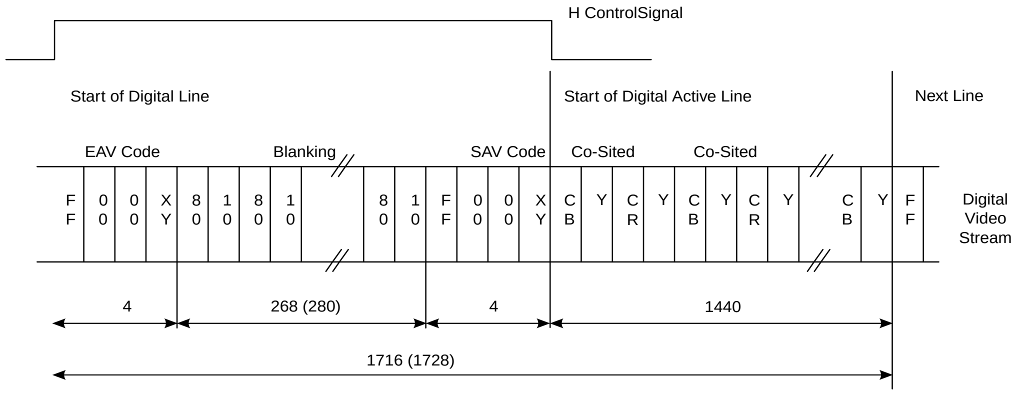


Figure 33-14. LCDIF Interface Signals in ITU-R BT.656 Digital Video Interface Mode

33.2.10 LCDIF Pin Usage by Interface Mode

The following table shows the pin usage for all of the supported modes when the PINCTRL registers are programmed for LCD functionality. See Pin Control and GPIO Overview for a more complete description of pin multiplexing options and how to program each pin individually.

The VSYNC signal has been mapped onto two pins, LCD_BUSY and LCD_VSYNC. The pin multiplexing can be programmed to select either of those pins to function as VSYNC.

NOTE

There is an option to internally mux the HSYNC, DOTCLK and ENABLE signals in the DOTCLK mode by setting the

MUX_SYNC_SIGNALS bit in the VDCTRL0 register. There is also an option to internally mux the LCD_WR_RWn and LCD_RD_E pins in the CTRL1 register for backward compatibility.

Table 33-1. Pin Usage in System Mode and VSYNC Mode

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_RS	LCD_ RS	LCD_ RS	LCD_ RS	LCD_RS	LCD_RS	LCD_ RS	LCD_RS	LCD_RS
LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS	LCD_CS	LCD_CS	LCD_ CS	LCD_CS
LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn	LCD_WR _RWn
LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E	LCD_RD_E
LCD_VSYNC* (Two options)	X	X	X	X	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC	LCD_ VSYNC
LCD_HSYNC	X	X	X	X	X	X	X	X
LCD_DOTCLK	X	X	X	X	X	X	X	X
LCD_EN-ABLE	X	X	X	X	X	X	X	X
LCD_D23	X	X	X	LCD_D23	X	X	X	LCD_D23
LCD_D22	X	X	X	LCD_D22	X	X	X	LCD_D22
LCD_D21	X	X	X	LCD_D21	X	X	X	LCD_D21
LCD_D20	X	X	X	LCD_D20	X	X	X	LCD_D20
LCD_D19	X	X	X	LCD_D19	X	X	X	LCD_D19
LCD_D18	X	X	X	LCD_D18	X	X	X	LCD_D18
LCD_D17	X	X	LCD_D17	LCD_D17	X	X	LCD_D17	LCD_D17
LCD_D16	X	X	LCD_D16	LCD_D16	X	X	LCD_D16	LCD_D16
LCD_D15 / VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	VSYNC (optional)	LCD_D15	VSYNC (optional)	LCD_D15
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X	LCD_D14	X	LCD_D14

PIN NAME	8-bit MPU LCD IF	16-bit MPU LCD IF	18-bit MPU LCD IF	24-bit MPU LCD IF	8-bit VSYNC LCD IF	16-bit VSYNC LCD IF	18-bit VSYNC LCD IF	24-bit VSYNC LCD IF
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X	LCD_D13	X	LCD_D13
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X	LCD_D12	X	LCD_D12
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X	LCD_D11	X	LCD_D11
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X	LCD_D10	X	LCD_D10
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X	LCD_D9	X	LCD_D9
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X	LCD_D8	X	LCD_D8
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET	LCD_RE-SET
LCD_BUSY / LCD_VSYNC	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)

Table 33-2. Pin Usage in DOTCLK Mode and DVI Mode

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_RS	X	X	X	X	CCIR_CLK
LCD_CS	X	X	X	X	X
LCD_WR_RWn	X	X	X	X	X
LCD_RD_E	X	X	X	X	X

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_VSYNC* (Two options)	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	X
LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	X
LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	X
LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	X
LCD_D23	X	X	X	LCD_D23	X
LCD_D22	X	X	X	LCD_D22	X
LCD_D21	X	X	X	LCD_D21	X
LCD_D20	X	X	X	LCD_D20	X
LCD_D19	X	X	X	LCD_D19	X
LCD_D18	X	X	X	LCD_D18	X
LCD_D17	X	X	LCD_D17	LCD_D17	X
LCD_D16	X	X	LCD_D16	LCD_D16	X
LCD_D15/ VSYNC*	X	LCD_D15	LCD_D15	LCD_D15	X
LCD_D14 / HSYNC**	X	LCD_D14	LCD_D14	LCD_D14	X
LCD_D13 / LCD_DOTCLK**	X	LCD_D13	LCD_D13	LCD_D13	X
LCD_D12 / ENABLE**	X	LCD_D12	LCD_D12	LCD_D12	X
LCD_D11	X	LCD_D11	LCD_D11	LCD_D11	X
LCD_D10	X	LCD_D10	LCD_D10	LCD_D10	X
LCD_D9	X	LCD_D9	LCD_D9	LCD_D9	X
LCD_D8	X	LCD_D8	LCD_D8	LCD_D8	X
LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7	LCD_D7
LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6	LCD_D6
LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5	LCD_D5
LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4	LCD_D4

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF	8-bit DVI LCD IF
LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3	LCD_D3
LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2	LCD_D2
LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1	LCD_D1
LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0	LCD_D0
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	X
LCD_BUSY / LCD_VSYNC	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	LCD_BUSY (OR optional LCD_VSYNC)	X

33.3 Behavior During Reset

HCLK and CLK_DIS_LCDIF must be running before making any changes to SFTRST or CLKGATE bits. A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See the section Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

33.4 Programmable Registers

LCDIF Hardware Register Format Summary

HW_LCDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8003_0000	LCDIF General Control Register (HW_LCDIF_CTRL)	32	R/W	C000_0000h	33.4.1/2082
8003_0010	LCDIF General Control1 Register (HW_LCDIF_CTRL1)	32	R/W	000F_0000h	33.4.2/2085
8003_0020	LCDIF General Control2 Register (HW_LCDIF_CTRL2)	32	R/W	0020_0000h	33.4.3/2088
8003_0030	LCDIF Horizontal and Vertical Valid Data Count Register (HW_LCDIF_TRANSFER_COUNT)	32	R/W	0001_0000h	33.4.4/2090
8003_0040	LCD Interface Current Buffer Address Register (HW_LCDIF_CUR_BUF)	32	R/W	0000_0000h	33.4.5/2091
8003_0050	LCD Interface Next Buffer Address Register (HW_LCDIF_NEXT_BUF)	32	R/W	0000_0000h	33.4.6/2091
8003_0060	LCD Interface Timing Register (HW_LCDIF_TIMING)	32	R/W	0000_0000h	33.4.7/2092

HW_LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8003_0070	LCDIF VSYNC Mode and Dotclk Mode Control Register0 (HW_LCDIF_VDCTRL0)	32	R/W	0000_0000h	33.4.8/2093
8003_0080	LCDIF VSYNC Mode and Dotclk Mode Control Register1 (HW_LCDIF_VDCTRL1)	32	R/W	0000_0000h	33.4.9/2094
8003_0090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (HW_LCDIF_VDCTRL2)	32	R/W	0000_0000h	33.4.10/2095
8003_00A0	LCDIF VSYNC Mode and Dotclk Mode Control Register3 (HW_LCDIF_VDCTRL3)	32	R/W	0000_0000h	33.4.11/2096
8003_00B0	LCDIF VSYNC Mode and Dotclk Mode Control Register4 (HW_LCDIF_VDCTRL4)	32	R/W	0000_0000h	33.4.12/2097
8003_00C0	Digital Video Interface Control0 Register (HW_LCDIF_DVICTRL0)	32	R/W	0000_0000h	33.4.13/2098
8003_00D0	Digital Video Interface Control1 Register (HW_LCDIF_DVICTRL1)	32	R/W	0000_0000h	33.4.14/2098
8003_00E0	Digital Video Interface Control2 Register (HW_LCDIF_DVICTRL2)	32	R/W	0000_0000h	33.4.15/2099
8003_00F0	Digital Video Interface Control3 Register (HW_LCDIF_DVICTRL3)	32	R/W	0000_0000h	33.4.16/2100
8003_0100	Digital Video Interface Control4 Register (HW_LCDIF_DVICTRL4)	32	R/W	0000_0000h	33.4.17/2101
8003_0110	RGB to YCbCr 4:2:2 CSC Coefficient0 Register (HW_LCDIF_CSC_COEFF0)	32	R/W	0000_0000h	33.4.18/2102
8003_0120	RGB to YCbCr 4:2:2 CSC Coefficient1 Register (HW_LCDIF_CSC_COEFF1)	32	R/W	0000_0000h	33.4.19/2103
8003_0130	RGB to YCbCr 4:2:2 CSC Coefficent2 Register (HW_LCDIF_CSC_COEFF2)	32	R/W	0000_0000h	33.4.20/2104
8003_0140	RGB to YCbCr 4:2:2 CSC Coefficient3 Register (HW_LCDIF_CSC_COEFF3)	32	R/W	0000_0000h	33.4.21/2105
8003_0150	RGB to YCbCr 4:2:2 CSC Coefficient4 Register (HW_LCDIF_CSC_COEFF4)	32	R/W	0000_0000h	33.4.22/2106
8003_0160	RGB to YCbCr 4:2:2 CSC Offset Register (HW_LCDIF_CSC_OFFSET)	32	R/W	0080_0010h	33.4.23/2107
8003_0170	RGB to YCbCr 4:2:2 CSC Limit Register (HW_LCDIF_CSC_LIMIT)	32	R/W	00FF_00FFh	33.4.24/2107
8003_0180	LCD Interface Data Register (HW_LCDIF_DATA)	32	R/W	0000_0000h	33.4.25/2109
8003_0190	Bus Master Error Status Register (HW_LCDIF_BM_ERROR_STAT)	32	R/W	0000_0000h	33.4.26/2109
8003_01A0	CRC Status Register (HW_LCDIF_CRC_STAT)	32	R/W	0000_0000h	33.4.27/2110
8003_01B0	LCD Interface Status Register (HW_LCDIF_STAT)	32	R	9500_0000h	33.4.28/2110
8003_01C0	LCD Interface Version Register (HW_LCDIF_VERSION)	32	R	0400_0000h	33.4.29/2112

HW_LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8003_01D0	LCD Interface Debug0 Register (HW_LCDIF_DEBUG0)	32	R	0E81_0000h	33.4.30/2112
8003_01E0	LCD Interface Debug1 Register (HW_LCDIF_DEBUG1)	32	R	0000_0000h	33.4.31/2114
8003_01F0	LCD Interface Debug2 Register (HW_LCDIF_DEBUG2)	32	R	0000_0000h	33.4.32/2115

33.4.1 LCDIF General Control Register (HW_LCDIF_CTRL)

The LCD Interface Control Register provides overall control of the LCDIF block.

HW_LCDIF_CTRL: 0x000

HW_LCDIF_CTRL_SET: 0x004

HW_LCDIF_CTRL_CLR: 0x008

HW_LCDIF_CTRL_TOG: 0x00C

The LCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

Address: HW_LCDIF_CTRL – 8003_0000h base + 0h offset = 8003_0000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	YCBCR422_INPUT	READ_WRITEB	WAIT_FOR_VSYNC_EDGE	DATA_SHIFT_DIR	SHIFT_NUM_BITS					DVI_MODE	BYPASS_COUNT	VSYNC_MODE	DOTCLK_MODE	DATA_SELECT
W	SFTRST	CLKGATE	YCBCR422_INPUT	READ_WRITEB	WAIT_FOR_VSYNC_EDGE	DATA_SHIFT_DIR	SHIFT_NUM_BITS					DVI_MODE	BYPASS_COUNT	VSYNC_MODE	DOTCLK_MODE	DATA_SELECT
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INPUT_DATA_SWIZZLE		CSC_DATA_SWIZZLE		LCD_DATABUS_WIDTH		WORD_LENGTH		RGB_TO_YCBCR422_CSC	Reserved	LCDIF_MASTER	RSRVD0	DATA_FORMAT_16_BIT	DATA_FORMAT_18_BIT	DATA_FORMAT_24_BIT	RUN
W	INPUT_DATA_SWIZZLE		CSC_DATA_SWIZZLE		LCD_DATABUS_WIDTH		WORD_LENGTH		RGB_TO_YCBCR422_CSC	▨	LCDIF_MASTER	▨	DATA_FORMAT_16_BIT	DATA_FORMAT_18_BIT	DATA_FORMAT_24_BIT	RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CTRL field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable normal operation of the LCDIF. When set to one, it forces a block level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 YCBCR422_ INPUT	Zero implies input data is in RGB color space. One implies input data is in YCbCr 4:2:2 format, such that YCbYCr are packed in a 32-bit word. It also means that there are 2 pixels in 4 bytes. If this bit is set, software should program the H_COUNT field in the TRANSFER_COUNT register to the total number of pixels that will have to be fetched by the LCDIF block per line and the BYTE_PACKING_FORMAT should be 0xF. The WORD_LENGTH does not matter in this case.
28 READ_WRITEB	By default, LCDIF is in the write mode. Setting this bit to 1 will make the hardware go into 6800/8080 system read mode. Make sure that DMA operation is selected (LCDIF_MASTER bit is 0) when performing read operations.
27 WAIT_FOR_ VSYNC_EDGE	Setting this bit to 1 will make the hardware wait for the triggering VSYNC edge before starting write transfers to the LCD. Used only in the VSYNC mode of operation.
26 DATA_SHIFT_ DIR	Use this bit to determine the direction of shift of transmit data. In the DVI mode, it works only on the active data, not on the timing codes and ancillary data. 0x0 **TXDATA_SHIFT_LEFT** — Data to be transmitted is shifted LEFT by SHIFT_NUM_BITS bits. 0x1 **TXDATA_SHIFT_RIGHT** — Data to be transmitted is shifted RIGHT by SHIFT_NUM_BITS bits.
25–21 SHIFT_NUM_ BITS	The data to be transmitted is shifted left or right by this number of bits.
20 DVI_MODE	Set this bit to 1 to get into theITU-R BT.656 digital video interface mode. Toggle this bit from 1 to 0 to make the hardware go out of DVI mode after completing all data transfer, deasserting the RUN bit and toggling the dma_end_cmd signal.
19 BYPASS_ COUNT	When this bit is 0, it means that LCDIF will stop the block operation and turn off the RUN bit after the amount of data indicated by the HW_LCDIF_TRANSFER_COUNT register has been transferred out. When this bit is set to 1, the block will continue normal operation indefinitely until it is told to stop. This bit must be 0 in system and VSYNC modes, and must be 1 in DOTCLK and DVI modes of operation.
18 VSYNC_MODE	Setting this bit to 1 will make the LCDIF hardware go into VSYNC mode. WAIT_FOR_VSYNC_EDGE can be used only if this bit is set. If VSYNC signal is required to be an output from the block, SYNC_SIGNALS_ON bit in HW_LCDIF_VDCTRL4 register must be set.
17 DOTCLK_MODE	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e. VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfer and deasserting the RUN bit.
16 DATA_SELECT	Command Mode polarity bit. This bit should only be changed when RUN is 0. 0x0 **CMD_MODE** — Command Mode. DCn signal is Low. 0x1 **DATA_MODE** — Data Mode. DCn signal is High.
15–14 INPUT_DATA_ SWIZZLE	This field specifies how to swap the bytes either in the HW_LCDIF_DATA register or those fetched by the AXI master part of LCDIF.The swizzle function is independent of the WORD_LENGTH bit. See the explanation of the HW_LCDIF_DATA below for names and definitions of data register fields. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian)

HW_LCDIF_CTRL field descriptions (continued)

Field	Description
	0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
13–12 CSC_DATA_ SWIZZLE	This field specifies how to swap the bytes after the data has been converted into an internal representation of 24 bits per pixel and before it is transmitted over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half word) first after the swizzle takes place. So, INPUT_DATA_SWIZZLE takes place first on the incoming data, and then CSC_DATA_SWIZZLE is applied. The swizzle function is independent of the WORD_LENGTH or the LCD_DATABUS_WIDTH fields. If RGB_TO_YCRCB422_CSC bit is set, the swizzle occurs on the Y, Cb, Cr values. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
11–10 LCD_DATABUS_ WIDTH	LCD Data bus transfer width. 0x0 **16_BIT** — 16-bit data bus mode. 0x1 **8_BIT** — 8-bit data bus mode. 0x2 **18_BIT** — 18-bit data bus mode. 0x3 **24_BIT** — 24-bit data bus mode.
9–8 WORD_LENGTH	Input data format. 0x0 **16_BIT** — Input data is 16 bits per pixel. 0x1 **8_BIT** — Input data is 8 bits wide. 0x2 **18_BIT** — Input data is 18 bits per pixel. 0x3 **24_BIT** — Input data is 24 bits per pixel.
7 RGB_TO_ YCBCR422_CSC	Set this bit to 1 to enable conversion from RGB to YCbCr colorspace. See the HW_LCDIF_CSC_ registers for further details.
6 Reserved	This bit is reserved. Reserved, always set to zero.
5 LCDIF_MASTER	Set this bit to make the LCDIF act as a bus master. If this bit is reset, the LCDIF will act in its traditional DMA slave mode.
4 RSRVD0	Reserved bits. Write as 0.
3 DATA_FORMAT_ 16_BIT	When this bit is 1 and WORD_LENGTH = 0, it implies that the the 16-bit data is in ARGB555 format. When this bit is 0 and WORD_LENGTH = 0, it implies that the 16-bit data is in RGB565 format. When WORD_LENGTH is not 0, this bit is a dont care.
2 DATA_FORMAT_ 18_BIT	Used only when WORD_LENGTH = 2, i.e. 18-bit.

HW_LCDIF_CTRL field descriptions (continued)

Field	Description
	0x0 **LOWER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that lower 18 bits contain RGB 666 and upper 14 bits do not contain any useful data. 0x1 **UPPER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that upper 18 bits contain RGB 666 and lower 14 bits do not contain any useful data.
1 DATA_FORMAT_24_BIT	Used only when WORD_LENGTH = 3, i.e. 24-bit. Note that this applies to both packed and unpacked 24-bit data. 0x0 **ALL_24_BITS_VALID** — Data input to the block is in 24 bpp format, such that all RGB 888 data is contained in 24 bits. 0x1 **DROP_UPPER_2_BITS_PER_BYTE** — Data input to the block is actually RGB 18 bpp, but there is 1 color per byte, hence the upper 2 bits in each byte do not contain any useful data, and should be dropped.
0 RUN	When this bit is set by software, the LCDIF will start fetching data in either the DMA mode or the bus master mode and sending it across the interface. This bit must remain set for all the time the block is in operation.

33.4.2 LCDIF General Control1 Register (HW_LCDIF_CTRL1)

The LCDIF Control Register provides overall control of the LCDIF block.

HW_LCDIF_CTRL1: 0x010

HW_LCDIF_CTRL1_SET: 0x014

HW_LCDIF_CTRL1_CLR: 0x018

HW_LCDIF_CTRL1_TOG: 0x01C

The LCDIF Control1 Register provides additional programming to the LCDIF. It implements some bits which are unlikely to change often in a particular application. It also carries interrupt-related bits which are common across more than one mode of operation.

Address: HW_LCDIF_CTRL1 – 8003_0000h base + 10h offset = 8003_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1				COMBINE_MPU_WR_STRB	BM_ERROR_IRQ_EN	BM_ERROR_IRQ	RECOVER_ON_UNDERFLOW	INTERLACE_FIELDS	START_INTERLACE_FROM_SECOND_FIELD	FIFO_CLEAR	IRQ_ON_ALTERNATE_FIELDS	BYTE_PACKING_FORMAT			
W	▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OVERFLOW_IRQ_EN	UNDERFLOW_IRQ_EN	CUR_FRAME_DONE_IRQ_EN	VSYNC_EDGE_IRQ_EN	OVERFLOW_IRQ	UNDERFLOW_IRQ	CUR_FRAME_DONE_IRQ	VSYNC_EDGE_IRQ	RSRVD0					BUSY_ENABLE	MODE86	RESET
W									▨	▨	▨	▨	▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CTRL1 field descriptions

Field	Description
31–28 RSRVD1	Reserved bits. Write as 0.
27 COMBINE_MPU_ WR_STRB	If this bit is not set, the write strobe will be driven on LCD_WR_RWn pin in the 8080 mode and on the LCD_RD_E pin in the 6800 mode. If it is set, the write strobe of both the 6800 and 8080 modes will be driven only on the LCD_WR_RWn pin. Note that this does not work for read strobe.
26 BM_ERROR_ IRQ_EN	This bit is set to enable bus master error interrupt in the LCDIF master mode.
25 BM_ERROR_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. This bit will be set when the LCDIF is in master mode and an error response was returned by the slave. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
24 RECOVER_ON_ UNDERFLOW	Set this bit to enable the LCDIF block to recover in the next field/frame if there was an underflow in the current field/frame.
23 INTERLACE_ FIELDS	Set this bit if it is required that the LCDIF block fetches odd lines in one field and even lines in the other field. It will work only in LCDIF_MASTER is set to 1.
22 START_ INTERLACE_ FROM_ SECOND_FIELD	The default is to grab the odd lines first and then the even lines. Set this bit if it is required to grab the even lines first and then the odd lines. (Line numbers start from 1, so odd lines are 1,3,5,etc. and even lines are 2,4,6, etc.)
21 FIFO_CLEAR	Set this bit to clear all the data in the latency FIFO (LFIFO), TXFIFO and the RXFIFO.
20 IRQ_ON_ ALTERNATE_ FIELDS	If this bit is set, the LCDIF block will assert the cur_frame_done interrupt only on alternate fields, otherwise it will issue the interrupt on both odd and even field. This bit is mostly relevant if INTERLACE_FIELDS is set. This feature is only available in DOTCLK and DVI modes.
19–16 BYTE_ PACKING_ FORMAT	This bitfield is used to show which data bytes in the 32-bit word written in the HW_LCDIF_DATA register are valid and should be transmitted. Default value 0xf indicates that all bytes are valid. For 8-bit transfers, any combination in this bitfield will mean valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 will mean that none of the bytes are valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (A-R-G-B where A value does not have be transmitted). When input data is is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of pixels that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF.(Note - YCBCR422_INPUT = 1 implies 2 pixels per 32 bits).

HW_LCDIF_CTRL1 field descriptions (continued)

Field	Description
15 OVERFLOW_ IRQ_EN	This bit is set to enable an overflow interrupt in the TXFIFO in the write mode.
14 UNDERFLOW_ IRQ_EN	This bit is set to enable an underflow interrupt in the TXFIFO in the write mode.
13 CUR_FRAME_ DONE_IRQ_EN	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12 VSYNC_EDGE_ IRQ_EN	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in the VSYNC and DOTCLK modes, or the beginning of every field in DVI mode.
11 OVERFLOW_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A latency FIFO (LFIFO) overflow in the write mode (system/VSYNC/DOTCLK/DVI mode) was detected, data samples have been lost. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
10 UNDERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A TXFIFO underflow in the write mode (system/VSYNC/DOTCLK/DVI mode) was detected. Could produce an error in the DOTCLK / DVI modes. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
9 CUR_FRAME_ DONE_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the VSYNC and system modes, this IRQ is asserted at the end of the data transfer indicated by HW_LCDIF_TRANSFER_COUNT register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
8 VSYNC_EDGE_ IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It is set whenever the leading VSYNC edge is detected in the VSYNC and DOTCLK modes. In the DVI mode, it is asserted every time the block enters a new field. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
7–3 RSRVD0	Reserved bits. Write as 0.
2 BUSY_ENABLE	This bit enables the use of the interface's busy signal input. This should be enabled for LCD controllers that implement a busy line (to stall the LCDIF from sending more data until ready). Otherwise this bit should be cleared. 0x0 **BUSY_DISABLED** — The busy signal from the LCD controller will be ignored. 0x1 **BUSY_ENABLED** — Enable the use of the busy signal from the LCD controller.
1 MODE86	This bit is used to select between the 8080 and 6800 series of microprocessor modes. This bit should only be changed when RUN is 0.

HW_LCDIF_CTRL1 field descriptions (continued)

Field	Description
	0x0 **8080_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as active low WR and active low RD signals respectively. 0x1 **6800_MODE** — Pins LCD_WR_RWn and LCD_RD_E function as Read/Writeb and active high Enable signals respectively.
0 RESET	Reset bit for the external LCD controller. This bit can be changed at any time. It CANNOT be reset by SFTRST. 0x0 **LCDRESET_LOW** — LCD_RESET output signal is low. 0x1 **LCDRESET_HIGH** — LCD_RESET output signal is high.

33.4.3 LCDIF General Control2 Register (HW_LCDIF_CTRL2)

The LCDIF Control Register provides overall control of the LCDIF block.

HW_LCDIF_CTRL2: 0x020

HW_LCDIF_CTRL2_SET: 0x024

HW_LCDIF_CTRL2_CLR: 0x028

HW_LCDIF_CTRL2_TOG: 0x02C

The LCDIF Control2 Register provides additional programming to the LCDIF. It implements some bits which are unlikely to change often in a particular application.

Address: HW_LCDIF_CTRL2 – 8003_0000h base + 20h offset = 8003_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD5								OUTSTANDING_REQS			BURST_LEN_8	RSRVD4	ODD_LINE_PATTERN		
W	▨	▨	▨	▨	▨	▨	▨	▨					▨			
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD3	EVEN_LINE_PATTERN			RSRVD2	READ_PACK_DIR	READ_MODE_OUTPUT_IN_RGB_FORMAT	READ_MODE_6_BIT_INPUT	RSRVD1	READ_MODE_NUM_PACKED_SUBWORDS			INITIAL_DUMMY_READ			RSRVD0
W	▨				▨				▨							▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CTRL2 field descriptions

Field	Description
31–24 RSRVD5	Reserved bits. Write as 0.
23–21 OUTSTANDING_ REQS	This bitfield indicates the maximum number of outstanding transactions that LCDIF should request when it is acting as a bus master. Default is 2 outstanding transactions. 0x0 **REQ_1** — 0x1 **REQ_2** — 0x2 **REQ_4** — 0x3 **REQ_8** — 0x4 **REQ_16** —
20 BURST_LEN_8	By default, when the LCDIF is in the bus master mode, it will issue AXI bursts of length 16 (except when in packed 24 bpp mode, it will issue bursts of length 15). When this bit is set to 1, the block will issue bursts of length 8 (except when in packed 24 bpp mode, it will issue bursts of length 9). Note that this bitfield is only applicable when LCDIF_MASTER is set to 1.
19 RSRVD4	Reserved bits. Write as 0.
18–16 ODD_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in ODD lines (line numbers 1,3,5,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
15 RSRVD3	Reserved bits. Write as 0.
14–12 EVEN_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in EVEN lines (line numbers 2,4,6,..). This bitfield must be 0 in DVI mode. 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
11 RSRVD2	Reserved bits. Write as 0.
10 READ_PACK_DIR	The default value of 0 indicates data is stored in the little endian format. When LCD_DATABUS_WIDTH is 8-bit, this bit provides the option of rearranging the data byte-wise in the big endian format. For example, if READ_MODE_NUM_PACKED_SUBWORDS = 3 and the order of incoming data is 0x11, 0x22 and 0x33, then setting this bit to 1 will cause the data to be stored as 0x00112233 as opposed to the default 0x00332211. This operation occurs after the shifting operation done by SHIFT_NUM_BITS bitfield.
9 READ_MODE_ OUTPUT_IN_ RGB_FORMAT	Setting this bit will enable the LCDIF to convert the incoming data to the RGB format given by WORD_LENGTH bitfield. This feature is not available when WORD_LENGTH is set to 8 bits. LCDIF performs this operation of converting to RGB format after the endianness has been determined by the READ_PACK_DIR bitfield.

HW_LCDIF_CTRL2 field descriptions (continued)

Field	Description
8 READ_MODE_6_ BIT_INPUT	Setting this bit to 1 indicates to LCDIF that even though LCD_DATABUS_WIDTH is set to 8 bits, the input data is actually only 6 bits wide and exists on D5-D0.
7 RSRVD1	Reserved bits. Write as 0.
6–4 READ_MODE_ NUM_PACKED_ SUBWORDS	Indicates the number of valid 8/16/18/24-bit subwords that will be packed in the 32-bit HW_LCDIF_DATA register in the read mode. The subword size (8,16, 18 or 24 bits) is determined by the LCD_DATABUS_WIDTH field. The swizzle operation is performed after READ_MODE_NUM_PACKED_SUBWORDS number of data has been received from the interface and stored in the little-endian format. For example, if LCD_DATABUS_WIDTH is set to 8-bit and data to be read back has to be stored in memory in 24-bit unpacked RGB format, set READ_MODE_NUM_PACKED_SUBWORDS to 0x3 so that each 32-bit word will contain only 3 valid bytes (RGB). Maximum value of READ_MODE_NUM_PACKED_SUBWORDS is 4 for 8-bit databus, 2 for 16-bit databus and 1 for 18/24-bit databus.
3–1 INITIAL_DUMMY_ READ	The value in this field determines the number of dummy 8/16/18/24-bit subwords that have to be read back from the LCD panel/controller. They will then not be stored in the read FIFO.
0 RSRVD0	Reserved bits. Write as 0.

33.4.4 LCDIF Horizontal and Vertical Valid Data Count Register (HW_LCDIF_TRANSFER_COUNT)

This register tells the LCDIF how much data will be sent for this frame, or transaction. The total number of words is a product of the V_COUNT and H_COUNT fields. The word size is specified by the WORD_LENGTH field.

This register gives the dimensions of the input frame. For normal operation, but V_COUNT and H_COUNT should be non-zero.

Address: HW_LCDIF_TRANSFER_COUNT – 8003_0000h base + 30h offset = 8003_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_COUNT																H_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_TRANSFER_COUNT field descriptions

Field	Description
31–16 V_COUNT	Number of horizontal lines per frame which contain valid data. In DOTCLK mode, V_COUNT should be the same as the number of active horizontal lines in a progressive frame. In DVI mode, V_COUNT should be the number of active horizontal lines per frame, and not per field.

HW_LCDIF_TRANSFER_COUNT field descriptions (continued)

Field	Description
15–0 H_COUNT	Total valid data (pixels) in each horizontal line. The data size is given by the WORD_LENGTH. When input data is is in YCbCr 4:2:2 format (YCBCR422_INPUT is 1), H_COUNT should be the number of 32-bit words that should be fetched by the block and the BYTE_PACKING_FORMAT should be 0xF. In 24-bit packed format (WORD_LENGTH=0x3, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 4 pixels. In 16-bit packed format (WORD_LENGTH=0x0, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 2 pixels.

33.4.5 LCD Interface Current Buffer Address Register (HW_LCDIF_CUR_BUF)

This register indicates the address of the current frame that is being transmitted by LCDIF.

When the LCDIF is behaving as a master, this address points to the address of the current frame of data being sent out through the LCDIF. When the current frame is done, the LCDIF block will assert the cur_frame_done interrupt for software to take action. The block will also copy the HW_LCDIF_NEXT_BUF_ADDR into this bitfield so that the software can program the next frame address into the HW_LCDIF_NEXT_BUF_ADDR bitfield. This address must always be double-word aligned.

Address: HW_LCDIF_CUR_BUF – 8003_0000h base + 40h offset = 8003_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CUR_BUF field descriptions

Field	Description
31–0 ADDR	start address of current buffer

33.4.6 LCD Interface Next Buffer Address Register (HW_LCDIF_NEXT_BUF)

This register indicates the address of next frame that will be transmitted by LCDIF.

When the LCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out through the LCDIF. It is upto the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the LCDIF. This address must always be double-word aligned.

Address: HW_LCDIF_NEXT_BUF – 8003_0000h base + 50h offset = 8003_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_NEXT_BUF field descriptions

Field	Description
31–0 ADDR	start address of next buffer

33.4.7 LCD Interface Timing Register (HW_LCDIF_TIMING)

The LCD interface timing register controls the various setup and hold times enforced by the LCD interface in the 6800/8080 system and VSYNC modes of operation.

The values used in this register are dependent on the particular LCD controller used, consult the users manual for the particular controller for required timings. Each field of the register must be non-zero, therefore the minimum value is: 0x01010101. NOTE: the timings are not automatically adjusted if the CLK_DIS_LCDIFn frequency changes--it may be necessary to adjust the timings if CLK_DIS_LCDIFn changes. NOTE: Each field in this register must be non-zero for the system and VSYNC modes to function. The settings in this register do not affect the DOTCLK and DVI modes.

Address: HW_LCDIF_TIMING – 8003_0000h base + 60h offset = 8003_0060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMD_HOLD								CMD_SETUP								DATA_HOLD								DATA_SETUP							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_TIMING field descriptions

Field	Description
31–24 CMD_HOLD	Number of CLK_DIS_LCDIFn cycles that the DCn signal is active after CEn is deasserted.

HW_LCDIF_TIMING field descriptions (continued)

Field	Description
23–16 CMD_SETUP	Number of CLK_DIS_LCDIFn cycles that the the DCn signal is active before CEn is asserted.
15–8 DATA_HOLD	Data bus hold time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is de-asserted in a cycle
7–0 DATA_SETUP	Data bus setup time in CLK_DIS_LCDIFn cycles. Also the time that the data strobe is asserted in a cycle.

33.4.8 LCDIF VSYNC Mode and Dotclk Mode Control Register0 (HW_LCDIF_VDCTRL0)

This register is used to control the VSYNC and DOTCLK modes of the LCDIF so as to work with different types of LCDs like moving picture displays and delta pixel displays.

HW_LCDIF_VDCTRL0: 0x070

HW_LCDIF_VDCTRL0_SET: 0x074

HW_LCDIF_VDCTRL0_CLR: 0x078

HW_LCDIF_VDCTRL0_TOG: 0x07C

This register gives general programmability to the VSYNC signal including polarity, direction, pulse width, etc.

Address: HW_LCDIF_VDCTRL0 – 8003_0000h base + 70h offset = 8003_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2		VSYNC_OEB	ENABLE_PRESENT	VSYNC_POL	HSYNC_POL	DOTCLK_POL	ENABLE_POL	RSRVD1		VSYNC_PERIOD_UNIT	VSYNC_PULSE_WIDTH_UNIT	HALF_LINE	HALF_LINE_MODE	VSYNC_PULSE_WIDTH[17:16]	
W	▨	▨							▨	▨						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VSYNC_PULSE_WIDTH[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VDCTRL0 field descriptions

Field	Description
31–30 RSRVD2	Reserved bits. Write as 0.

HW_LCDIF_VDCTRL0 field descriptions (continued)

Field	Description
29 VSYNC_OEB	0 means the VSYNC signal is an output, 1 means it is an input. Should be set to 0 in the DOTCLK mode. 0x0 **VSYNC_OUTPUT** — The VSYNC pin is in the output mode and the VSYNC signal has to be generated by the LCDIF block. 0x1 **VSYNC_INPUT** — The VSYNC pin is in the input mode and the LCD controller sends the VSYNC signal to the block.
28 ENABLE_ PRESENT	Setting this bit to 1 will make the hardware generate the ENABLE signal in the DOTCLK mode, thereby making it the true RGB interface along with the remaining three signals VSYNC, HSYNC and DOTCLK.
27 VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26 HSYNC_POL	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25 DOTCLK_POL	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24 ENABLE_POL	Default 0 active low during valid data transfer on each horizontal line.
23–22 RSRVD1	Reserved bits. Write as 0.
21 VSYNC_ PERIOD_UNIT	Default 0 for counting VSYNC_PERIOD in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines. CLK_DIS_LCDIFn cycles should be used in the VSYNC mode, while horizontal line should be used in the DOTCLK mode.
20 VSYNC_PULSE_ WIDTH_UNIT	Default 0 for counting VSYNC_PULSE_WIDTH in terms of CLK_DIS_LCDIFn cycles. Set it to 1 to count in terms of complete horizontal lines.
19 HALF_LINE	Setting this bit to 1 will make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e. VSYNC_PERIOD field plus half horizontal line), otherwise it is just VSYNC_PERIOD. Should be only used in the DOTCLK mode, not in the VSYNC interface mode.
18 HALF_LINE_ MODE	When this bit is 0, the first field (VSYNC period) will end in half a horizontal line and the second field will begin with half a horizontal line. When this bit is 1, all fields will end with half a horizontal line, and none will begin with half a horizontal line.
17–0 VSYNC_PULSE_ WIDTH	Number of units for which VSYNC signal is active. For the DOTCLK mode, the unit is determined by the VSYNC_PULSE_WIDTH_UNIT. If the VSYNC_PULSE_WIDTH_UNIT is 0 for DOTCLK mode, VSYNC_PULSE_WIDTH must be less than HSYNC_PERIOD. For the VSYNC interface mode, it should be in terms of number of CLK_DIS_LCDIFn cycles only.

33.4.9 LCDIF VSYNC Mode and Dotclk Mode Control Register1 (HW_LCDIF_VDCTRL1)

This register is used to control the VSYNC signal in the VSYNC and DOTCLK modes of the block.

This register determines the period and duty cycle of the VSYNC signal when it is generated in the block.

Address: HW_LCDIF_VDCTRL1 – 8003_0000h base + 80h offset = 8003_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VSYNC_PERIOD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VDCTRL1 field descriptions

Field	Description
31–0 VSYNC_PERIOD	Total number of units between two positive or two negative edges of the VSYNC signal. If HALF_LINE is set, it is implicitly calculated to be VSYNC_PERIOD plus half HSYNC_PERIOD.

33.4.10 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (HW_LCDIF_VDCTRL2)

This register is used to control the HSYNC signal in the DOTCLK mode of the block.

This register determines the period and duty cycle of the HSYNC signal when it is generated in the block.

Address: HW_LCDIF_VDCTRL2 – 8003_0000h base + 90h offset = 8003_0090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSYNC_PULSE_WIDTH														HSYNC_PERIOD																	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VDCTRL2 field descriptions

Field	Description
31–18 HSYNC_PULSE_WIDTH	Number of CLK_DIS_LCDIFn cycles for which HSYNC signal is active.
17–0 HSYNC_PERIOD	Total number of CLK_DIS_LCDIFn cycles between two positive or two negative edges of the HSYNC signal.

33.4.11 LCDIF VSYNC Mode and Dotclk Mode Control Register3 (HW_LCDIF_VDCTRL3)

This register is used to determine the vertical and horizontal wait counts.

This register determines the back porches of HSYNC and VSYNC signals when they are generated by the block.

Address: HW_LCDIF_VDCTRL3 – 8003_0000h base + A0h offset = 8003_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0		MUX_SYNC_SIGNALS	VSYNC_ONLY	HORIZONTAL_WAIT_CNT											
W	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VERTICAL_WAIT_CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VDCTRL3 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29 MUX_SYNC_SIGNALS	When this bit is set, the LCDIF block will internally mux HSYNC with LCD_D14, DOTCLK with LCD_D13 and ENABLE with LCD_D12, otherwise these signals will go out on separate pins. This feature can be used to maintain backward compatability with 37xx.
28 VSYNC_ONLY	This bit must be set to 1 in the VSYNC mode of operation, and 0 in the DOTCLK mode of operation.
27–16 HORIZONTAL_ WAIT_CNT	In the DOTCLK mode, wait for this number of clocks from falling edge (or rising if HSYNC_POL is 1) of HSYNC signal to account for horizontal back porch plus the number of DOTCLKs before the moving picture information begins.
15–0 VERTICAL_ WAIT_CNT	In the VSYNC interface mode, wait for this number of CLK_DIS_LCDIFn cycles from the falling VSYNC edge (or rising if VSYNC_POL is 1) before starting LCD transactions and is applicable only if WAIT_FOR_VSYNC_EDGE is set. Minimum is CMD_SETUP+5. In the DOTCLK mode, it accounts for the veritcal back porch lines plus the number of horizontal lines before the moving picture begins. The unit for this parameter is inherently the same as the VSYNC_PERIOD_UNIT.

33.4.12 LCDIF VSYNC Mode and Dotclk Mode Control Register4 (HW_LCDIF_VDCTRL4)

This register is used to control the DOTCLK mode of the block.

This register determines the active data in each horizontal line in the DOTCLK mode. Note that the total number of active horizontal lines in the DOTCLK mode is the same as the V_COUNT bitfield in the HW_LCDIF_TRANSFER_COUNT register.

Address: HW_LCDIF_VDCTRL4 – 8003_0000h base + B0h offset = 8003_00B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DOTCLK_DLY_SEL			RSRVD0										SYNC_ SIGNALS_ON	DOTCLK_H_ VALID_ DATA_CNT [17:16]	
W				▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DOTCLK_H_VALID_DATA_CNT[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VDCTRL4 field descriptions

Field	Description
31–29 DOTCLK_DLY_ SEL	This bitfield selects the amount of time by which the DOTCLK signal should be delayed before coming out of the LCD_DOTCK pin. 0 = 2ns; 1=4ns;2=6ns;3=8ns. Remaining values are reserved.
28–19 RSRVD0	Reserved bits, write as 0.
18 SYNC_ SIGNALS_ON	Set this field to 1 if the LCD controller requires that the VSYNC or VSYNC/HSYNC/DOTCLK control signals should be active atleast one frame before the data transfers actually start and remain active atleast one frame after the data transfers end. The hardware does not count the number of frames automatically. Rather, the VSYNC edge interrupt can be monitored by software to count the number of frames that have occured after this bit is set and then the RUN bit can be set to start the data transactions. This bit must always be set in the DOTCLK mode of operation, and it must be set in the VSYNC mode of operation when VSYNC signal is an output.
17–0 DOTCLK_H_ VALID_DATA_ CNT	Total number of CLK_DIS_LCDIFn cycles on each horizontal line that carry valid data in DOTCLK mode.

33.4.13 Digital Video Interface Control0 Register (HW_LCDIF_DVICTRL0)

The Digital Video interface Control0 register provides the overall control of the Digital Video interface.

This register gives information about the horizontal active, horizontal blanking and total number of lines in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x106);//262
//625/50 video system
HW_LCDIF_DVICTRL0_H_ACTIVE_CNT_WR(0x5A0);//1440
HW_LCDIF_DVICTRL0_H_BLANKING_CNT_WR(0x112);//274
```

Address: HW_LCDIF_DVICTRL0 – 8003_0000h base + C0h offset = 8003_00C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1				H_ACTIVE_CNT												RSRVD0				H_BLANKING_CNT											
W	▨	▨	▨	▨													▨	▨	▨	▨												
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DVICTRL0 field descriptions

Field	Description
31–28 RSRVD1	Reserved bits, write as 0.
27–16 H_ACTIVE_CNT	Number of active video samples to be transmitted. (Mostly will be 1440 for both PAL and NTSC). Must always be a multiple of 4.
15–12 RSRVD0	Reserved bits, write as 0.
11–0 H_BLANKING_CNT	Number of blanking samples to be inserted between EAV and SAV during horizontal blanking interval.

33.4.14 Digital Video Interface Control1 Register (HW_LCDIF_DVICTRL1)

The Digital Video interface Control1 register provides the overall control of the Digital Video interface.

This register contains information about the Field1 start and end, and the Field2 start in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x4);//4
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x109);//265
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x10A);//266
//625/50 video system
HW_LCDIF_DVICTRL1_F1_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL1_F1_END_LINE_WR(0x138);//312
HW_LCDIF_DVICTRL1_F2_START_LINE_WR(0x139);//313
```

Address: HW_LCDIF_DVICTRL1 – 8003_0000h base + D0h offset = 8003_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0		F1_START_LINE										F1_END_LINE[19:16]			
W	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	F1_END_LINE[15:10]						F2_START_LINE									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DVICTRL1 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 F1_START_LINE	Vertical line number from which Field 1 begins.
19–10 F1_END_LINE	Vertical line number at which Field1 ends.
9–0 F2_START_LINE	Vertical line number from which Field 2 begins.

33.4.15 Digital Video Interface Control2 Register (HW_LCDIF_DVICTRL2)

The Digital Video interface Control2 register provides the overall control of the Digital Video interface.

This register contains information about the Field2 end, and the Vertical Blanking1 interval in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x3);//3
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x108);//264
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x11A);//282
//625/50 video system
HW_LCDIF_DVICTRL2_F2_END_LINE_WR(0x271);//625
HW_LCDIF_DVICTRL2_V1_BLANK_START_LINE_WR(0x137);//311
HW_LCDIF_DVICTRL2_V1_BLANK_END_LINE_WR(0x14F);//335
```

Address: HW_LCDIF_DVICTRL2 – 8003_0000h base + E0h offset = 8003_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0		F2_END_LINE										V1_BLANK_START_LINE [19:16]			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	V1_BLANK_START_LINE[15:10]						V1_BLANK_END_LINE									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DVICTRL2 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 F2_END_LINE	Vertical line number at which Field 2 ends.
19–10 V1_BLANK_ START_LINE	Vertical line number towards the end of Field1 where first Vertical Blanking interval starts.
9–0 V1_BLANK_ END_LINE	Vertical line number in the beginning part of Field2 where first Vertical Blanking interval ends.

33.4.16 Digital Video Interface Control3 Register (HW_LCDIF_DVICTRL3)

The Digital Video interface Control3 register provides the overall control of the Digital Video interface.

This register contains information about the Vertical Blanking2 interval in the ITU-R BT.656 interface.

EXAMPLE

```
//525/60 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x1);//1
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x13);//19
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x20D);//525
//625/50 video system
HW_LCDIF_DVICTRL3_V2_BLANK_START_LINE_WR(0x270);//624
HW_LCDIF_DVICTRL3_V2_BLANK_END_LINE_WR(0x16);//22
HW_LCDIF_DVICTRL0_V_LINES_CNT_WR(0x271);//625
```

Address: HW_LCDIF_DVICTRL3 – 8003_0000h base + F0h offset = 8003_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD0		V2_BLANK_START_LINE										V2_BLANK_END_LINE [19:16]			
W	▨	▨														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	V2_BLANK_END_LINE[15:10]						V_LINES_CNT									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DVICTRL3 field descriptions

Field	Description
31–30 RSRVD0	Reserved bits, write as 0.
29–20 V2_BLANK_ START_LINE	Vertical line number towards the end of Field2 where second Vertical Blanking interval starts.
19–10 V2_BLANK_ END_LINE	Vertical line number in the beginning part of Field1 where second Vertical Blanking interval ends.
9–0 V_LINES_CNT	Total number of vertical lines per frame (generally 525 or 625)

33.4.17 Digital Video Interface Control4 Register (HW_LCDIF_DVICTRL4)

The Digital Video interface Control4 register provides the overall control of the Digital Video interface.

This register is used to add side borders to the output if the input frame width is less than 720 pixels.

EXAMPLE

```
        //If input frame has only 640 pixels per line, but output is supposed to have 720
pixels per line.
        HW_LCDIF_DVICTRL4_H_FILL_CNT_WR(0x50);//80
        HW_LCDIF_DVICTRL4_Y_FILL_VALUE_WR(0x10);//16
        HW_LCDIF_DVICTRL4_CB_FILL_VALUE_WR(0x80);//128
        HW_LCDIF_DVICTRL4_CR_FILL_VALUE_WR(0x80);//128
```

Address: HW_LCDIF_DVICTRL4 – 8003_0000h base + 100h offset = 8003_0100h

Bit	31–24	23–16	15–8	7–0
R/W	Y_FILL_VALUE	CB_FILL_VALUE	CR_FILL_VALUE	H_FILL_CNT
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

HW_LCDIF_DVICTRL4 field descriptions

Field	Description
31–24 Y_FILL_VALUE	Value of Y component of filler data
23–16 CB_FILL_VALUE	Value of CB component of filler data
15–8 CR_FILL_VALUE	Value of CR component of filler data.
7–0 H_FILL_CNT	Number of active video samples that have to be filled with the filler data in the front and back portions of the active horizontal interval. Must be a multiple of 4. This field will have to be programmed if the input frame has less than 720 pixels per line.

33.4.18 RGB to YCbCr 4:2:2 CSC Coefficient0 Register (HW_LCDIF_CSC_COEFF0)

HW_LCDIF_CSC_COEFF0 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by:

Y = C0*R + C1*G + C2*B + Y_offset

Cb= C3*R + C4*G + C5*B + CbCr_offset

Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
        HW_LCDIF_CSC_COEFF0_C0_WR(0x41);//0.257x256=65
        HW_LCDIF_CSC_COEFF0_CSC_SUBSAMPLE_FILTER_WR(0x3);
```

Address: HW_LCDIF_CSC_COEFF0 – 8003_0000h base + 110h offset = 8003_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1						C0									
W	▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0														CSC_ SUBSAMPLE_ FILTER	
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CSC_COEFF0 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C0	Two's complement red multiplier coefficient for Y
15–2 RSRVD0	Reserved bits, write as 0.
1–0 CSC_ SUBSAMPLE_ FILTER	This register describes the filtering and subsampling scheme to be performed on the chroma components in order to convert from YCbCr 4:4:4 to YCbCr 4:2:2 space. Note that the following descriptions apply individually to Cb and Cr. 0x0 **SAMPLE_AND_HOLD** — No filtering, simply keep every chroma value for samples numbered 2n and discard chroma values associated with all samples numbered 2n+1. 0x1 **RSRVD** — Reserved 0x2 **INTERSTITIAL** — Chroma samples numbered 2n and 2n+1 are averaged (weights 1/2, 1/2) and that chroma value replaces the two chroma values at 2n and 2n+1. This chroma now exists horizontally halfway between the two luma samples. 0x3 **COSITED** — Chroma samples numbered 2n-1, 2n, and 2n+1 are averaged (weights 1/4,1/2,1/4) and that chroma value exists at the same site as the luma sample numbered 2n and the chroma samples at 2n+1 are discarded.

33.4.19 RGB to YCbCr 4:2:2 CSC Coefficient1 Register (HW_LCDIF_CSC_COEFF1)

HW_LCDIF_CSC_COEFF1 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF1_C1_WR(0x81);//0.504x256=129
HW_LCDIF_CSC_COEFF1_C2_WR(0x19);//0.098x256=25
```

Address: HW_LCDIF_CSC_COEFF1 – 8003_0000h base + 120h offset = 8003_0120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						C2										RSRVD0						C1									
W	▨	▨	▨	▨	▨	▨											▨	▨	▨	▨	▨	▨										
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CSC_COEFF1 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C2	Two's complement blue multiplier coefficient for Y
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C1	Two's complement green multiplier coefficient for Y

33.4.20 RGB to YCbCr 4:2:2 CSC Coefficent2 Register (HW_LCDIF_CSC_COEFF2)

HW_LCDIF_CSC_COEFF2 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF2_C3_WR(0x3DB);//-0.148x256=-37
HW_LCDIF_CSC_COEFF2_C4_WR(0x3B6);//-0.291x256=-74
```

Address: HW_LCDIF_CSC_COEFF2 – 8003_0000h base + 130h offset = 8003_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						C4										RSRVD0						C3									
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CSC_COEFF2 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C4	Two's complement green multiplier coefficient for Cb
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C3	Two's complement red multiplier coefficient for Cb

33.4.21 RGB to YCbCr 4:2:2 CSC Coefficient3 Register (HW_LCDIF_CSC_COEFF3)

HW_LCDIF_CSC_COEFF3 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF3_C5_WR(0x70);//0.439x256=112
HW_LCDIF_CSC_COEFF3_C6_WR(0x70);//0.439x256=112
```

Address: HW_LCDIF_CSC_COEFF3 – 8003_0000h base + 140h offset = 8003_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						C6										RSRVD0						C5									
W	▨																▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CSC_COEFF3 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C6	Two's complement red multiplier coefficient for Cr
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C5	Two's complement blue multiplier coefficient for Cb

33.4.22 RGB to YCbCr 4:2:2 CSC Coefficient4 Register (HW_LCDIF_CSC_COEFF4)

HW_LCDIF_CSC_COEFF4 register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

EXAMPLE

```
HW_LCDIF_CSC_COEFF4_C7_WR(0x3A2);//-0.368x256=-94
HW_LCDIF_CSC_COEFF4_C8_WR(0x3EE);//-0.071x256=-18
```

Address: HW_LCDIF_CSC_COEFF4 – 8003_0000h base + 150h offset = 8003_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						C8										RSRVD0						C7									
W	▨	▨	▨	▨	▨	▨											▨	▨	▨	▨	▨	▨										
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CSC_COEFF4 field descriptions

Field	Description
31–26 RSRVD1	Reserved bits, write as 0.
25–16 C8	Two's complement blue multiplier coefficient for Cr
15–10 RSRVD0	Reserved bits, write as 0.
9–0 C7	Two's complement green multiplier coefficient for Cr

33.4.23 RGB to YCbCr 4:2:2 CSC Offset Register (HW_LCDIF_CSC_OFFSET)

HW_LCDIF_CSC_OFFSET register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by: Y = C0*R + C1*G + C2*B + Y_offset Cb= C3*R + C4*G + C5*B + CbCr_offset Cr= C6*R + C7*G + C8*B + CbCr_offset

This register carries programming information about RGB to YCbCr 4:2:2 CSC.

Address: HW_LCDIF_CSC_OFFSET – 8003_0000h base + 160h offset = 8003_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1							CBCR_OFFSET									RSRVD0							Y_OFFSET								
W	▨	▨	▨	▨	▨	▨	▨										▨	▨	▨	▨	▨	▨	▨									
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

HW_LCDIF_CSC_OFFSET field descriptions

Field	Description
31–25 RSRVD1	Reserved bits, write as 0.
24–16 CBCR_OFFSET	Two's complement offset for the Cb and Cr components
15–9 RSRVD0	Reserved bits, write as 0.
8–0 Y_OFFSET	Two's complement offset for the Y component

33.4.24 RGB to YCbCr 4:2:2 CSC Limit Register (HW_LCDIF_CSC_LIMIT)

HW_LCDIF_CSC_LIMIT register provides overall control over color space conversion from RGB to 4:2:2 YCbCr. The equations for the conversion are given by:

Y = C0*R + C1*G + C2*B + Y_offset

Cb= C3*R + C4*G + C5*B + CbCr_offset

Cr= C6*R + C7*G + C8*B + CbCr_offset

The coefficients are as follows:

LCDIF_CSC_COEFF0.C0 = 0x41

LCDIF_CSC_COEFF1.C1 = 0x81

LCDIF_CSC_COEFF1.C2 = 0x19

LCDIF_CSC_COEFF2.C3 = 0x3DB

LCDIF_CSC_COEFF2.C4 = 0x3B6

LCDIF_CSC_COEFF3.C5 = 0x70

LCDIF_CSC_COEFF3.C6 = 0x70

LCDIF_CSC_COEFF4.C7 = 0x3A2

LCDIF_CSC_COEFF4.C8 = 0x3EE

LCDIF_CSC_OFFSET.CBCR_OFFSET = 128

LCDIF_CSC_OFFSET.Y_OFFSET = 16

This register carries programming information about RGB to YCbCr 4:2:2 CSC. Note that the values in this register are unsigned.

EXAMPLE

```
HW_LCDIF_CSC_LIMIT_CBCR_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_CBCR_MAX_WR(0xF0);//240
HW_LCDIF_CSC_LIMIT_Y_MIN_WR(0x10);//16
HW_LCDIF_CSC_LIMIT_Y_MAX_WR(0xEB);//235
```

Address: HW_LCDIF_CSC_LIMIT – 8003_0000h base + 170h offset = 8003_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CBCR_MIN								CBCR_MAX								Y_MIN								Y_MAX							
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

HW_LCDIF_CSC_LIMIT field descriptions

Field	Description
31–24 CBCR_MIN	Lower limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
23–16 CBCR_MAX	Upper limit of Cb and Cr after RGB to 4:2:2 YCbCr conversion
15–8 Y_MIN	Lower limit of Y after RGB to 4:2:2 YCbCr conversion
7–0 Y_MAX	Upper limit of Y after RGB to 4:2:2 YCbCr conversion

33.4.25 LCD Interface Data Register (HW_LCDIF_DATA)

The data sent to an external LCD controller is written to this register. Data can be written to this register (from the processor\'s perspective) as bytes half-words (16 bits) or words (32 bits) as appropriate.

This register holds the 32-bit word written by either the CPU or the DMA into LCDIF. This data then gets sent out by the block across the interface. When the block is in bus master mode, this register gets the value of the lower 32 bits of the 64-bit data bus whenever it is received.

Address: HW_LCDIF_DATA – 8003_0000h base + 180h offset = 8003_0180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA_THREE								DATA_TWO								DATA_ONE								DATA_ZERO							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DATA field descriptions

Field	Description
31–24 DATA_THREE	Byte 3 (most significant byte) of data written to LCDIF by the DMA or the CPU.
23–16 DATA_TWO	Byte 2 of data written to LCDIF by the DMA or the CPU.
15–8 DATA_ONE	Byte 1 of data written to LCDIF by the DMA or the CPU.
7–0 DATA_ZERO	Byte 0 (least significant byte) of data written to LCDIF by the DMA or the CPU.

33.4.26 Bus Master Error Status Register (HW_LCDIF_BM_ERROR_STAT)

This register reflects the virtual address at which the AXI master received an error response from the slave.

When the BM_ERROR_IRQ is asserted, the address of the bus error is updated in the register.

Address: HW_LCDIF_BM_ERROR_STAT – 8003_0000h base + 190h offset = 8003_0190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_BM_ERROR_STAT field descriptions

Field	Description
31–0 ADDR	Virtual address at which bus master error occurred.

33.4.27 CRC Status Register (HW_LCDIF_CRC_STAT)

This register reflects the CRC value of each frame sent out by LCDIF. The CRC is done on the final output bus, so the value will be dependent on the LCD_DATABUS_WIDTH bitfield even if the input data is the same.

This register will be updated when the CUR_FRAME_DONE_IRQ is asserted. In the case of DVI mode, the CRC is calculated for the entire frame, not separately for each field in the frame.

The CRC equation is as follows:

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Address: HW_LCDIF_CRC_STAT – 8003_0000h base + 1A0h offset = 8003_01A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CRC_VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_CRC_STAT field descriptions

Field	Description
31–0 CRC_VALUE	Calculated CRC value.

33.4.28 LCD Interface Status Register (HW_LCDIF_STAT)

The LCD interface status register can be used to check the current status of the LCDIF block.

The LCD interface status register that contains read only views of some parameters or current state of the block.

Address: HW_LCDIF_STAT – 8003_0000h base + 1B0h offset = 8003_01B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	DMA_REQ	LFIFO_FULL	LFIFO_EMPTY	TXFIFO_FULL	TXFIFO_EMPTY	BUSY	DVI_CURRENT_FIELD	RSRVD0[23:16]							
W																
Reset	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD0[15:9]							LFIFO_COUNT								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_STAT field descriptions

Field	Description
31 PRESENT	0: LCDIF not present on this product 1: LCDIF is present.
30 DMA_REQ	Reflects the current state of the DMA Request line for the LCDIF. The DMA Request line toggles for each new request.
29 LFIFO_FULL	Read only view of the signal that indicates that LCD read datapath FIFO is full, will be generally used in the write mode of the LCD interface.
28 LFIFO_EMPTY	Read only view of the signal that indicates that LCD read dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
27 TXFIFO_FULL	Read only view of the signal that indicates that LCD write datapath FIFO is full, will be generally used in the write mode of the LCD interface.
26 TXFIFO_EMPTY	Read only view of the signal that indicates that LCD write dapatath FIFO is empty, will be generally used in the read mode of the LCD interface.
25 BUSY	Read only view of the input busy signal from the external LCD controller.

HW_LCDIF_STAT field descriptions (continued)

Field	Description
24 DVI_CURRENT_ FIELD	Read only view of the current field being transmitted. DVI_CURRENT_FIELD = 0 means field 1. DVI_CURRENT_FIELD = 1 means field 2.
23–9 RSRVD0	Reserved bits. Write as 0.
8–0 LFIFO_COUNT	Read only view of the current count in Latency buffer (LFIFO).

33.4.29 LCD Interface Version Register (HW_LCDIF_VERSION)

The LCD interface version register can be used to read the version of the LCDIF IP being used in this SoC.

The LCD interface debug register is for diagnostic use only.

Address: HW_LCDIF_VERSION – 8003_0000h base + 1C0h offset = 8003_01C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of RTL version.

33.4.30 LCD Interface Debug0 Register (HW_LCDIF_DEBUG0)

The LCD interface debug0 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: HW_LCDIF_DEBUG0 – 8003_0000h base + 1D0h offset = 8003_01D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	STREAMING_END_DETECTED	WAIT_FOR_VSYNC_EDGE_OUT	SYNC_SIGNALS_ON_REG	DMACMDKICK	ENABLE	HSYNC	VSYNC	CUR_FRAME_TX	EMPTY_WORD	CUR_STATE						
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨						
Reset	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved	Reserved	Reserved	Reserved	CUR_REQ_STATE		MST_AVALID	MST_OUTSTANDING_REQS					MST_WORDS			
W	▨	▨	▨	▨	▨		▨	▨					▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DEBUG0 field descriptions

Field	Description
31 STREAMING_END_DETECTED	Read only view of the DOTCLK_MODE or DVI_MODE bit going from 1 to 0.
30 WAIT_FOR_VSYNC_EDGE_OUT	Read only view of WAIT_FOR_VSYNC_EDGE bit in the VSYNC mode after it comes out of the TXFIFO.
29 SYNC_SIGNALS_ON_REG	Read only view of internal sync_signals_on_reg signal.
28 DMACMDKICK	Read only view of the DMA command kick signal.
27 ENABLE	Read only view of ENABLE signal.
26 HSYNC	Read only view of HSYNC signal.

HW_LCDIF_DEBUG0 field descriptions (continued)

Field	Description
25 VSYNC	Read only view of VSYNC signal.
24 CUR_FRAME_TX	This bit is 1 for the time the current frame is being transmitted in the VSYNC mode. Useful for VSYNC mode debug.
23 EMPTY_WORD	Indicates that the current word is empty.
22–16 CUR_STATE	Read only view of the current state machine state in the current mode of operation.
15 Reserved	This bit is reserved. Reserved, always set to zero.
14 Reserved	This bit is reserved. Reserved, always set to zero.
13 Reserved	This bit is reserved. Reserved, always set to zero.
12 Reserved	This bit is reserved. Reserved, always set to zero.
11–10 CUR_REQ_STATE	Read only view of the request state machine.
9 MST_AVALID	Read only view of the mst_avalid signal issued by the AXI bus master.
8–4 MST_ OUTSTANDING_ REQS	Read only view of the current outstanding requests issued by the AXI bus master.
3–0 MST_WORDS	Read only view of the current bursts issued by the AXI bus master.

33.4.31 LCD Interface Debug1 Register (HW_LCDIF_DEBUG1)

The LCD interface debug1 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: HW_LCDIF_DEBUG1 – 8003_0000h base + 1E0h offset = 8003_01E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	H_DATA_COUNT																V_DATA_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DEBUG1 field descriptions

Field	Description
31–16 H_DATA_COUNT	Read only view of the current state of the horizontal data counter.
15–0 V_DATA_COUNT	Read only view of the current state of the vertical data counter.

33.4.32 LCD Interface Debug2 Register (HW_LCDIF_DEBUG2)

The LCD interface debug2 register provides a diagnostic view of the state machine and other useful internal signals.

The LCD interface debug register is for diagnostic use only.

Address: HW_LCDIF_DEBUG2 – 8003_0000h base + 1F0h offset = 8003_01F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MST_ADDRESS																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LCDIF_DEBUG2 field descriptions

Field	Description
31–0 MST_ADDRESS	Read only view of the current address issued by the AXI bus master.

Chapter 34
Pixel Pipeline (PXP)

34.1 Pixel Pipeling (PXP) Overview

The pixel pipeline is used to perform alpha blending of graphic or video buffers with graphics data before sending to an LCD display or TV encoder. The PXP also supports image rotation for hand-held devices that require both portrait and landscape image support.

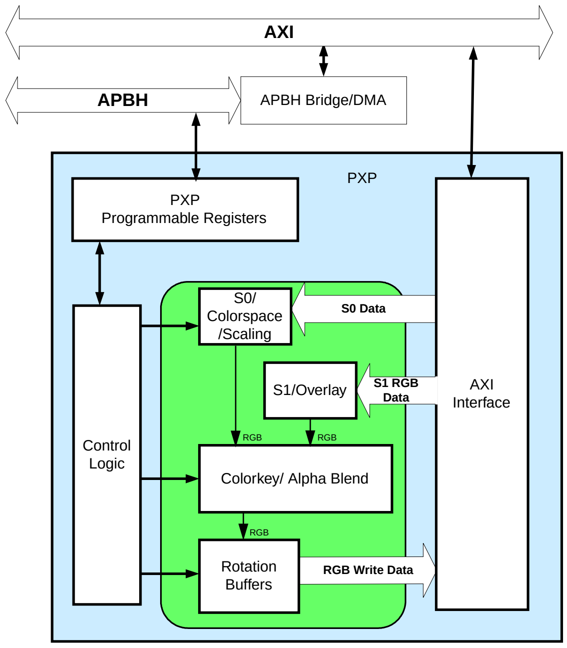


Figure 34-1. Pixel Pipeline (PXP) Block Diagram

The PXP is organized as having a background image (S0) and one or more overlay images that can be blended with the background. Each overlay image must be a multiple of the block size in pixels in both height and width and the offset of the overlay into the background image must also be a multiple of the block size in pixels. As the PXP processes data, it reads each NxN block from the background image and finds the highest priority (lowest numbered) overlay that is co-located at that block coordinate. The PXP then fetches the overlay and performs the alpha blending and color key operations on the two blocks. The resulting pixel block is then written to the corresponding block in the output buffer.

For the S0 plane, the PXP supports RGB images (unscaled) or color space conversion (YUV->RGB) and scaling of YUV images. The S1 plane consists of up to eight overlay regions consisting of 16- or 32-bit RGB data. The S0 and S1 planes may then be combined by alpha blending, color key substitution, or raster operations (ROPs) to form the output image. Finally, the resulting image may be clockwise rotated in 90 degree increments and/or flipped horizontally and/or vertically. The PXP also supports letterboxing and interlacing of progressive content (by writing alternate lines to different frame buffers).

The flow of data through the PXP is shown below.

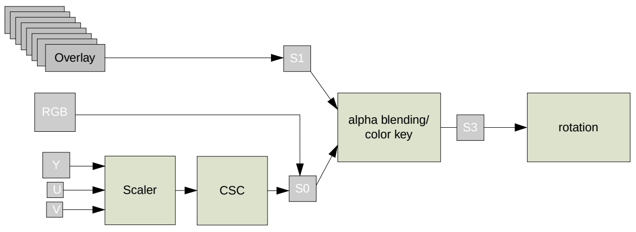


Figure 34-2. Pixel Pipeline (PXP) Data Flow

34.1.1 Image Support

The PXP's S0 buffer supports the following image formats:

- 24-bit unpacked RGB (32bpp)
- 24-bit packed RGB (24bpp)
- 16-bit RGB in either 555 or 565 format
- 3-plane YUV/YCbCr in 4:2:0 or 4:2:2 format
- 2-plane YUV/YCbCr in 4:2:0 or 4:2:2 format
- 2-plane YUV/YCbCr in 4:2:2 format

The PXP's S1 buffer supports the following image formats:

- 32-bit RGB (with or without alpha)
- 16-bit RGB in either 555, 565, or 1555 (alpha)

The PXP's output buffer supports:

- 32-bit RGB (with alpha)
- 24-bit packed RGB (24bpp)

- 16-bit RGB in either 565, 555, or 1555 format
- YUV 4:4:4/4:2:2 1-plane
- YUV 4:2:2/4:2:0 2-plane
- Interlaced output processing

Internally, all image data is handled as 32bpp data (either RGB or YUV, depending on output mode selection) for all steps after the color space conversion. Input RGB images are always converted to the equivalent 32bpp format before processing.

34.1.2 Block Size Selection

The PXP can be configured to process blocks that are either 8x8 pixels or 16x16 pixels. The granularity of image size and location of video or overlay buffers within the final destination frame buffer has twice the precision when selecting 8x8 pixel block sizes. When selecting a 16x16 pixel block size, the accesses to fetch S0 and S1 images and write the final frame buffer are more efficient since twice as much data is requested and processed per memory request. When optimizing the system for memory bandwidth and image processing time, configure the PXP to process 16x16 pixel blocks.

The control registers that are in block size units need to be programmed consistently with the BLOCK_SIZE control bit setting. If the source image is 32x32 pixels, then the width and height setting will be 4 when selecting 8x8 pixel block size and 2 when selecting 16x16 block size.

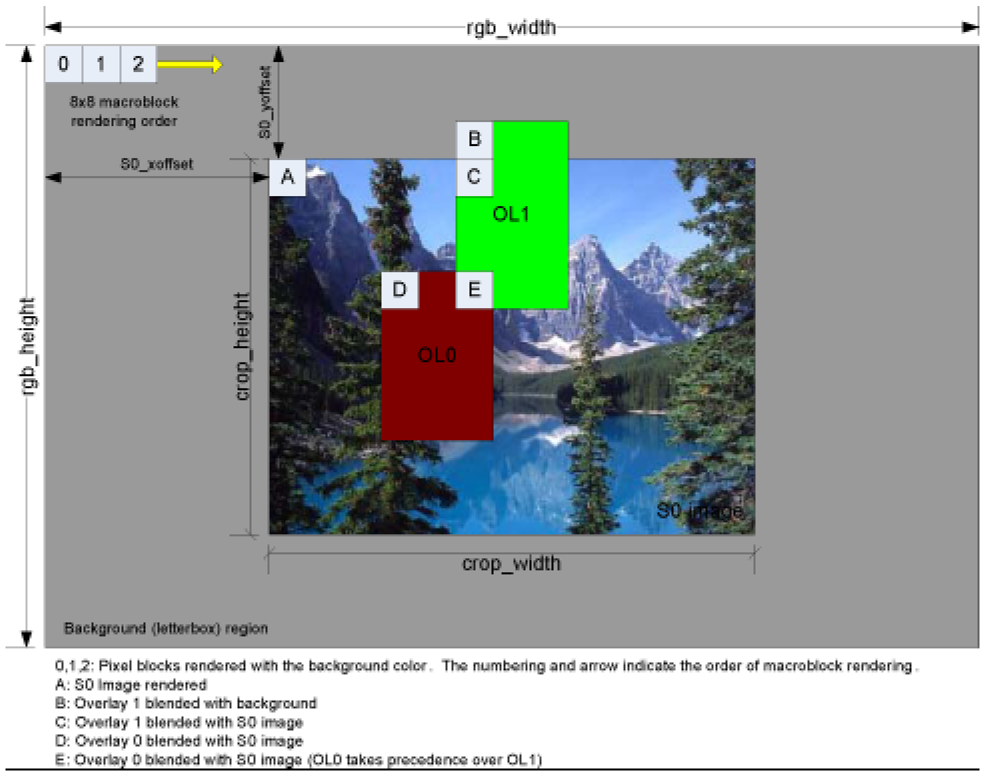
34.1.3 PXP Limitations/Issues

- The PXP's scalar uses a bilinear scaling algorithm and can scale YUV images from 1/4x to 4096x in 12-bit fractional steps.
- When using the NEXT register, the interrupt enable setting should remain the same for all frames. If not, the PXP will change the interrupt enable register value and possibly cause the loss of an interrupt.
- The PXP cannot rotate/flip video in the interlaced modes.
- When performing input interlacing, the input image and overlays must be multiples of 8x16 (in 8x8 block size mode) or 16x32 (16x16 pixel blocks) pixels. Overlays must also reside on the same boundaries.

34.2 Operation

The PXP operates by rendering the output frame buffer in 8x8 or 16x16 pixel macroblocks in display order (left to right, then top to bottom). At each output macroblock location, the PXP determines whether the S0 buffer is visible based on the cropping register and S0 offset parameters. If the S0 plane is visible, the PXP will fetch and process the required data from the S0 image, otherwise, the S0's contribution to the output macroblock will be the S0BACKGROUND register value. This value is effectively the color of the letterboxed region or background color.

The PXP will also determine if an overlay is present for that macroblock location, and if so, instruct the S1 buffer to fetch the required data. If multiple overlays cover the macroblock, the PXP will select only the lowest numbered overlay and direct the S1 buffer to load the data for this overlay. For areas with no overlays, the S1 buffer contributes nothing to the rendered image. The following figure shows the order in which the output blocks are generated (blocks 0, 1, 2) and indicates how various blocks are rendered (blocks A-E).



0,1,2: Pixel blocks rendered with the background color. The numbering and arrow indicate the order of macroblock rendering.
A: S0 Image rendered
B: Overlay 1 blended with background
C: Overlay 1 blended with S0 image
D: Overlay 0 blended with S0 image
E: Overlay 0 blended with S0 image (OL0 takes precedence over OL1)

Figure 34-3. Pixel Pipeline (PXP) Macro Blocks

It is important to understand how the PXP renders each output macroblock to properly understand how it accomplishes cropping, letterboxing, and overlay blending. The following sections will provide more details on these operations.

The PXP also has the ability to rotate/flip images for cases when the pixel scan order is not in the traditional left-to-right/top-to-bottom raster scan (landscape raster). This can occur when a handheld device with a traditional landscape scan is rotated into a portrait orientation (in which the scan order is now bottom-to-top/left-to-right or vice versa) or when a cell phone oriented display (portrait raster) is rotated into a landscape orientation for viewing videos. In these cases, the PXP still renders the image in scan-order format (as sent to the device), but it will traverse the input images based on the transformations required.

The following sections detail each of the PXP's functional capabilities.

34.2.1 Pixel Handling

All pixels are internally represented as 24-bit RGB or YUV/YCbCr values with an 8-bit alpha value at all stages in the PXP after the color space converter (CSC). The output format selected determines the colorspace representation of pixels after the CSC stage of the pixel pipeline. It should be noted that pixels in the YUV/YCbCr color space cannot be blended with S1 overlay pixels since this channel only processes RGB pixel formats.

Input pixels are converted into the RGB format using the following rules:

- 32-bit ARGB8888 pixels are read directly with no conversion for both the S0 and overlay images.
- 32-bit RGB888 pixels are assumed to have an alpha value of 0xFF (full opaque).
- 16-bit RGB565 and RGB555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of 0xFF (opaque). The expansion process replicates the upper pixel bits into the lower pixel bits (for instance a 16-bit RGB565 triplet of 0x1F/0x20/0x07 would be expanded to 0xFF/0x82/0x39).
- 16-bit RGB1555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of either 0x00 or 0xFF, based on the 1-bit alpha value in the pixel. The ALPHA_MULTIPLY function is useful in this scenario to allow scaling of the opaque pixels to a semi-transparent value.

Input pixels are processed in the YUV/YCbCr format using the following rules:

- All pixels are processed through the scaling engine for conversion to the YUV/YCbCr 4:4:4 24-bit format.

- When not resizing the input image, the input pixels are still processed through the scaling engine to convert 4:2:2 or 4:2:0 formats to 4:4:4 formats. Essentially, the chroma values are resized to contain a unique chroma sample for each pixel site.
- S1, or overlays cannot be supported in this case since S1 RGB pixels cannot be converted to YUV/YCbCr.

Output pixels will retain the effective alpha value of the overlay or can be set to a programmed alpha value using the ALPHA field of the S0PARAM register. 16-bit pixels values are formed from the most significant bits of the 24-bit pixel values.

34.2.2 S0 Cropping/Masking

The PXP's cropping operation should be viewed as a mask on the output image through which the background S0 plane can be viewed. Using this definition clarifies a subtlety on the usage of cropping an image when the image is scaled. When scaling is not used, the input and output image sizes are the same, therefore, the operation is analogous to cropping the input source image.

The background output image can be cropped to a width and height independent of the image size at a given offset into the image (all sizes are in terms of block size pixel units) using the values in the S0CROP register. The XBASE and YBASE provide the coordinates of the first block to be displayed from the source image and the WIDTH and HEIGHT parameters specify an effective size of the resulting image in the output buffer.

Cropping must be enabled by setting the CROP bit in the CTRL register to a 1. When not set, the visible portions of the S0 image will be rendered based on the WIDTH and HEIGHT specified in the S0SIZE field. The following figure indicates how the various cropping parameters relate to the source and RGB images (non-scaled case).

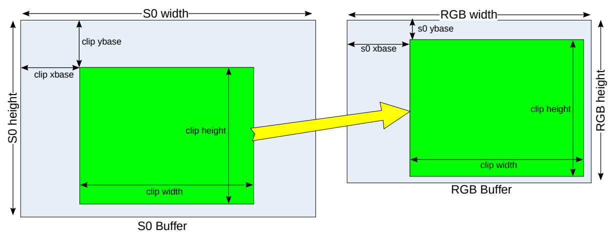


Figure 34-4. Pixel Pipeline (PXP) Cropping

It is important to note that when scaling an image, software **must** specify a valid cropping region since the PXP will default to using the source image size. When downscaling, this is not an issue, but with upscaling the resulting image will be a scaled up version of the source, but cropped to the same size as the source image as shown below.

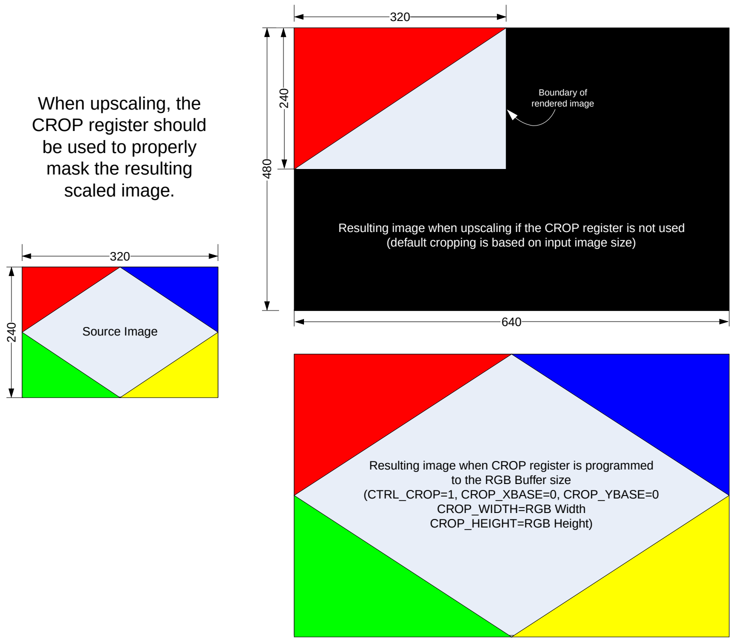

When upscaling, the CROP register should be used to properly mask the resulting scaled image.
320
240
480
640
Boundary of rendered image
Resulting image when upscaling if the CROP register is not used (default cropping is based on input image size)
Source Image
Resulting image when CROP register is programmed to the RGB Buffer size (CTRL_CROP=1, CROP_XBASE=0, CROP_YBASE=0 CROP_WIDTH=RGB Width CROP_HEIGHT=RGB Height)

Figure 34-5. Pixel Pipeline (PXP) Scaling and Cropping Example

The cropping extents should fall completely within the S0 buffer to avoid displaying incorrect data. The PXP hardware does not check for these conditions and will render the image as shown in the following two diagrams. (Note that the cropping width and height can be viewed as applying to the input buffer only because it is not scaled. In actuality, it is applied to the output buffer).

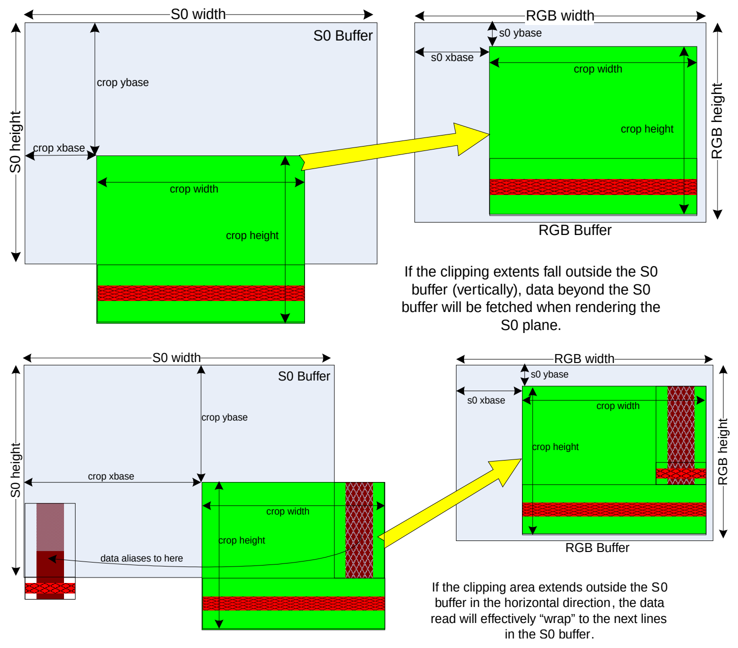

S0 width
S0 Buffer
S0 height
crop ybase
crop xbase
crop width
crop height
RGB width
s0 ybase
s0 xbase
RGB height
RGB Buffer
If the clipping extents fall outside the S0 buffer (vertically), data beyond the S0 buffer will be fetched when rendering the S0 plane.
data aliases to here
If the clipping area extends outside the S0 buffer in the horizontal direction, the data read will effectively "wrap" to the next lines in the S0 buffer.

Figure 34-6. Invalid PXP Cropping Examples

34.2.3 Scaling

The PXP can scale YUV images from 1/4x to about 4096x using a bilinear scaling algorithm. The hardware is capable of scaling with 12-bit fractional resolution, or in 1/4096th pixel increments with independent scaling ratios for the X and Y direction. The scaler also implements an initial offset, which can be useful when scaling by powers of 2 in order to ensure that the resulting pixels are averages (select ½ pixel offset) of the source pixels instead of producing a decimated or replicated image. Also, the offset can be used to achieve per pixel addressing on the input source image.

The scaling parameter is specified to the hardware in terms of the inverse of the scaling ratio desired. This can also be viewed as the step size between computed sample values. For instance, when scaling by 2x the inverse is 1/2, therefore the scaler will increment by 1/2 pixel steps across the input image and compute the bilinear average for each sample point.

The scaling values are represented by 12-bit fractional values in the scaling register and hardware. The scaling ratios are computed as the input size divided by the output size. The resulting decimal value must then be converted into a 12-bit fixed point value by multiplying by 2^{12} or 4096 to produce the value programmed into the scaling registers.

To scale an image from 400x300 to 320x200, the horizontal XSCALE factor is computed as

$$\mathrm{XSCALE} = \frac{\mathrm{InputSize}}{\mathrm{OutputSize}} \times 4096 = \frac{400}{320} \times 4096 = 5120 \times 4096 = 0x0140\_0000$$

The vertical YSCALE can be similarly computed as

$$\mathrm{YSCALE} = \frac{\mathrm{InputSize}}{\mathrm{OutputSize}} \times 4096 = \frac{300}{200} \times 4096 = 6144 \times 4096 = 0x0180\_0000$$

The scaler will use the CROP_XBASE and CROP_YBASE values as an offset into the source S0 image for the origin of the input image to be scaled. The CROP_WIDTH and CROP_HEIGHT parameters will be used to determine the extent of the **scaled** image in the output buffer. It is tempting to view the cropping width and height as being applied to the input buffer, but this is incorrect -- the PXP uses these values as a mask on the output buffer to determine which regions of the output buffer require data from the scaled input image.

To enable scaling, the HW_PXP_CTRL_SCALE bit must be set and the desired scaling ratios written into the HW_PXP_S0SCALE registers. Initial offsets should be programmed into the HW_PXP_S0OFFSET register.

34.2.4 Color Space Conversion (CSC)

The CSC module receives scaled YUV/YCbCr444 pixels from the scale engine and converts the pixels to the RGB888 color space. These pixels are loaded into the pixel FIFO for processing by the alpha blend module. The CSC module can also be bypassed to allow pixel output formats in the YUV/YCbCr color space.

The following equations are used to perform YUV/YCbCr -> RGB conversion. The constants will be stored in the PXP control registers as two's compliment values to allow flexibility in the implementation and to allow for differences in the video encode and decode operations. In addition, this provides a software mechanism to manipulate brightness or contrast.

$$R = C0(Y+Yoffset) + C1(V+UVoffset)$$
$$G = C0(Y+Yoffset) + C3(U+UVoffset) + C2(V+UVoffset)$$
$$B = C0(Y+Yoffset) + C4(U+UVoffset)$$

Note

In the equations above, U and V are synonymous with Cb and Cr in regards to the color space format of the source frame buffer.

Saturation of each color channel is checked and corrected for excursions outside the nominal YUV/YCbCr color spaces. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

The following table indicates the expected coefficients for YUV and YCbCr modes of operation:

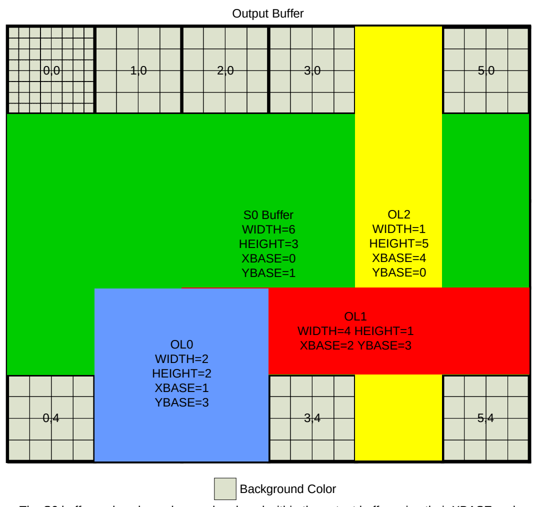
Table 34-1. Coefficients for YUV and YCbCr Operation

Coefficient	YUV	YCbCr
Yoffset	0x000	0x1F0 (-16)
UVoffset	0x000	0x180 (-128)
C0	0x100 (1.00)	0x12A (1.164)
C1	0x123 (1.140)	0x198 (1.596)
C2	0x76B (-0.581)	0x730 (-0.813)
C3	0x79B (-0.394)	0x79C (-0.392)
C4	0x208 (2.032)	0x204 (2.017)

By default, the PXP color space coefficients are set to support the conversion of YUV data to RGB data. If YCbCr input is present, software must change the coefficient registers appropriately (see the register definitions for values). Software must also set the YCBCR_MODE bit in the COEFF0 register to ensure proper conversion of YUV versus YCbCr data.

34.2.5 Overlays

The PXP supports up to eight overlays that can be used to merge graphic data with video (or other graphic data). Each overlay consists of a rectangular area that is a multiple of Ôn' (where n is the block size) pixels in both the vertical and horizontal directions. Overlays must also be located on NxN boundaries within the output image. As the PXP processes each NxN macroblock, it determines if any of the enabled overlays cover the block and then merges the overlay data with the background image as specified in the overlay's control registers. If multiple overlays overlap for a given NxN block, the PXP will select the lowest numbered one for the blending operation. If the desired affect is to blend the overlays together, this can be accomplished as a multi-step process using the IN_PLACE functionality (see In-place Rendering).



The S0 buffer and each overlay can be placed within the output buffer using their XBASE and YBASE registers and the dimensions of each region are set using their WIDTH and HEIGHT parameters. Overlay 0 has the highest priority (effectively it is the highest in the stacking order) and the S0 buffer and background color have the lowest priority.

Overlays can be blended with the background or S0 planes, but not with each other. Effectively only a single overlay is active for each 8x8 pixel block.

Figure 34-7. Pixel Pipeline Overlay Support

Each overlay can perform one of three classes of operations between the overlay and the underlying background (S0) image: alpha blending, color keying, or raster operations.

An overlay can be enabled by writing the address of the overlay image to the OLn register, the overlay's size and location information into the OLnSIZE register, and then setting the OLnPARAM_ENABLE bit. The OLnPARAM registers also contain further controls to select the modes of operation (below).

34.2.6 Alpha Blending

The alpha value for an individual pixel represents a mathematical weighting factor applied to the S1 pixel. An alpha value of 0x00 corresponds to a transparent pixel and a value of 0xFF corresponds to an opaque pixel.

The effective alpha value for an overlay pixel is determined by the ALPHA bit-field and the two ALPHA control bits in the OLnPARAM register. If the ALPHA_CTRL field is set to ALPHA_OVERRIDE, the alpha value for the pixel is taken from the ALPHA bit-field. This can be useful for applying a constant alpha to an entire image or for image formats that do not include an alpha value. If ALPHA_MULTIPLY is selected, the pixel's alpha value will be multiplied by the ALPHA value in order to allow scaling of the pixel's alpha or to provide better control for pixel formats such as RGB1555, which only contains a single bit of alpha.

For each color channel, the equation used to blend two source pixels is defined below:

$E\alpha$ = Embedded alpha associated with S1 pixel
$\alpha = G\alpha * E\alpha + 0x80$
$G\alpha$ = PIO programmed global alpha (8-bit value)

The result for the red channel as an example is as follows:

$$Yr[7:0] = (\alpha * S1.r) + ((1 - \alpha) * S0.r)$$

When alpha is 0xFF, the S1 pixel will not be blended with S0, but S1 will be passed as the output pixel and will not be blended with S0. In this case, S0 will be discarded. Likewise, if alpha is 0x00 for a given pixel, S0 will be loaded as the output pixel.

Alpha values in the overlays are loaded from the source image for all pixel formats. For formats that do not support an alpha value, the pixel is assigned an alpha value of 0xFF (opaque). This can be modified by the overlay processing by setting either the ALPHA_MULTIPLY or ALPHA_OVERRIDE bit in the associated OLnPARAM register.

34.2.7 Color Key

Pixels may be made transparent to the corresponding overlay by using the S0 color key registers. If an S0 pixel matches the range specified by the S0COLORKEYLOW and S0COLORKEYHIGH registers, the pixel from the associated overlay will be displayed. If no overlay is present for that block, a black pixel will be generated since the default overlay pixel is 0x00000000 (transparent black pixel).

The most common use for this is when a bitmap does not support an alpha-field or for applications such as "green screen" where an image is substituted for a solid background color as shown below.

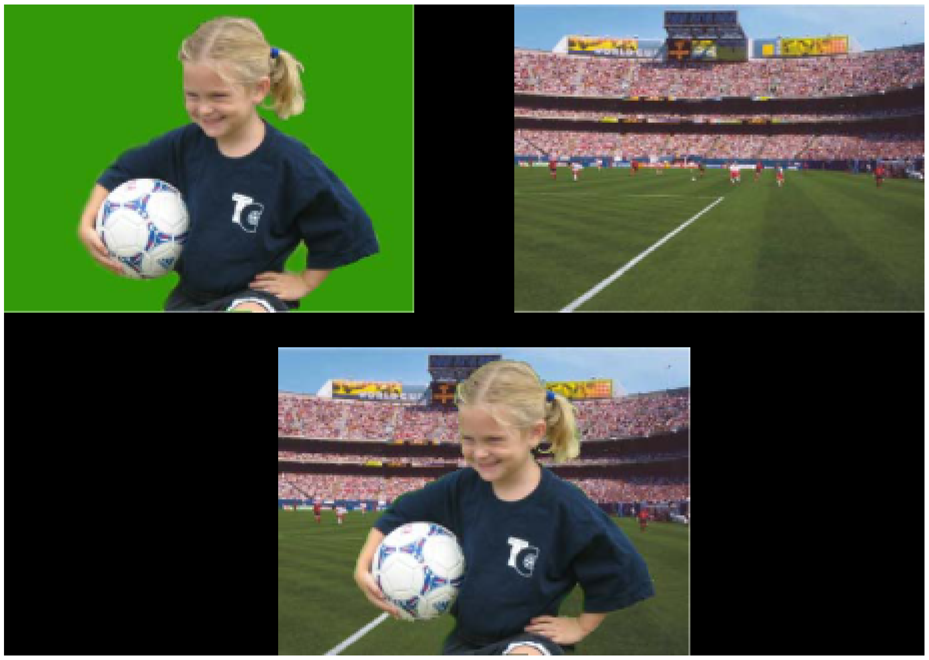


Figure 34-8. Pixel Pipeline (PXP) Color Key Example

The green portion of the overlay image can be color keyed to display the contents of the S0 buffer for locations that match the color range. For this example, the color range is

OL Colorkey: 00<R<80 70<G<ff 00<B<80

Conversely, background color keying could also have been used if the images had been swapped.

If color keying is enabled for an overlay, any pixels matching the color key parameters will be handled as color keyed pixels. Non-matching pixels will be alpha blended or handled by ROP operations as normal.

34.2.8 Raster Operations (ROPs)

In addition to alpha blending and color keying, the PXP's alpha blender also supports a set of raster operations that may be performed between the active overlay and the background image. The operations are done on a per-pixel basis and are performed using the 24-bit overlay and background image values. The following table lists the supported ROP operations

Table 34-2. Supported ROP Operations

Mnemonic	Value	Operation
MASKOL	0x0	OL & S0
MASKNOTOL	0x1	~OL & S0
MASKOLNOT	0x2	OLL & ~S0
MERGEOL	0x3	OL | S0
MERGEOLNET	0x4	~OL | S0
MERGEOLNOT	0x5	OL | ~S0
NOTCOPYOL	0x6	~OL
NOT	0x7	~S0
NOTMASKOL	0x8	~(OL & S0) (nand)
NOTMERGEOL	0x9	~(OL | S0) (nor)
XOROL	0xA	OL ^ S0 (xor)
NOTXOROL	0xB	~(OL ^ S0) (xnor)

These operations are specified in the overlay's PARAM register and must be enabled by setting the ALPHA_CTRL field to ROPs.

34.2.9 Rotation

Rotation is an inherently inefficient operation, especially for a graphics device operating in a raster-scan fashion since the resulting memory fetches would be non-contiguous. The PXP solves this problem by operating on NxN pixel blocks. This allows the PXP to rotate a subportion of the image, where it can fetch N lines of pixels, process them, and then write N lines of pixels regardless of the rotation orientation.

Rotation is mainly useful for reorganizing the frame buffer for handheld LCD displays for cases when the user rotates the device from a portrait to landscape orientation. Consider the following scenario:

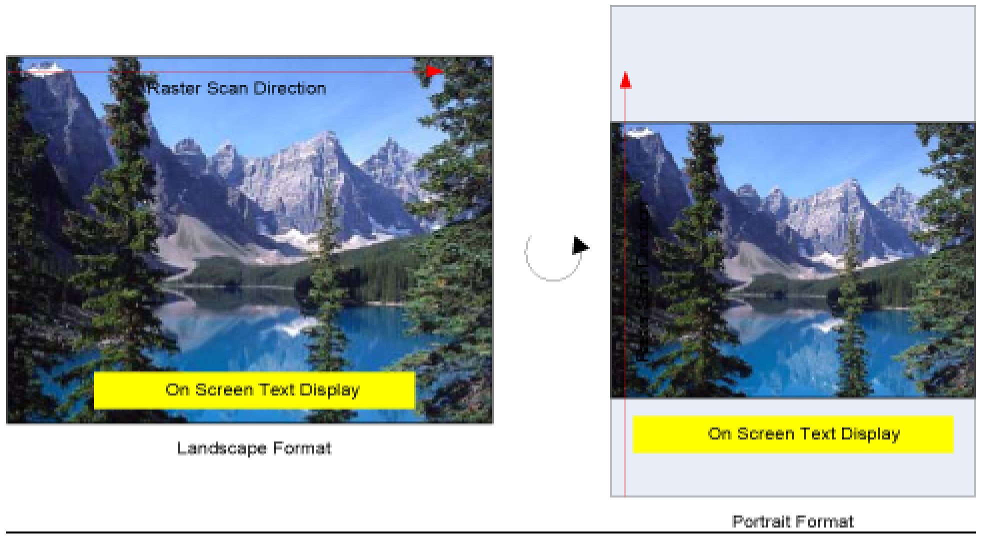


Figure 34-9. Pixel Pipeline (PXP) Rotation Example 1

While this looks like a trivial operation, consider what the frame buffer must look like in memory before being sent to the LCD in raster-scan format

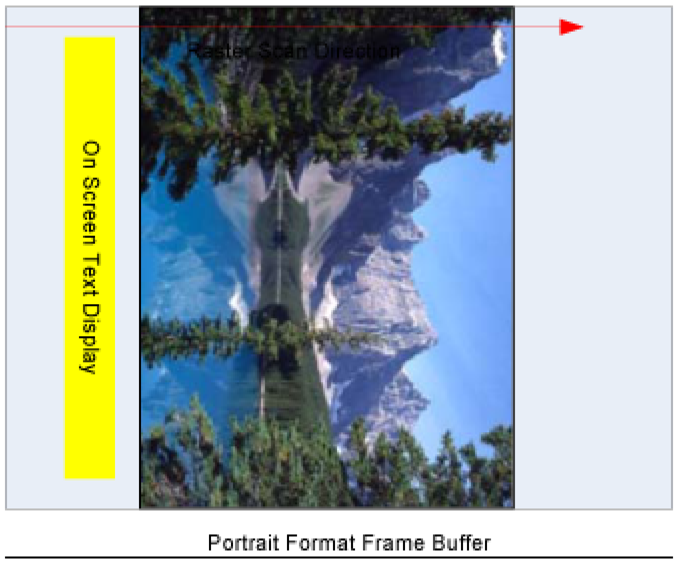


Figure 34-10. Pixel Pipeline (PXP) Rotation Example 2

Not only must the image be rotated, but any on-screen graphics must also be rendered in a different orientation. By building rotation into the rendering process, the PXP allows software to construct the image in the traditional portrait format and simply rotate the image/overlays for the LCD interface during composition.

The rotation operations are defined as rotations in a clockwise direction and the flip operations will flip the pixels in the specified direction.

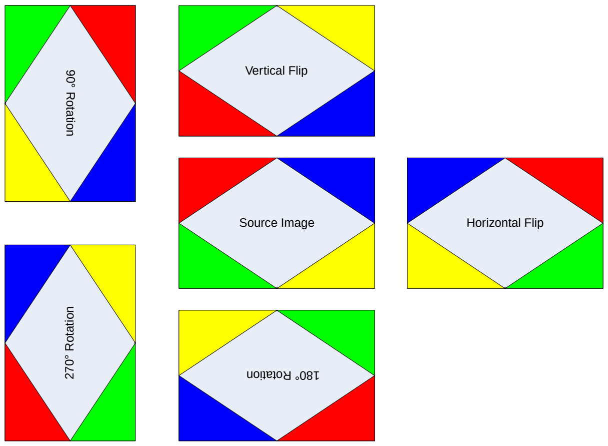


Figure 34-11. Pixel Pipeline (PXP) Rotation and Flip Definition

The PXP supports rotation in 90 degree increments as well as horizontal and vertical flip operations. These can be done in any combination (for example, 90 degree rotation with both vertical and horizontal flip). When a flip operation is specified in combination with a rotation operation, the PXP will render the output such that the effect of the flip operation(s) occur BEFORE the rotation operation.

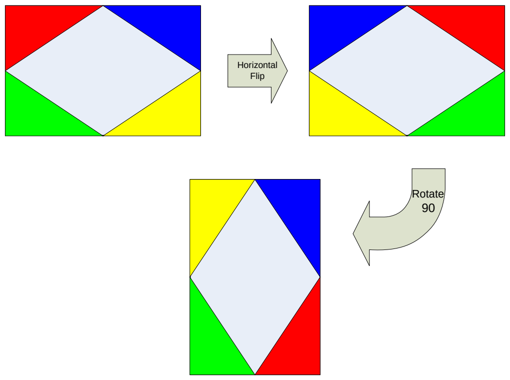


Figure 34-12. Pixel Pipeline (PXP) Rotation Plus Flip Definition

Rotations and flip operations are enabled by setting the VFLIP, HFLIP, and ROTATE fields of the HW_PXP_CTRL register.

34.2.10 In-place Rendering

The PXP also has the ability to process an image and write the resulting buffer back to the original S0 buffer. This is referred to as in place rendering. This scenario may be useful when software wishes to alpha blend multiple images where the overlays effectively overlap each other.

When the IN_PLACE control bit is set to 1, the control logic will optimize the PXP's operations to only process the blocks that match an overlay region since all other pixels will be unmodified. This considerably reduces the processing time as well as the memory bandwidth used.

In place rendering is enabled by setting the IN_PLACE bit in the HW_PXP_CTRL registers. Note the following restrictions when rendering in place:

- The source buffer is used as the destination buffer (RGBBUF is not used)
- Only RGB S0 images are supported (not YUV)
- The output RGB format must be programmed to the same value as the input RGB format

34.2.11 Interlaced Video Support

The PXP has some minimal ability to generate interlaced video content from a progressive source. There two available options, based on the bandwidth requirements and how software is managing video frames. The PXP can either interlace on the input side (by reading every other line of input data) or on the output side (by writing the individual lines of video into two separate fields). Generally, output interleaving should be used since it is the most flexible mode (it allows scaling and full overlay support) and it only requires a single pass of the PXP to generate two separate output fields. Input interleaving can be beneficial in cases where the PXP is running at 60 fps, since it requires fewer fetches to produce the output data.

The PXP will perform input interlacing when the INTERLACED_INPUT field is programmed to either FIELD0 or FIELD1 (to select the desired field). When performing output interlacing, the PXP will write field0 data to the OUTBUF pointer and the field1 data to the OUTBUF2 pointer. The OUTPUT_INTERLACING field of the HW_PXP_CTRL register controls which of these fields (or both) are generated.

Note

Output interlacing AND 2-plane output modes are not supported concurrently.

34.2.12 Queueing Frame Operations

The PXP supports a primitive ability to queue up one operation while the current operation is running. This is enabled through the use of the HW_PXP_NEXT register. When this register is written, it enables the PXP to reload its current register contents with the data found at the location pointed to by this address (when it completes processing of the current frame (note that if virtual memory is used, this will be a virtual memory address). This feature may be useful in helping to reduce the interrupt latency in servicing the PXP.

If the PXP is idle when the HW_PXP_NEXT register is written, the PXP treats this as an indication that it should immediately load the values at the pointer and begin processing the frame. This ability should allow software to use the same routines when programming the PXP (so that the first frame does not differ from subsequent frames).

When loading values from the NEXT register, nearly all registers in the PXP are reloaded, including the interrupt enable bit in the control register. It is recommended that the interrupt enable value not be changed when using queued operations to ensure that interrupts are not spuriously lost or generated. The following table indicates the registers that are affected and the offset into the block address in memory.

Table 34-3. Registers and Offsets

Offset	Register	OFFSET	REGISTER
0x00	CTRL	0x60	OL2
0x04	RGBBUF	0x64	OL2SIZE
0x08	RGBBUF2	0x68	OL2PARAM
0x0C	RGBSIZE	0x6C	OL2PARAM2
0x10	S0BUF	0x70	OL3
0x14	S0UBUF	0x74	OL3SIZE
0x18	S0VBUF	0x78	OL3PARAM
0x1C	S0PARAM	0x7C	OL3PARAM2
0x20	S0BACKGROUND	0x80	OL4
0x24	S0CROP	0x84	OL4SIZE
0x28	S0SCALE	0x88	OL4PARAM
0x2C	S0OFFSET	0x8C	OL4PARAM2
0x30	S0COLORKEYLOW	0x90	OL5
0x34	S0COLORKEYHIGH	0x94	OL5SIZE
0x38	OLCOLORKEYLOW	0x98	OL5PARAM
0x3C	OLCOLORKEYHIGH	0x9C	OL5PARAM2
0x40	OL0	0xA0	OL6

Offset	Register	OFFSET	REGISTER
0x44	OL0SIZE	0xA4	OL6SIZE
0x48	OL0PARAM	0xA8	OL6PARAM
0x4C	OL0PARAM2	0xAC	OL6PARAM2
0x50	OL1	0xB0	OL7
0x54	OL1SIZE	0xB4	OL7SIZE
0x58	OL1PARAM	0xB8	OL7PARAM
0x5C	OL1PARAM2	0xBC	OL7PARAM2

34.3 Examples

This section includes several examples of programming the PXP to render an output image. The image could be either a still image or one frame of a sequence of video images. For each case, the input and output images will be shown along with a table of PXP register settings. In all examples, pointers to the data structures with image data will be referred to in the following notation: **imagename_type*, where imagename indicates which image is being used and type indicates either luma (y), chroma (u, v) or RGB data (rgb). All register names are assumed to have the **HW_PXP_** register prefix. The registers can be written in any order except the HW_PXP_CTRL register, which must be written last since it enables the PXP's operation.

34.3.1 Basic QVGA Example

This example shows how to perform basic color space conversion of a 3-plane YUV image into an RGB image suitable for an LCD device.

Table 34-4. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.

RGBSIZE	0xFF1400F0	ALPHA=0xFF
		WIDTH=0x140=320
		HEIGHT=0x0F0=240

S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer

Register	Value	Description
SOVBUF	*morraine_v	Pointer to input V buffer

S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels)
		HEIGHT=0x1E=30 (30*8=240 pixels)

S0BACKGROUND	0x00000000	Black background region
S0CROP	0x00000000	No Cropping
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

CTRL	0x00009003	S0_FORMAT=9
		(YUV420) IRQ_ENABLE=1
		ENABLE=1

The resulting image is simply the RGB equivalent of the YUV image:

Figure 34-13. Example: RGB Equivalent of YUV image

34.3.2 Basic QVGA with Overlays

This example is similar to the last, but adds two overlay images, one for a logo and the other as a time counter/control bar. The two overlay images are shown below. (Note that the black background is actually transparent in the real image).

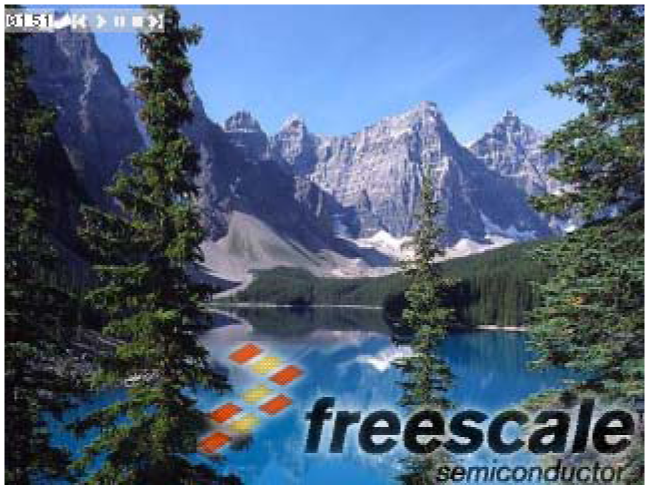


Figure 34-14. Example: QVGA with Overlays

Table 34-5. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.

RGBSIZE	0xFF1400F0	ALPHA=0xFF
		WIDTH=0x140=320
		HEIGHT=0x0F0=240

Register	Value	Description
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer

S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels)
		HEIGHT=0x1E=30 (30*8= 240 pixels)

S0BACKGROUND	0x00000000	Black background region
S0CROP	0x00000000	No Cropping
S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*overlay1_rgb	Pointer to control graphic

OL0SIZE	0x00000A02	WIDTH=0x0A=80 pixels
		HEIGHT=0x02=16pixels

OL0PARAM	0x0000FF01	ALPHA=0xFF FORMAT=0x0 (RGB8888)
		ALPHA_CTRL=0 (embedded alpha)
		ENABLE=1

OL1	*logo_rgb	Pointer to logo graphic

OL1SIZE	0x0A181D06	XBASE=0x0A=80pixels
		YBASE=0x18=192pixels
		WIDTH=0x1D=232pixels
		HEIGHT=0x06=48pixels

OL1PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

Register	Value	Description

CTRL	0x00009003	S0_FORMAT=9
		(YUV420) IRQ_ENABLE=1
		ENABLE=1

The resulting image is shown below. Note the presence of the overlays in the upper left and lower right corners of the image.

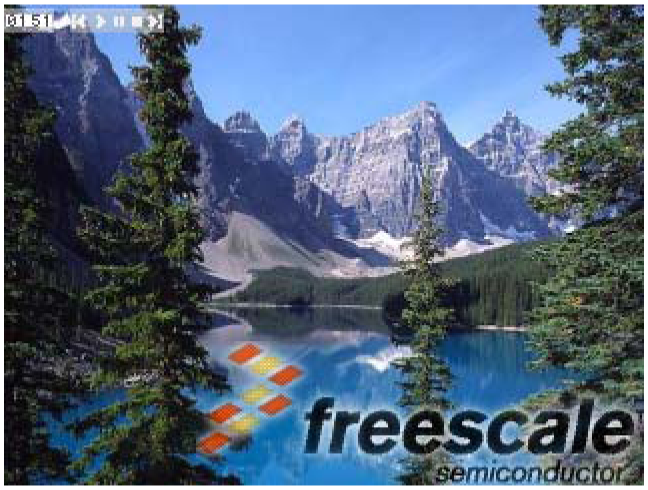


Figure 34-15. Example: QVGA with Overlays

34.3.3 Cropped QVGA Example

This example displays the same image as the first example, but does so on a portrait-oriented display (240x320) without the overlays. Changes from the first example are shown in bold.

Table 34-6. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.

RGBSIZE	0xFF0F0140	ALPHA=0xFF
		WIDTH=0x0F0=240 pixels
		HEIGHT=0x140=320 pixels

S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer

Register	Value	Description

S0PARAM	0x00**05**281E	**YBASE=0x05=40pixels**
		WIDTH=0x28=40 (40*8=320 pixels)
		HEIGHT=0x1E=30 (30*8= 240 pixels)

S0BACKGROUND	0x00000000	Black background region

S0CROP	0x05001E1E	XBASE=0x05=40 pixels
		YBASE=00=0pixels
		WIDTH=0x1E=240pixels
		HEIGHT=0x1E=240 pixels

S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

CTRL	0x00089003	**CROP=1**
		S0_FORMAT=9
		(YUV420) IRQ_ENABLE=1
		ENABLE=1

In this case, we have now changed the RGB size to reflect the portrait nature of the display. The S0PARAM_YBASE has been changed to 0x05 (40 pixels) to place the S0 plane down 40 pixels from the top of the screen. The cropping register is now also used to control the cropping extents. The CROP_XBASE is set to 0x05 (40 pixels) to move the origin of the S0 buffer to the (40,0) location within the buffer. The CROP_WIDTH/CROP_HEIGHT are also programmed to ensure that the resulting image in the output buffer is cropped to 240x240 pixels. Since the image no longer covers the entire output buffer, the S0BACKGROUND register is used to letterbox the image in black. The resulting image is shown below.

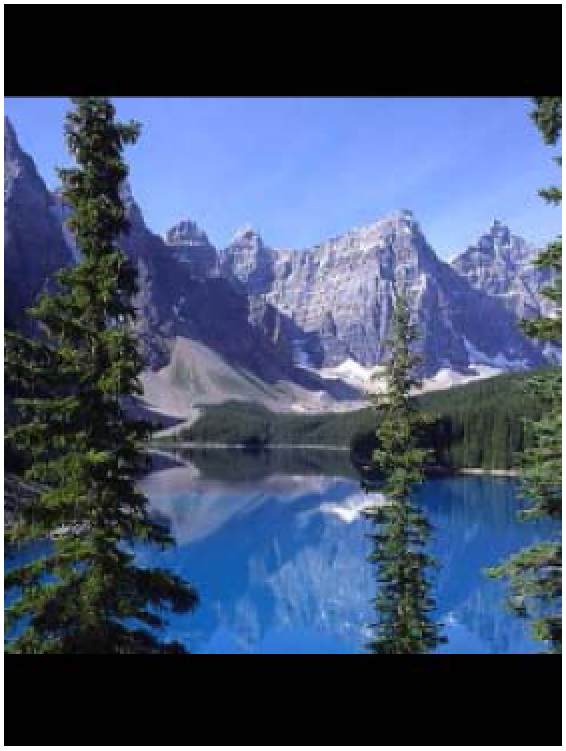


Figure 34-16. Example: Cropped QVGA

34.3.4 Upscale QVGA to VGA with Overlays

In this example, the image will be upscaled from QVGA to VGA resolution and displayed with the two overlays from the second example. Changes from the second example are shown in bold.

Table 34-7. Register Use for Conversion

Register	Value	Description
RGBBUF	*example1_rgb	Pointer to the output buffer.
RGBSIZE	0xFF2801E0	ALPHA=0xFF **WIDTH=0x280=640** **HEIGHT=0x1E0=480**
S0BUF	*morraine_y	Pointer to input Y buffer
S0UBUF	*morraine_u	Pointer to input U buffer
SOVBUF	*morraine_v	Pointer to input V buffer

S0PARAM	0x0000281E	WIDTH=0x28=40 (40*8=320 pixels)
		HEIGHT=0x1E=30 (30*8= 240 pixels)

Register	Value	Description
S0BACKGROUND	0x00000000	Black background region

S0CROP	0x0000503C	WIDTH=0x50=640pixels
		HEIGHT=0x3C=320pixels

S0SCALE	0x08000800	XSCALE=0x0800=2x scale
		YSCALE=0x0800=2x scale

S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*overlay1_rgb	Pointer to control graphic

OL0SIZE	0x**23**000A02	**XBASE=0x23=280pixels**
		WIDTH=0x0A=80 pixels
		HEIGHT=0x02=16pixels

OL0PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL1	*logo_rgb	Pointer to logo graphic

OL1SIZE	0x**1936**1D06	**XBASE=0x19=200pixels**
		YBASE=0x36=432pixels
		WIDTH=0x1D=232pixels
		HEIGHT=0x06=48pixels

OL1PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

Register	Value	Description
CTRL	0x000c9003	**SCALE=1** **CROP=1** S0_FORMAT=9 (YUV420) IRQ_ENABLE=1 ENABLE=1

The resulting image is shown in the figure below. The overlays have moved in this image and that the overall image size is now larger than before.

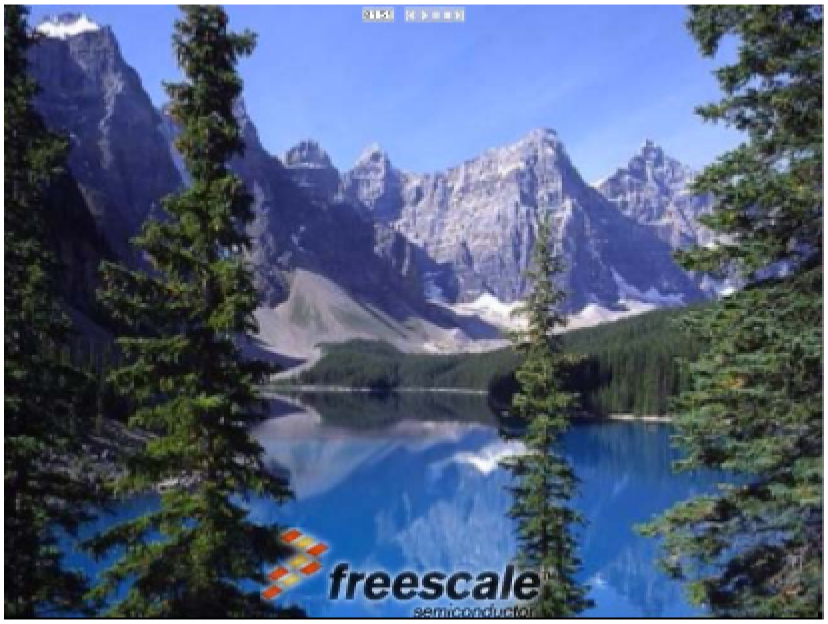


Figure 34-17. Example: Upscale QVGA to VGA with Overlays

34.3.5 Downscale VGA to WQVGA (480x272) to fill screen

In this example, a VGA image will be downscaled to fix the extents of a 480x272 WQVGA display. This means that the aspect ratio of the resulting image will not match that of the source image, therefore the scaling factors in the horizontal and vertical directions will differ from each other.

Table 34-8. Register Use for Conversion

Register	Value	Description
RGBBUF	*example_rgb	Pointer to the output buffer.

Register	Value	Description

RGBSIZE	0xFFf1E0110	ALPHA=0xFF
		WIDTH=0x1E0=480
		HEIGHT=0x110=272

S0BUF	*garden_y	Pointer to input Y buffer
S0UBUF	*garden_u	Pointer to input U buffer
SOVBUF	*garden_v	Pointer to input V buffer

S0PARAM	0x0000503C	WIDTH=0x50=80=640 pixels
		HEIGHT=0x3C=60=480 pixels

S0BACKGROUND	0x00000000	Black background region

S0CROP	0x00003C22	WIDTH=0x3C=480 pixels
		HEIGHT=0x22=272 pixels

S0SCALE	0x1C3C1555	YSCALE=0x1C3C=1/1.765x
		XSCALE=0x1555=1/1.333x

S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0PARAM	0x00000000	Overlay 0 disabled
OL1PARAM	0x00000000	Overlay 1 disabled
OL2PARAM	0x00000000	Overlay 2 disabled
OL3PARAM	0x00000000	Overlay 3 disabled
OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

CTRL	0x000C9003	SCALE=1
		CROP=1
		S0_FORMAT=9
		(YUV420) IRQ_ENABLE=1
		ENABLE=1

Note that the scaling factors are computed as (source/dest)*4096, thus in the horizontal direction 640/480*4096=5461=0x1555. In the vertical direction, the scaling factor is computed as 480/272*4096=7228=0x1C3C. The original source image and resulting scaled images are shown below:

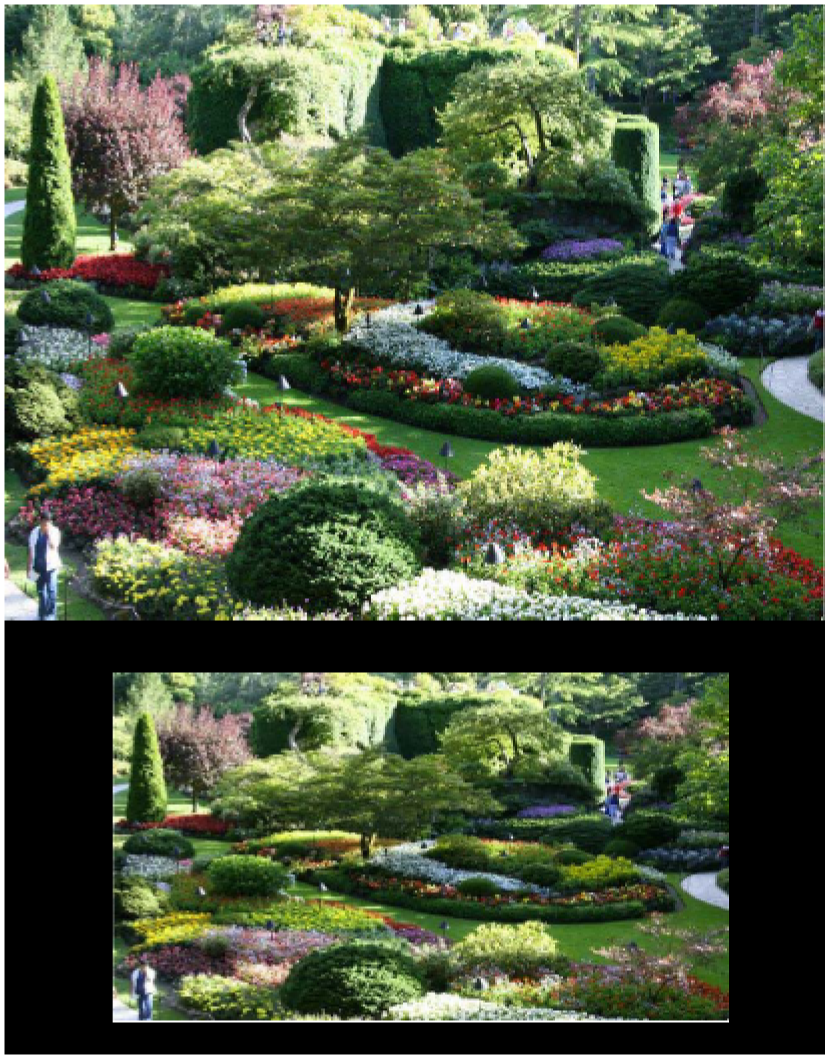


Figure 34-18. Example: Downscale VGA to WQVGA (480x272) to fill screen

34.3.6 Downscale VGA to QVGA with Overlapping Overlays

The final example will perform a 1/2x scaling of a VGA image to QVGA to maintain the aspect ratio. It will also add four overlays to present the image as if it were a photo album application.

Table 34-9. Register Use for Conversion

Register	Value	Description
RGBBUF	*example_rgb	Pointer to the output buffer.

RGBSIZE	0xFFf1E0110	ALPHA=0xFF
		WIDTH=0x1E0=480
		HEIGHT=0x110=272

S0BUF	*garden_y	Pointer to input Y buffer
S0UBUF	garden_u	Pointer to input U buffer
SOVBUF	garden_v	Pointer to input V buffer

S0PARAM	0x0000503C	WIDTH=0x50=80=640 pixels
		HEIGHT=0x3C=60=480 pixels

S0BACKGROUND	0x00000040	Dark Blue background region

S0CROP	0x0000281E	WIDTH=0x28=320 pixels
		HEIGHT=0x1E=240 pixels

S0SCALE	0x20002000	YSCALE=0x2000=1/2x
		XSCALE=0x1555=1/2x

S0OFFSET	0x08000800	XOFFSET=0x0800 (1/2 pixel)
		YOFFSET=0x0800 (1/2 pixel)

S0CSCCOEFF0 S0CSCCOEFF1 S0CSCCOEFF2	0x04030000 0x01230208 0x076b079b	YUV->RGB Coefficient Values
OL0	*prev_rgb	Pointer to previous graphic

OL0SIZE	0x0B1B0402	XBASE=0x0B=88pixels
		YBASE=0x1B=216pixels
		WIDTH=0x04=32 pixels
		HEIGHT=0x02=16pixels

OL0PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL1	*next_rgb	Pointer to next graphic

Register	Value	Description

OL1SIZE	0x2D1B0402	XBASE=0x2D=360pixels
		YBASE=0x1B=216pixels
		WIDTH=0x04=32 pixels
		HEIGHT=0x02=16pixels

OL1PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL2	*text_overlay	Pointer to text graphic

OL2SIZE	0x00000A1E	XBASE=0x00=0pixels
		YBASE=0x00=0pixels
		WIDTH=0x0A=80pixels
		HEIGHT=0x1E=240pixels

OL2PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL3	*border_rgb	Pointer to rectangular border graphic

OL3SIZE	0x0A00281E	XBASE=0x0A=80pixels
		YBASE=0x00=0pixels
		WIDTH=0x28=320pixels
		HEIGHT=0x1E=240pixels

OL3PARAM	0x0000FF01	ALPHA=0xFF
		FORMAT=0x0
		(RGB8888) ALPHA_CTRL=0
		(embedded alpha) ENABLE=1

OL4PARAM	0x00000000	Overlay 4 disabled
OL5PARAM	0x00000000	Overlay 5 disabled
OL6PARAM	0x00000000	Overlay 6 disabled
OL7PARAM	0x00000000	Overlay 7 disabled

Register	Value	Description

CTRL	0x000C9003	SCALE=1
		CROP=1
		S0_FORMAT=9
		(YUV420) IRQ_ENABLE=1
		ENABLE=1

The resulting image is shown below. The text is rendered in a transparent overlay (overlay #2) on the right side of the screen. The background color (#000040) is dark blue and shows through the overlay as the background color. Overlay #3 applies a thin white alpha-blended border around the image to frame it. Overlays #0 and #1 generate the Next> and <Prev images alpha blended onto the image. Because these overlays are higher priority (lower numbered) than the border, they are used at these locations instead of overlay #3.

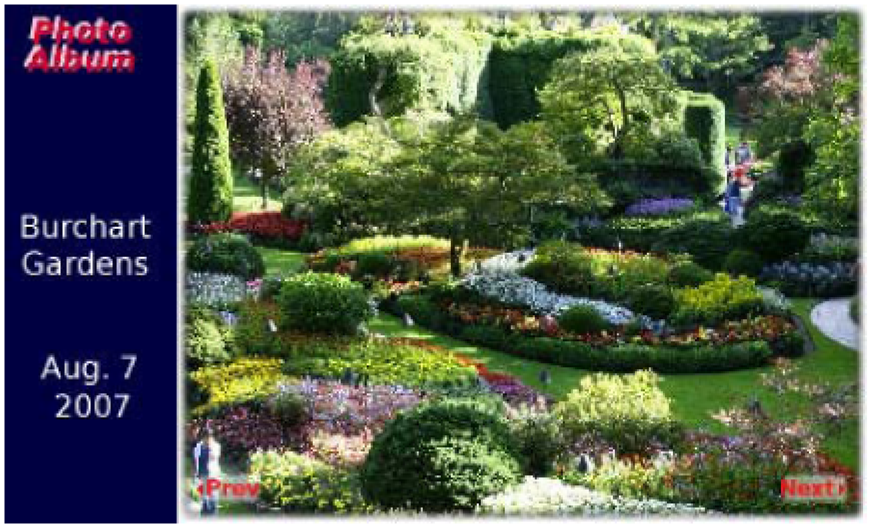


Figure 34-19. Example: Downscale VGA to QVGA with Overlapping Overlays

34.4 Programmable Registers

PXP Hardware Register Format Summary

HW_PXP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_A000	PXP Control Register 0 (HW_PXP_CTRL)	32	R/W	C000_0000h	34.4.1/2149
8002_A010	PXP Status Register (HW_PXP_STAT)	32	R/W	0000_0000h	34.4.2/2152
8002_A020	Output Frame Buffer Pointer (HW_PXP_OUTBUF)	32	R/W	0000_0000h	34.4.3/2154
8002_A030	Output Frame Buffer Pointer #2 (HW_PXP_OUTBUF2)	32	R/W	0000_0000h	34.4.4/2154

HW_PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_A040	PXP Output Buffer Size (HW_PXP_OUTSIZE)	32	R/W	0000_0000h	34.4.5/2155
8002_A050	PXP Source 0 (video) Input Buffer Pointer (HW_PXP_S0BUF)	32	R/W	0000_0000h	34.4.6/2156
8002_A060	Source 0 U/Cb or 2 Plane UV Input Buffer Pointer (HW_PXP_S0UBUF)	32	R/W	0000_0000h	34.4.7/2156
8002_A070	Source 0 V/Cr Input Buffer Pointer (HW_PXP_S0VBUF)	32	R/W	0000_0000h	34.4.8/2157
8002_A080	PXP Source 0 (video) Buffer Parameters (HW_PXP_S0PARAM)	32	R/W	0000_0000h	34.4.9/2158
8002_A090	Source 0 Background Color (HW_PXP_S0BACKGROUND)	32	R/W	0000_0000h	34.4.10/2158
8002_A0A0	Source 0 Cropping Register (HW_PXP_S0CROP)	32	R/W	0000_0000h	34.4.11/2159
8002_A0B0	Source 0 Scale Factor Register (HW_PXP_S0SCALE)	32	R/W	1000_1000h	34.4.12/2160
8002_A0C0	Source 0 Scale Offset Register (HW_PXP_S0OFFSET)	32	R/W	0000_0000h	34.4.13/2161
8002_A0D0	Color Space Conversion Coefficient Register 0 (HW_PXP_CSCCOEFF0)	32	R/W	0400_0000h	34.4.14/2162
8002_A0E0	Color Space Conversion Coefficient Register 1 (HW_PXP_CSCCOEFF1)	32	R/W	0123_0208h	34.4.15/2163
8002_A0F0	Color Space Conversion Coefficient Register 2 (HW_PXP_CSCCOEFF2)	32	R/W	079B_076Ch	34.4.16/2164
8002_A100	PXP Next Frame Pointer (HW_PXP_NEXT)	32	R/W	0000_0000h	34.4.17/2165
8002_A180	PXP S0 Color Key Low (HW_PXP_S0COLORKEYLOW)	32	R/W	00FF_FFFFh	34.4.18/2167
8002_A190	PXP S0 Color Key High (HW_PXP_S0COLORKEYHIGH)	32	R/W	0000_0000h	34.4.19/2168
8002_A1A0	PXP Overlay Color Key Low (HW_PXP_OLCOLORKEYLOW)	32	R/W	00FF_FFFFh	34.4.20/2168
8002_A1B0	PXP Overlay Color Key High (HW_PXP_OLCOLORKEYHIGH)	32	R/W	0000_0000h	34.4.21/2169
8002_A1D0	PXP Debug Control Register (HW_PXP_DEBUGCTRL)	32	R/W	0000_0000h	34.4.22/2170
8002_A1E0	PXP Debug Register (HW_PXP_DEBUG)	32	R	0000_0000h	34.4.23/2171
8002_A1F0	PXP Version Register (HW_PXP_VERSION)	32	R	0200_0000h	34.4.24/2171
8002_A200	PXP Overlay 0 Buffer Pointer (HW_PXP_OL0)	32	R/W	0000_0000h	34.4.25/2172
8002_A210	PXP Overlay 0 Size (HW_PXP_OL0SIZE)	32	R/W	0000_0000h	34.4.26/2173
8002_A220	PXP Overlay 0 Parameters (HW_PXP_OL0PARAM)	32	R/W	0000_0000h	34.4.27/2173
8002_A230	PXP Overlay 0 Parameters 2 (HW_PXP_OL0PARAM2)	32	R	0000_0000h	34.4.28/2175
8002_A240	PXP Overlay 1 Buffer Pointer (HW_PXP_OL1)	32	R/W	0000_0000h	34.4.29/2175
8002_A250	PXP Overlay 1 Size (HW_PXP_OL1SIZE)	32	R/W	0000_0000h	34.4.30/2176
8002_A260	PXP Overlay 1 Parameters (HW_PXP_OL1PARAM)	32	R/W	0000_0000h	34.4.31/2177
8002_A270	PXP Overlay 1 Parameters 2 (HW_PXP_OL1PARAM2)	32	R	0000_0000h	34.4.32/2178
8002_A280	PXP Overlay 2 Buffer Pointer (HW_PXP_OL2)	32	R/W	0000_0000h	34.4.33/2179
8002_A290	PXP Overlay 2 Size (HW_PXP_OL2SIZE)	32	R/W	0000_0000h	34.4.34/2179
8002_A2A0	PXP Overlay 2 Parameters (HW_PXP_OL2PARAM)	32	R/W	0000_0000h	34.4.35/2180

HW_PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8002_A2B0	PXP Overlay 2 Parameters 2 (HW_PXP_OL2PARAM2)	32	R	0000_0000h	34.4.36/2182
8002_A2C0	PXP Overlay 3 Buffer Pointer (HW_PXP_OL3)	32	R/W	0000_0000h	34.4.37/2182
8002_A2D0	PXP Overlay 3 Size (HW_PXP_OL3SIZE)	32	R/W	0000_0000h	34.4.38/2183
8002_A2E0	PXP Overlay 3 Parameters (HW_PXP_OL3PARAM)	32	R/W	0000_0000h	34.4.39/2184
8002_A2F0	PXP Overlay 3 Parameters 2 (HW_PXP_OL3PARAM2)	32	R	0000_0000h	34.4.40/2185
8002_A300	PXP Overlay 4 Buffer Pointer (HW_PXP_OL4)	32	R/W	0000_0000h	34.4.41/2186
8002_A310	PXP Overlay 4 Size (HW_PXP_OL4SIZE)	32	R/W	0000_0000h	34.4.42/2186
8002_A320	PXP Overlay 4 Parameters (HW_PXP_OL4PARAM)	32	R/W	0000_0000h	34.4.43/2187
8002_A330	PXP Overlay 4 Parameters 2 (HW_PXP_OL4PARAM2)	32	R	0000_0000h	34.4.44/2189
8002_A340	PXP Overlay 5 Buffer Pointer (HW_PXP_OL5)	32	R/W	0000_0000h	34.4.45/2189
8002_A350	PXP Overlay 5 Size (HW_PXP_OL5SIZE)	32	R/W	0000_0000h	34.4.46/2190
8002_A360	PXP Overlay 5 Parameters (HW_PXP_OL5PARAM)	32	R/W	0000_0000h	34.4.47/2191
8002_A370	PXP Overlay 5 Parameters 2 (HW_PXP_OL5PARAM2)	32	R	0000_0000h	34.4.48/2192
8002_A380	PXP Overlay 6 Buffer Pointer (HW_PXP_OL6)	32	R/W	0000_0000h	34.4.49/2193
8002_A390	PXP Overlay 6 Size (HW_PXP_OL6SIZE)	32	R/W	0000_0000h	34.4.50/2193
8002_A3A0	PXP Overlay 6 Parameters (HW_PXP_OL6PARAM)	32	R/W	0000_0000h	34.4.51/2194
8002_A3B0	PXP Overlay 6 Parameters 2 (HW_PXP_OL6PARAM2)	32	R	0000_0000h	34.4.52/2196
8002_A3C0	PXP Overlay 7 Buffer Pointer (HW_PXP_OL7)	32	R/W	0000_0000h	34.4.53/2196
8002_A3D0	PXP Overlay 7 Size (HW_PXP_OL7SIZE)	32	R/W	0000_0000h	34.4.54/2197
8002_A3E0	PXP Overlay 7 Parameters (HW_PXP_OL7PARAM)	32	R/W	0000_0000h	34.4.55/2198
8002_A3F0	PXP Overlay 7 Parameters 2 (HW_PXP_OL7PARAM2)	32	R	0000_0000h	34.4.56/2199

34.4.1 PXP Control Register 0 (HW_PXP_CTRL)

The CTRL register contains controls for the PXP module.

HW_PXP_CTRL: 0x000

HW_PXP_CTRL_SET: 0x004

HW_PXP_CTRL_CLR: 0x008

HW_PXP_CTRL_TOG: 0x00C

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
    HW_PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
    HW_PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: HW_PXP_CTRL – 8002_A000h base + 0h offset = 8002_A000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSVD2		INTERLACED_OUTPUT		INTERLACED_INPUT		BLOCK_SIZE	ALPHA_OUTPUT	IN_PLACE	DELTA	CROP	SCALE	UPSAMPLE	SUBSAMPLE
W			▨	▨								▨			▨	▨
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S0_FORMAT				VFLIP	HFLIP	ROTATE		OUTBUF_FORMAT				Reserved	NEXT_IRQ_ENABLE	IRQ_ENABLE	ENABLE
W													▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_CTRL field descriptions

Field	Description
31 SFTRST	Set this bit to zero to enable normal PXP operation. Set this bit to one (default) to disable clocking with the PXP and hold it in its reset (lowest power) state. This bit can be turned on and then off to reset the PXP block to its default state.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29–28 RSVD2	Reserved, always set to zero.
27–26 INTERLACED_ OUTPUT	Determines how the PXP writes it's output data. Output interlacing should not be used in conjunction with input interlacing. Splitting frames into fields is most efficient using output interlacing. 2-plane output formats AND interlaced output is NOT supported. 0x0 **PROGRESSIVE** — All data written in progressive format to the OUTBUF Pointer. 0x1 **FIELD0** — Interlaced output: only data for field 0 is written to the OUTBUF Pointer. 0x2 **FIELD1** — Interlaced output: only data for field 1 is written to the OUTBUF2 Pointer. 0x3 **INTERLACED** — Interlaced output: data for field 0 is written to OUTBUF and data for field 1 is written to OUTBUF2.
25–24 INTERLACED_ INPUT	When set, causes the fetch side of the PXP to fetch every other line from the source buffers. This effectively produces one field of interlaced output data. Scaling should NOT be enabled for interlaced operation and only overlays with boundaries on 8x16 multiples are supported.

HW_PXP_CTRL field descriptions (continued)

Field	Description
	0x0 **PROGRESSIVE** — All data will be read and processed in progressive format. 0x2 **FIELD0** — Interlaced, Field 0: only data for field 0 (even lines) is read/processed. 0x3 **FIELD1** — Interlaced, Field 1: only data for field 1 (odd lines) is read/processed.
23 BLOCK_SIZE	Select the block size to process. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 ALPHA_OUTPUT	Indicates that alpha component in output buffer pixels should be overridden by HW_PXP_OUTSIZE.ALPHA register. If 0, retain their alpha value from the computed alpha for that pixel.
21 IN_PLACE	When set, this enables the PXP to perform an alpha blend operation on an existing buffer (output buffer is set to S0 buffer). In this case, the PXP will perform the alpha blending of the overlays into the source buffer. Since only pixels containing an overlay are processed, the PXP does this very efficiently.
20 DELTA	Reserved for future use.
19 CROP	Indicates that the S0 plane should use the cropping register to provide the extents for the output S0 buffer cropping. If not set, the input video cropping extents will be inferred from the S0 WIDTH and HEIGHT fields. When scaling, the CROP bit and controls should be used to specify the scaled image size in the output buffer.
18 SCALE	This bit indicates that the output image should be scaled (only YUV/YCbCr images may be scaled -- RGB scaling is not supported). The XSCALE and YSCALE registers should be programmed accordingly. In addition, the CROP bit and the S0CROP registers should be programmed to ensure that the scaled image is properly cropped in the output buffer. When this bit is zero, the contents of the scaling registers are ignored.
17 UPSAMPLE	Reserved for future use.
16 SUBSAMPLE	Reserved for future use.
15–12 S0_FORMAT	Source 0 buffer format. To select between YUV and YCbCr formats, see bit 31 of the CSCCOEFF0 register. 0x0 **ARGB8888** — 32-bit pixels 0x1 **RGB888** — 32-bit pixels (unpacked 24-bit format) 0x4 **RGB565** — 16-bit pixels 0x5 **RGB555** — 16-bit pixels 0x8 **YUV422** — 16-bit pixels 0x9 **YUV420** — 16-bit pixels 0xA **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0xB **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0xC **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0xD **YUV2P420** — 16-bit pixels 0xE **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0xF **YVU2P420** — 16-bit pixels
11 VFLIP	Indicates that the output buffer should be flipped vertically (effect applied before rotation).
10 HFLIP	Indicates that the output buffer should be flipped horizontally (effect applied before rotation).

HW_PXP_CTRL field descriptions (continued)

Field	Description
9–8 ROTATE	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–4 OUTBUF_ FORMAT	Output framebuffer format. The UV byte lanes are synonymous with CbCr byte lanes for YUV output pixel formats. For example, the YUV2P420 format should be selected when the output is YCbCr 2-plane 420 output format. 0x0 **ARGB8888** — 32-bit pixels 0x1 **RGB888** — 32-bit pixels (unpacked 24-bit pixel in 32 bit DWORD.) 0x2 **RGB888P** — 24-bit pixels (packed 24-bit format) 0x3 **ARGB1555** — 16-bit pixels 0x4 **RGB565** — 16-bit pixels 0x5 **RGB555** — 16-bit pixels 0x7 **YUV444** — 32-bit pixels (1-plane XYUV unpacked) 0xA **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0xB **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0xC **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0xD **YUV2P420** — 16-bit pixels (2-plane UV) 0xE **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0xF **YVU2P420** — 16-bit pixels (2-plane VU)
3 Reserved	This bit is reserved. Reserved, always set to zero.
2 NEXT_IRQ_ ENABLE	Next command interrupt enable. When set, the PXP will issue an interrupt when a queued command initiated by a write to the PXP_NEXT register has been loaded into the PXP's registers. This interrupt also indicates that a new command may now be queued.
1 IRQ_ENABLE	Interrupt enable. NOTE: When using the HW_PXP_NEXT functionality to reprogram the PXP, the new value of this bit will be used and may therefore enable or disable an interrupt unintentionally.
0 ENABLE	Enables PXP operation with specified parameters. The ENABLE bit will remain set while the PXP is active and will be cleared once the current operation completes. Software should use the IRQ bit in the HW_PXP_STAT when polling for PXP completion.

34.4.2 PXP Status Register (HW_PXP_STAT)

The PXP Interrupt Status register provides interrupt status information.

HW_PXP_STAT: 0x010

HW_PXP_STAT_SET: 0x014

HW_PXP_STAT_CLR: 0x018

HW_PXP_STAT_TOG: 0x01C

This register provides PXP interrupt status and the current X/Y block coordinate that is being processed.

EXAMPLE

```
    HW_PXP_STAT_CLR(BM_PXP_STAT_IRQ);  // clear CSC interrupt
```

Address: HW_PXP_STAT – 8002_A000h base + 10h offset = 8002_A010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BLOCKX								BLOCKY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2								AXI_ERROR_ID				NEXT_IRQ	AXI_READ_ ERROR	AXI_WRITE_ ERROR	IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_STAT field descriptions

Field	Description
31–24 BLOCKX	Indicates the X coordinate of the block currently being rendered.
23–16 BLOCKY	Indicates the X coordinate of the block currently being rendered.
15–8 RSVD2	Reserved, always set to zero.
7–4 AXI_ERROR_ID	Indicates the AXI ID of the failing bus operation.
3 NEXT_IRQ	Indicates that a command issued with the Next Command functionality has been issued and that a new command may be initiated with a write to the PXP_NEXT register.
2 AXI_READ_ ERROR	Indicates PXP encountered an AXI read error and processing has been terminated.
1 AXI_WRITE_ ERROR	Indicates PXP encountered an AXI write error and processing has been terminated.
0 IRQ	Indicates current PXP interrupt status. The IRQ is routed through the pxp_irq when the IRQ_ENABLE bit in the control register is set.

34.4.3 Output Frame Buffer Pointer (HW_PXP_OUTBUF)

Output Framebuffer Pointer. This register points to the beginning of the output frame buffer. This pointer is used for progressive format and field 0 when generating interlaced output.

This register is used by the logic to point to the current output location for the output frame buffer.

EXAMPLE

```
HW_PXP_OUTBUF_WR( buffer );
```

Address: HW_PXP_OUTBUF – 8002_A000h base + 20h offset = 8002_A020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OUTBUF field descriptions

Field	Description
31–0 ADDR	Current address pointer for the output frame buffer. The address MUST be word-aligned for proper PXP operation.

34.4.4 Output Frame Buffer Pointer #2 (HW_PXP_OUTBUF2)

Output Framebuffer Pointer #2. This register points to the beginning of the output frame buffer for eitherfield 1 when generating interlaced output or for the UV buffer when in YUV 2-plane output modes. Both interlaced output AND 2-plane output modes are not supported. This register is NOT used as the pointer to the 2nd buffer when in LCDIF_HANDSHAKE mode.

This register is used by the logic to point to the current output location for the field 1 or UV output frame buffer.

EXAMPLE

```
HW_PXP_OUTBUF_WR( field0 );  // buffer for interlaced field 0
HW_PXP_OUTBUF2_WR( field1 ); // buffer for interlaced field 1
```

Address: HW_PXP_OUTBUF2 – 8002_A000h base + 30h offset = 8002_A030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OUTBUF2 field descriptions

Field	Description
31–0 ADDR	Current address pointer for the output frame buffer. The address MUST be word-aligned for proper PXP operation.

34.4.5 PXP Output Buffer Size (HW_PXP_OUTSIZE)

This register contains framebuffer size information for the output buffer (independent of the rotation). When rotating the framebuffer, user should set this register to final size (after rotation)

This register sets the size of the output frame buffer in pixels, not blocks. The frame buffer need not be a multiple of NxN pixels. Partial blocks will be written for output frame buffer sizes that are not divisable by N pixels in either dimension.

EXAMPLE

```
HW_PXP_OUTSIZE.U.WIDTH=320;   // set width
HW_PXP_OUTSIZE.U.HEIGHT=240;  // set height

HW_PX_OUTSIZE_WR( BF_PXP_OUTSIZE_WIDTH(320) |  BF_PXP_OUTSIZE_HEIGHT(240) );
```

Address: HW_PXP_OUTSIZE – 8002_A000h base + 40h offset = 8002_A040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ALPHA								WIDTH												HEIGHT											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OUTSIZE field descriptions

Field	Description
31–24 ALPHA	When generating an output buffer with an alpha component, the value in this field will be used.
23–12 WIDTH	Indicates number of horizontal PIXELS in the output image (independent of the rotation). The image size is not required to be a multiple of 8 pixels. The PXP will handle clipping the pixel output at this boundary.

HW_PXP_OUTSIZE field descriptions (continued)

Field	Description
11–0 HEIGHT	Indicates the number of vertical PIXELS in the output image (independent of the rotation). The image size is not required to be a multiple of 8 pixels. The PXP will handle clipping the pixel output at this boundary.

34.4.6 PXP Source 0 (video) Input Buffer Pointer (HW_PXP_S0BUF)

S0 Input Buffer Pointer. This should be programmed to the starting address of the RGB data or Y (luma) data for the S0 plane.

This register contains the pointer to the Luma/RGB buffer.

EXAMPLE

```
HW_PXP_S0BUF_WR(image_rgb); // RGB image

HW_PXP_S0BUF_WR(image_y);   // Y (luma) image data
HW_PXP_S0UBUF_WR(image_u);  // U (Cb) image data
HW_PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: HW_PXP_S0BUF – 8002_A000h base + 50h offset = 8002_A050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0BUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 RGB or Y (luma) input buffer. The address MUST be word-aligned for proper PXP operation.

34.4.7 Source 0 U/Cb or 2 Plane UV Input Buffer Pointer (HW_PXP_S0UBUF)

S0 Chroma (U/Cb/UV) Input Buffer Pointer. This register points to the beginning of the Source 0 U/Cb input buffer. In two plane operation, this register points to the beginning of the Source 0 UV chroma input buffer.

This register contains the pointer to the Chroma U/Cb or 2 plane UV buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE

```
HW_PXP_S0BUF_WR(image_y);   // Y (luma) image data
HW_PXP_S0UBUF_WR(image_u);  // U (Cb) image data
HW_PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: HW_PXP_S0UBUF – 8002_A000h base + 60h offset = 8002_A060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0UBUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 (video) U/Cb or 2 plane UV Chroma input buffer. The address MUST be word-aligned for proper PXP operation.

34.4.8 Source 0 V/Cr Input Buffer Pointer (HW_PXP_S0VBUF)

S0 Chroma (V/Cr) Input Buffer Pointer. This register points to the beginning of the Source 0 V/Cr input buffer. In two plane operation, this register is not used.

This register contains the pointer to the Chroma V/Cr buffer when performing colorspace conversion. This register is unused when processing RGB data.

EXAMPLE

```
HW_PXP_S0BUF_WR(image_y);   // Y (luma) image data
HW_PXP_S0UBUF_WR(image_u);  // U (Cb) image data
HW_PXP_S0VBUF_WR(image_v);  // V (Cr) image data
```

Address: HW_PXP_S0VBUF – 8002_A000h base + 70h offset = 8002_A070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0VBUF field descriptions

Field	Description
31–0 ADDR	Address pointer for the S0 (video) V/Cr Chroma input buffer. The address MUST be word-aligned for proper PXP operation.

34.4.9 PXP Source 0 (video) Buffer Parameters (HW_PXP_S0PARAM)

This register contains buffer information for the S0 input RGB/YUV buffer.

The S0 Parameter register contains the size of the S0 input buffer (WIDTH, HEIGHT) as well as provides an offset for the display of this buffer within the output frame buffer (XBASE,YBASE). All four values are in terms of NxN pixel blocks. In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_S0PARAM_WR(0x0101281E); // S0 buffer will appear at offset (8,8) in the output buffer.
                               // the size is 0x28 (40*8=320 pixels) by 0x1E (30*8=240 pixels)
```

Address: HW_PXP_S0PARAM – 8002_A000h base + 80h offset = 8002_A080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0PARAM field descriptions

Field	Description
31–24 XBASE	This field indicates the horizontal offset location (in NxN block) of the S0 buffer within the output frame buffer.
23–16 YBASE	This field indicates the vertical offset location (in NxN block) of the S0 buffer within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.10 Source 0 Background Color (HW_PXP_S0BACKGROUND)

S0 Background Pixel Color. This register provides a pixel value used when processing blocks outside of the region specified by the S0SIZE register. This value can effectively be used to set the color of the letterboxing region around a video image.

This register contains a pixel value to be used for any S0 blocks that fall outside the S0 extents. This is effectively a background or letterbox color.

EXAMPLE

```
HW_PXP_S0BACKGROUND_WR(0x00000000);  // letterbox is black
HW_PXP_S0BACKGROUND_WR(0x00800000);  // letterbox is dark red
HW_PXP_S0BACKGROUND_WR(0x00008000);  // letterbox is dark green
HW_PXP_S0BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: HW_PXP_S0BACKGROUND – 8002_A000h base + 90h offset = 8002_A090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COLOR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0BACKGROUND field descriptions

Field	Description
31–0 COLOR	Background color (in 32bpp format) for any pixels not in the S0 buffer range specified in the S0SIZE register.

34.4.11 Source 0 Cropping Register (HW_PXP_S0CROP)

This register contains controls for image/video cropping. XBASE and YBASE select the origin of the S0 buffer for PXP operations. The WIDTH and HEIGHT determine the visible size of the selected region in the output frame buffer. Software should program the input framebuffer cropped width/height values into these fields. Cropping is applied in the output buffer, therefore after any scaling operations. Scaled regions may need to be cropped to avoid artifacts at the edge of a scaled region.

The cropping register can be used to specify cropping extents for S0 plane in the output buffer. It is only used if the CROP bit is set in the PXP_CTRL register is set. When this bit is not set, no cropping of the input image will be performed and the PXP will default to using the S0 WIDTH and HEIGHT parameters. Cropping should always be used when scaling images since the PXP cannot determine the scaled image size. In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_S0CROP_WR(0x02021810);  // S0 origin is at (16,16) -- 0x0202
                               // output width is 192 (0x18->24*8=192 pixels)
                               // output height is 128 (0x10->16*8=128 pixels)
```

Address: HW_PXP_S0CROP – 8002_A000h base + A0h offset = 8002_A0A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0CROP field descriptions

Field	Description
31–24 XBASE	This field indicates the horizontal offset (in terms of N-pixel blocks) into the S0 buffer which is considered the origin of the image. This allows selection of a subset of a source image for processing.
23–16 YBASE	This field indicates the vertical offset (in terms of N-pixel blocks) into the S0 buffer which is considered the origin of the image. This allows selection of a subset of a source image for processing.
15–8 WIDTH	Ouput buffer cropped video width (in terms of N pixel blocks, non-rotated). This field should be programmed to the desired cropped width of the S0 plane in the output buffer. When scaling is not used, this value is effectively the width of the input buffer that should appear in the output buffer. For scaling operations, it's important that this field be programmed to the width of the scaled size of the S0 output image.
7–0 HEIGHT	Output buffer cropped video height (in terms of N pixel blocks, non-rotated). This field should be programmed to the desired cropped height of the S0 plan in the output buffer. When scaling is not used, this value is effectively the height of the input buffer that should appear in the output buffer. For scaling operations, it's important that this field be programmed to the height of the scaled size of the S0 output image.

34.4.12 Source 0 Scale Factor Register (HW_PXP_S0SCALE)

S0 Scale Factor. This register provides the scale factor for the S0 (video) buffer.

The maximum down scaling factor is 1/4 such that the output image in either axis is 1/4th the size of the source. The maximum up scaling factor is 2^12 for either axis. The reciprocal of the scale factor should be loaded into this register. To reduce the S0 buffer by a factor of two in the output frame buffer, a value of 10.0000_0000_0000 should be loaded into this register. The scale up by a factor of 4, the value of 1/4, or 00.0100_0000_0000, should be loaded into this register. To scale up by 8/5, the value of 00.1010_0000_0000 should be loaded.

EXAMPLE

```
HW_PXP_S0SCALE_WR(0x10001000);  // 1:1  scaling (0x1.000)
HW_PXP_S0SCALE_WR(0x08000800);  // 2x   scaling (0x0.800)
HW_PXP_S0SCALE_WR(0x20002000);  // 1/2x scaling (0x2.000)
```

Address: HW_PXP_S0SCALE – 8002_A000h base + B0h offset = 8002_A0B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD2	YSCALE														
W	▨															
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1	XSCALE														
W	▨															
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0SCALE field descriptions

Field	Description
31 RSVD2	Reserved, always set to zero.
30–16 YSCALE	This is a three bit integer and 12 bit fractional representation (###.####_####_####) of the Y scaling factor for the S0 source buffer. The maximum value programmed should be 4 since scaling down by a factor greater than 4 is not supported.
15 RSVD1	Reserved, always set to zero.
14–0 XSCALE	This is a three bit integer and 12 bit fractional representation (###.####_####_####) of the X scaling factor for the S0 source buffer. The maximum value programmed should be 4 since scaling down by a factor greater than 4 is not supported.

34.4.13 Source 0 Scale Offset Register (HW_PXP_S0OFFSET)

S0 Scale Offset. This register provides the initial scale offset for the S0 (video) buffer.

The X and Y offset provides the ability to access the source image with a per pixel or per sub-pixel granularity. To shift the source input image by a single pixel, for example, a value of 0x200 (for 8x8 block size) would be loaded into this offset field. For a 8x8 block size, 0x200 (or 1/8), will provide a fixed offset of 1 pixel for the entire PXP operation. With this setting for 16x16 block size, the value of 0x200 will provide a fixed offset of 2 pixels since 1/8 of a 16 pixel block is 2. The fixed offset values can also be used for sub-pixel adjustments in the bilinear scaling filter. For example, when scaling an image down by a factor of 2, an initial offset of 0x0 would result in sub-sampling every other pixel. If a fixed offset of 0x100 (1/16) with 8x8 block size selected is programmed, all pixels are used in scaling the final output pixel value.

EXAMPLE

```
HW_PXP_S0SCALE_WR(0x20002000);   // 1/2x scaling (0x2.000)
HW_PXP_S0OFFSET_WR(0x01000100);  // half-pixel offset for 8x8 block size in both X and Y to
ensure averaging versus pixel replication
```

Address: HW_PXP_S0OFFSET – 8002_A000h base + C0h offset = 8002_A0C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD2				YOFFSET												RSVD1				XOFFSET											
W	▨	▨	▨	▨													▨	▨	▨	▨												
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0OFFSET field descriptions

Field	Description
31–28 RSVD2	Reserved, always set to zero.
27–16 YOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the Y scaling offset. This represents a fixed block offset which gets added to the scaled block address to determine source data for the scaling engine.
15–12 RSVD1	Reserved, always set to zero.
11–0 XOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the X scaling offset. This represents a fixed block offset which gets added to the scaled block address to determine source data for the scaling engine.

34.4.14 Color Space Conversion Coefficient Register 0 (HW_PXP_CSCCOEFF0)

This register contains color space conversion coefficients in two's compliment notation.

The Coeffient 0 register contains coeffients used in the color space conversion algorithm. The Y and UV offsets are added to the source buffer to normalize them before the conversion. C0 is the coeffient that is used to multiply the luma component of the data for all three RGB components.

EXAMPLE

```
//  The equations used for Colorspace conversion are:
//     R = C0*(Y+YOFFSET)                   + C1(V+UV_OFFSET)
//     G = C0*(Y+YOFFSET) + C3(U+UV_OFFSET) + C2(V+UV_OFFSET)
//     R = C0*(Y+YOFFSET) + C4(U+UV_OFFSET)

   HW_PXP_CSCCOEFF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   HW_PXP_CSCCOEFF1_WR(0x01230208); // YUV coefficients: C1, C4
   HW_PXP_CSCCOEFF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: HW_PXP_CSCCOEFF0 – 8002_A000h base + D0h offset = 8002_A0D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	YCBCR_MODE	RSVD1		C0											UV_OFFSET [17:16]	
W		▨	▨													
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	UV_OFFSET[15:9]							Y_OFFSET								
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_CSCCOEFF0 field descriptions

Field	Description
31 YCBCR_MODE	Set to 1 when performing YCbCr conversion to RGB. Set to 0 when converting YUV to RGB data. This bit changes the behavior of the scaler when performing U/V scaling.
30–29 RSVD1	Reserved, always set to zero.
28–18 C0	Two's compliment Y multiplier coefficient. YUV=0x100 (1.000) YCbCr=0x12A (1.164)
17–9 UV_OFFSET	Two's compliment phase offset implicit for CbCr data. Generally used for YCbCr to RGB conversion. YCbCr=0x180, YUV=0x000 (typically -128 or 0x180 to indicate normalized -0.5 to 0.5 range)
8–0 Y_OFFSET	Two's compliment amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically -16 (0x1F0)

34.4.15 Color Space Conversion Coefficient Register 1 (HW_PXP_CSCCOEFF1)

This register contains color space conversion coefficients in two's compliment notation.

The Coeffient 1 register contains coeffients used in the color space conversion algorithm. C1 is the coeffient that is used to multiply the chroma (Cr/V) component of the data for the red component. C4 is the coeffient that is used to multiply the chroma (Cb/U) component of the data for the blue component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
HW_PXP_CSCCOEFF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
HW_PXP_CSCCOEFF1_WR(0x01230208); // YUV coefficients: C1, C4
```

```
   HW_PXP_CSCCOEFF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: HW_PXP_CSCCOEFF1 – 8002_A000h base + E0h offset = 8002_A0E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1					C1											RSVD0					C4										
W	▨	▨	▨	▨	▨												▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

HW_PXP_CSCCOEFF1 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C1	Two's compliment Red V/Cr multiplier coefficient. YUV=0x123 (1.140) YCbCr=0x198 (1.596)
15–11 RSVD0	Reserved, always set to zero.
10–0 C4	Two's compliment Blue U/Cb multiplier coefficient. YUV=0x208 (2.032) YCbCr=0x204 (2.017)

34.4.16 Color Space Conversion Coefficient Register 2 (HW_PXP_CSCCOEFF2)

This register contains color space conversion coefficients in two's compliment notation.

The Coeffient 2 register contains coeffients used in the color space conversion algorithm. C2 is the coeffient that is used to multiply the chroma (Cr/V) component of the data for the green component. C3 is the coeffient that is used to multiply the chroma (Cb/U) component of the data for the green component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
// NOTE: The default values for the CSCCOEFF2 register are incorrect.  C2 should be 0x76B and
 C3 should be 0x79C for proper operation.

   HW_PXP_CSCCOEFF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   HW_PXP_CSCCOEFF1_WR(0x01230208); // YUV coefficients: C1, C4
   HW_PXP_CSCCOEFF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: HW_PXP_CSCCOEFF2 – 8002_A000h base + F0h offset = 8002_A0F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1					C2											RSVD0					C3										
W	▨	▨	▨	▨	▨												▨	▨	▨	▨	▨											
Reset	0	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0

HW_PXP_CSCCOEFF2 field descriptions

Field	Description
31–27 RSVD1	Reserved, always set to zero.
26–16 C2	Two's compliment Green V/Cr multiplier coefficient. YUV=0x76B (-0.581) YCbCr=0x730 (-0.813)
15–11 RSVD0	Reserved, always set to zero.
10–0 C3	Two's compliment Green U/Cb multiplier coefficient. YUV=0x79C (-0.394) YCbCr=0x79C (-0.392)

34.4.17 PXP Next Frame Pointer (HW_PXP_NEXT)

This register contains a pointer to a data structure used to reload the PXP registers at the end of the current frame.

HW_PXP_NEXT: 0x100

HW_PXP_NEXT_SET: 0x104

HW_PXP_NEXT_CLR: 0x108

HW_PXP_NEXT_TOG: 0x10C

To enable this functionality, software must write this register while the PXP is processing the current data frame (if the PXP is currently idle, this will also initiate an immediate load of registers from the pointer). The process of writing this register (WRITE operation) will set a semaphore in hardware to notify the control logic that a register reload operation must be performed when the current frame processing is complete. At the end of a frame, the PXP will fetch the register settings from this location, signal an interrupt to software, then proceed with rendering the next frame of data. Software may cancel the reload operation by issuing a CLEAR operation to this register. SET and TOGGLE operations should not be used when addressing this register. All registers will be reloaded with the exception of the following: STAT, CSCCOEFFn, NEXT, VERSION. All other registers will be loaded in the order they appear in the register map. Once the pointer's contents have been loaded into the PXP's registers, the NEXT_IRQ interrupt will be issued (see the PXP_STATUS register).

EXAMPLE

```
// create register command structure in memory
  u32* pxp_commands0[48], pxp_commands1;
  u32  rc;

// initialize control structure for frame 0
  pxp_commands0[0] = ...; // CTRL
  pxp_commands0[1] = ...; // OUT Buffer
  ...
  pxp_commands0[47] = ..; // Overlay7 param2

// initialize control structure for frame 1
  pxp_commands1[0] = ...; // CTRL
  pxp_commands1[1] = ...; // OUT Buffer
  ...
  pxp_commands1[47] = ..; // Overlay7 param2

// poll until a command isn't queued
  while (rc=HW_PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  HW_PXP_NEXT_WR(pxp_commands0);  // enable PXP operation 0 via command pointer

// poll until first command clears
  while (rc=HW_PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  HW_PXP_NEXT_WR(pxp_commands1);  // enable PXP operation 1 via command pointer
```

Address: HW_PXP_NEXT – 8002_A000h base + 100h offset = 8002_A100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	POINTER[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POINTER[15:2]														RSVD	ENABLED
W															▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_NEXT field descriptions

Field	Description
31–2 POINTER	A pointer to a data structure containing register values to be used when processing the next frame. The pointer must be 32-bit aligned and should reside in on-chip or off-chip memory.
1 RSVD	Reserved, always set to zero.
0 ENABLED	Indicates that the next frame functionality has been enabled. This bit reflects the status of the hardware semaphore indicating that a reload operation is pending at the end of the current frame.

34.4.18 PXP S0 Color Key Low (HW_PXP_S0COLORKEYLOW)

This register contains the color key low value for the S0 buffer.

When processing an image, the if the PXP finds a pixel in the background image with a color that falls in the range from the S0COLORKEYLOW to S0COLORKEYHIGH range, it will substitute the color found in the matching overlay. If no overlay is present or if the overlay also matches its colorkey range, the s0background color is used.

EXAMPLE

```
                                    // colorkey values between
HW_PXP_S0COLORKEYLOW_WR (0x008000); // medium green and
HW_PXP_S0COLORKEYHIGH_WR(0x00FF00); // light green
```

Address: HW_PXP_S0COLORKEYLOW – 8002_A000h base + 180h offset = 8002_A180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PXP_S0COLORKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	Low range of RGB color key applied to S0 buffer. To disable S0 colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

34.4.19 PXP S0 Color Key High (HW_PXP_S0COLORKEYHIGH)

This register contains the color key high value for the S0 buffer.

When processing an image, the if the PXP finds a pixel in the background image with a color that falls in the range from the S0COLORKEYLOW to S0COLORKEYHIGH range, it will substitute the color found in the matching overlay. If no overlay is present or if the overlay also matches its colorkey range, the s0background color is used.

EXAMPLE

```
                                    // colorkey values between
HW_PXP_S0COLORKEYLOW_WR (0x008000); // medium green and
HW_PXP_S0COLORKEYHIGH_WR(0x00FF00); // light green
```

Address: HW_PXP_S0COLORKEYHIGH – 8002_A000h base + 190h offset = 8002_A190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_S0COLORKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	High range of RGB color key applied to S0 buffer. To disable S0 colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

34.4.20 PXP Overlay Color Key Low (HW_PXP_OLCOLORKEYLOW)

This register contains the color key low value for the OL buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the OLCOLORKEYLOW to OLCOLORKEYHIGH range, it will use the S0 pixel value for that location. If no S0 image is present or if the S0 image also matches its colorkey range, the s0background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
HW_PXP_OLCOLORKEYLOW_WR (0x000000); // black and
```

```
HW_PXP_OLCOLORKEYHIGH_WR(0x800000); // medium red
```

Address: HW_PXP_OLCOLORKEYLOW – 8002_A000h base + 1A0h offset = 8002_A1A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

HW_PXP_OLCOLORKEYLOW field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.
23–0 PIXEL	Low range of RGB color key applied to OL buffer. Each overlay has an independent colorkey enable.

34.4.21 PXP Overlay Color Key High (HW_PXP_OLCOLORKEYHIGH)

This register contains the color key high value for the OL buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the OLCOLORKEYLOW to OLCOLORKEYHIGH range, it will use the S0 pixel value for that location. If no S0 image is present or if the S0 image also matches its colorkey range, the s0background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
HW_PXP_OLCOLORKEYLOW_WR (0x000000); // black and
HW_PXP_OLCOLORKEYHIGH_WR(0x800000); // medium red
```

Address: HW_PXP_OLCOLORKEYHIGH – 8002_A000h base + 1B0h offset = 8002_A1B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1								PIXEL																							
W	▨	▨	▨	▨	▨	▨	▨	▨																								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OLCOLORKEYHIGH field descriptions

Field	Description
31–24 RSVD1	Reserved, always set to zero.

HW_PXP_OLCOLORKEYHIGH field descriptions (continued)

Field	Description
23–0 PIXEL	High range of RGB color key applied to OL buffer. Each overlay has an independent colorkey enable.

34.4.22 PXP Debug Control Register (HW_PXP_DEBUGCTRL)

This register controls the debug features of the PXP.

This register controls the PXP Debug features. This register is not intended for customer use.

EXAMPLE

Address: HW_PXP_DEBUGCTRL – 8002_A000h base + 1D0h offset = 8002_A1D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD[15:9]							RESET_TLB_STATS	SELECT							
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_DEBUGCTRL field descriptions

Field	Description
31–9 RSVD	Reserved, always set to zero.
8 RESET_TLB_ STATS	Fixed read-only value reflecting the MINOR field of the RTL version.
7–0 SELECT	Index into one of the PXP debug registers. The data for the selected register will be returned 0x0 **NONE** — None 0x1 **CTRL** — Control Debug 0x2 **S0REGS** — S0 Debug 0x3 **S0BAX** — S0 BA X Scale 0x4 **S0BAY** — S0 BA Y Scale 0x5 **PXBUF** — PXBUF Debug 0x6 **ROTATION** — Rotation Debug 0x7 **ROTBUF0** — Rotation Buffer 0

HW_PXP_DEBUGCTRL field descriptions (continued)

Field	Description
	0x8 **ROTBUF1** — Rotation Buffer 1 0xF0 **TLBCOUNT** — TLB Lookup Count 0xF1 **TLBHIT** — TLB Hit Count 0xF2 **TLBMISS** — TLB Miss Count 0xF3 **TLBLAT** — TLB Latency Count 0xF8 **TLBSTATE** — TLB State Information

34.4.23 PXP Debug Register (HW_PXP_DEBUG)

This register returns selected debug register values.

Debug register. Select the appropriate register in debug control and the values are returned here.

EXAMPLE

Address: HW_PXP_DEBUG – 8002_A000h base + 1E0h offset = 8002_A1E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_DEBUG field descriptions

Field	Description
31–0 DATA	Debug data

34.4.24 PXP Version Register (HW_PXP_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

This register indicates the RTL version in use.

EXAMPLE

```
if (HW_PXP_VERSION.B.MAJOR != 2) Error();
```

Address: HW_PXP_VERSION – 8002_A000h base + 1F0h offset = 8002_A1F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

34.4.25 PXP Overlay 0 Buffer Pointer (HW_PXP_OL0)

Overlay 0 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 0 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(0,overlay_ptr);
```

Address: HW_PXP_OL0 – 8002_A000h base + 200h offset = 8002_A200h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL0 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 0 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.26 PXP Overlay 0 Size (HW_PXP_OL0SIZE)

This register contains buffer size/location information for the Overlay 0 input buffer.

This register contains information about Overlay 0 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(0,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL0SIZE – 8002_A000h base + 210h offset = 8002_A210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL0SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.27 PXP Overlay 0 Parameters (HW_PXP_OL0PARAM)

This register contains buffer parameters for the Overlay 0 input buffer.

The S1 Overlay 0 Parameter register provides additional controls for Overlay 0.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(0,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL0PARAM – 8002_A000h base + 220h offset = 8002_A220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL0PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha

HW_PXP_OL0PARAM field descriptions (continued)

Field	Description
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.28 PXP Overlay 0 Parameters 2 (HW_PXP_OL0PARAM2)

This register contains buffer parameters for the Overlay 0 input buffer.

The Overlay 0 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL0PARAM2 – 8002_A000h base + 230h offset = 8002_A230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL0PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.29 PXP Overlay 1 Buffer Pointer (HW_PXP_OL1)

Overlay 1 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 1 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(1,overlay_ptr);
```

Address: HW_PXP_OL1 – 8002_A000h base + 240h offset = 8002_A240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL1 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 1 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.30 PXP Overlay 1 Size (HW_PXP_OL1SIZE)

This register contains buffer size/location information for the Overlay 1 input buffer.

This register contains information about Overlay 1 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(1,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL1SIZE – 8002_A000h base + 250h offset = 8002_A250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL1SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.

HW_PXP_OL1SIZE field descriptions (continued)

Field	Description
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.31 PXP Overlay 1 Parameters (HW_PXP_OL1PARAM)

This register contains buffer parameters for the Overlay 1 input buffer.

The S1 Overlay 1 Parameter register provides additional controls for Overlay 1.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(1,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL1PARAM – 8002_A000h base + 260h offset = 8002_A260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL1PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field.

HW_PXP_OL1PARAM field descriptions (continued)

Field	Description
	0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.32 PXP Overlay 1 Parameters 2 (HW_PXP_OL1PARAM2)

This register contains buffer parameters for the Overlay 1 input buffer.

The Overlay 1 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL1PARAM2 – 8002_A000h base + 270h offset = 8002_A270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL1PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.33 PXP Overlay 2 Buffer Pointer (HW_PXP_OL2)

Overlay 2 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 2 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(2,overlay_ptr);
```

Address: HW_PXP_OL2 – 8002_A000h base + 280h offset = 8002_A280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL2 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 2 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.34 PXP Overlay 2 Size (HW_PXP_OL2SIZE)

This register contains buffer size/location information for the Overlay 2 input buffer.

This register contains information about Overlay 2 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(2,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL2SIZE – 8002_A000h base + 290h offset = 8002_A290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL2SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.35 PXP Overlay 2 Parameters (HW_PXP_OL2PARAM)

This register contains buffer parameters for the Overlay 2 input buffer.

The S1 Overlay 2 Parameter register provides additional controls for Overlay 2.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(2,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL2PARAM – 8002_A000h base + 2A0h offset = 8002_A2A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL2PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels.

HW_PXP_OL2PARAM field descriptions (continued)

Field	Description
	0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.36 PXP Overlay 2 Parameters 2 (HW_PXP_OL2PARAM2)

This register contains buffer parameters for the Overlay 2 input buffer.

The Overlay 2 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL2PARAM2 – 8002_A000h base + 2B0h offset = 8002_A2B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL2PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.37 PXP Overlay 3 Buffer Pointer (HW_PXP_OL3)

Overlay 3 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 3 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(3,overlay_ptr);
```

Address: HW_PXP_OL3 – 8002_A000h base + 2C0h offset = 8002_A2C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL3 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 3 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.38 PXP Overlay 3 Size (HW_PXP_OL3SIZE)

This register contains buffer size/location information for the Overlay 3 input buffer.

This register contains information about Overlay 3 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(3,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL3SIZE – 8002_A000h base + 2D0h offset = 8002_A2D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL3SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.39 PXP Overlay 3 Parameters (HW_PXP_OL3PARAM)

This register contains buffer parameters for the Overlay 3 input buffer.

The S1 Overlay 3 Parameter register provides additional controls for Overlay 3.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(3,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL3PARAM – 8002_A000h base + 2E0h offset = 8002_A2E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	ALPHA								FORMAT				ENABLE_COLORKEY	ALPHA_CNTL		ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL3PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0

HW_PXP_OL3PARAM field descriptions (continued)

Field	Description
	0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.40 PXP Overlay 3 Parameters 2 (HW_PXP_OL3PARAM2)

This register contains buffer parameters for the Overlay 3 input buffer.

The Overlay 3 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL3PARAM2 – 8002_A000h base + 2F0h offset = 8002_A2F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL3PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.41 PXP Overlay 4 Buffer Pointer (HW_PXP_OL4)

Overlay 4 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 4 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(4,overlay_ptr);
```

Address: HW_PXP_OL4 – 8002_A000h base + 300h offset = 8002_A300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL4 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 4 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.42 PXP Overlay 4 Size (HW_PXP_OL4SIZE)

This register contains buffer size/location information for the Overlay 4 input buffer.

This register contains information about Overlay 4 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(4,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL4SIZE – 8002_A000h base + 310h offset = 8002_A310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R/W	XBASE								YBASE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL4SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.43 PXP Overlay 4 Parameters (HW_PXP_OL4PARAM)

This register contains buffer parameters for the Overlay 4 input buffer.

The S1 Overlay 4 Parameter register provides additional controls for Overlay 4.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(4,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL4PARAM – 8002_A000h base + 320h offset = 8002_A320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL4PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.

HW_PXP_OL4PARAM field descriptions (continued)

Field	Description
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.44 PXP Overlay 4 Parameters 2 (HW_PXP_OL4PARAM2)

This register contains buffer parameters for the Overlay 4 input buffer.

The Overlay 4 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL4PARAM2 – 8002_A000h base + 330h offset = 8002_A330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL4PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.45 PXP Overlay 5 Buffer Pointer (HW_PXP_OL5)

Overlay 5 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 5 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(5,overlay_ptr);
```

Address: HW_PXP_OL5 – 8002_A000h base + 340h offset = 8002_A340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL5 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 5 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.46 PXP Overlay 5 Size (HW_PXP_OL5SIZE)

This register contains buffer size/location information for the Overlay 5 input buffer.

This register contains information about Overlay 5 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(5,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL5SIZE – 8002_A000h base + 350h offset = 8002_A350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL5SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.47 PXP Overlay 5 Parameters (HW_PXP_OL5PARAM)

This register contains buffer parameters for the Overlay 5 input buffer.

The S1 Overlay 5 Parameter register provides additional controls for Overlay 5.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(5,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL5PARAM – 8002_A000h base + 360h offset = 8002_A360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL5PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0

HW_PXP_OL5PARAM field descriptions (continued)

Field	Description
	0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.48 PXP Overlay 5 Parameters 2 (HW_PXP_OL5PARAM2)

This register contains buffer parameters for the Overlay 5 input buffer.

The Overlay 5 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL5PARAM2 – 8002_A000h base + 370h offset = 8002_A370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL5PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.49 PXP Overlay 6 Buffer Pointer (HW_PXP_OL6)

Overlay 6 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 6 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(6,overlay_ptr);
```

Address: HW_PXP_OL6 – 8002_A000h base + 380h offset = 8002_A380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL6 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 6 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.50 PXP Overlay 6 Size (HW_PXP_OL6SIZE)

This register contains buffer size/location information for the Overlay 6 input buffer.

This register contains information about Overlay 6 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(6,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL6SIZE – 8002_A000h base + 390h offset = 8002_A390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL6SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.51 PXP Overlay 6 Parameters (HW_PXP_OL6PARAM)

This register contains buffer parameters for the Overlay 6 input buffer.

The S1 Overlay 6 Parameter register provides additional controls for Overlay 6.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(6,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL6PARAM – 8002_A000h base + 3A0h offset = 8002_A3A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL6PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0 0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.

HW_PXP_OL6PARAM field descriptions (continued)

Field	Description
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.52 PXP Overlay 6 Parameters 2 (HW_PXP_OL6PARAM2)

This register contains buffer parameters for the Overlay 6 input buffer.

The Overlay 6 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL6PARAM2 – 8002_A000h base + 3B0h offset = 8002_A3B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL6PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

34.4.53 PXP Overlay 7 Buffer Pointer (HW_PXP_OL7)

Overlay 7 Buffer Address Pointer. This register points to the beginning of the RGB Overlay 7 input buffer.

This register points to the is used by the logic to point to the current output location for the RGB frame buffer.

EXAMPLE

```
u32* overlay_ptr;
HW_PXP_OLn_WR(7,overlay_ptr);
```

Address: HW_PXP_OL7 – 8002_A000h base + 3C0h offset = 8002_A3C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL7 field descriptions

Field	Description
31–0 ADDR	Address pointer for the overlay 7 buffer. The address MUST be word-aligned for proper PXP operation.

34.4.54 PXP Overlay 7 Size (HW_PXP_OL7SIZE)

This register contains buffer size/location information for the Overlay 7 input buffer.

This register contains information about Overlay 7 indicating the size of the overlay (in NxN blocks) and the overlay's location within the output frame buffer (in NxN blocks). In 16 pixel block size mode, only the low 7 bits of each field can be used and the most significant bit must be set to 0.

EXAMPLE

```
HW_PXP_OLnSIZE_WR(7,0x10000401); // 32x8 overlay at offset +128+0
```

Address: HW_PXP_OL7SIZE – 8002_A000h base + 3D0h offset = 8002_A3D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	XBASE								YBASE								WIDTH								HEIGHT							
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL7SIZE field descriptions

Field	Description
31–24 XBASE	This field indicates the X-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
23–16 YBASE	This field indicates the Y-coordinate (in blocks) of the top-left NxN block in the overlay within the output frame buffer.
15–8 WIDTH	Indicates number of horizontal NxN blocks in the image (non-rotated).
7–0 HEIGHT	Indicates the number of vertical NxN blocks in the image (non-rotated).

34.4.55 PXP Overlay 7 Parameters (HW_PXP_OL7PARAM)

This register contains buffer parameters for the Overlay 7 input buffer.

The S1 Overlay 7 Parameter register provides additional controls for Overlay 7.

EXAMPLE

```
u32 olparam;
      olparam  = BF_PXP_OLnPARAM_ENABLE    (1);
      olparam |= BF_PXP_OLnPARAM_ALPHA_CNTL(BV_PXP_OLnPARAM_ALPHA_CNTL__ROPs);
      olparam |= BF_PXP_OLnPARAM_FORMAT    (BV_PXP_OLnPARAM_FORMAT__ARGB8888);
      olparam |= BF_PXP_OLnPARAM_ROP       (BV_PXP_OLnPARAM_ROP__XOROL);
HW_PXP_OLnPARAM_WR(7,olparam);  // enable overlay to perform XOR ROP using RGB8888 overlay
```

Address: HW_PXP_OL7PARAM – 8002_A000h base + 3E0h offset = 8002_A3E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD1												ROP			
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ALPHA								FORMAT				ENABLE_ COLORKEY	ALPHA_ CNTL		ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL7PARAM field descriptions

Field	Description
31–20 RSVD1	Reserved, always set to zero.
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CNTL field. 0x0 **MASKOL** — OL AND S0 0x1 **MASKNOTOL** — nOL AND S0 0x2 **MASKOLNOT** — OL AND nS0 0x3 **MERGEOL** — OL OR S0 0x4 **MERGENOTOL** — nOL OR S0 0x5 **MERGEOLNOT** — OL OR nS0 0x6 **NOTCOPYOL** — nOL 0x7 **NOT** — nS0 0x8 **NOTMASKOL** — OL NAND S0 0x9 **NOTMERGEOL** — OL NOR S0 0xA **XOROL** — OL XOR S0

HW_PXP_OL7PARAM field descriptions (continued)

Field	Description
	0xB **NOTXOROL** — OL XNOR S0
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE bits are set. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when enabled in the ALPHA_CNTL field.
7–4 FORMAT	Indicates the input buffer format for overlay 0. 0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x3 **ARGB1555** — 16-bit pixels with alpha 0x4 **RGB565** — 16-bit pixels without alpha 0x5 **RGB555** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this overlay. Pixels found in the overlay colorkey range will be displayed as transparent (the S0 pixel will be used).
2–1 ALPHA_CNTL	Determines how the alpha value is constructed for this overlay. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the OL pixel alpha value will be used to blend the OL with S0. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the overlay and S0 pixels.
0 ENABLE	Indicates that the overlay is active for this operation.

34.4.56 PXP Overlay 7 Parameters 2 (HW_PXP_OL7PARAM2)

This register contains buffer parameters for the Overlay 7 input buffer.

The Overlay 7 Parameter 2 register is reserved for future use.

Address: HW_PXP_OL7PARAM2 – 8002_A000h base + 3F0h offset = 8002_A3F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD																															
W																																
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_PXP_OL7PARAM2 field descriptions

Field	Description
31–0 RSVD	Reserved, always set to zero.

Chapter 35
Serial Audio Interface (SAIF)

35.1 Serial Audio Interface (SAIF) Overview

The SAIF provides the following functions:

- 3-, 4-, or 5-wire serial interface to industry's most common analog codecs.
- Transmit or receive (half-duplex).
- 16-bit to 24-bit serial stereo digital audio PCM play/record.
- Two, four, or six channels supported—three stereo pairs (mono supported in two-channel mode).
- Generic frame control supports DSP Compatible SIF, I^{2}S, left- and right-justified frame formats, as well as other non-standard variants of these formats.
- Master and slave BITCLK and LRCLK modes (clocks driven to codec or received from codec), as well as optional master MCLK mode.
- Supports a continuous range of sample rates from 8 KHz to 192 KHz using a high-resolution fractional divider driven by the PLL.
- Programmable over-sample rate for MCLK output (32x, 48x, 64x, 96x, 128x, 192x, 256x, 384x, and 512x) supports codecs found in systems with both audio and video.
- Four-entry FIFOs (per sample pair) buffer either two-channel sample pairs (17-bit through 24-bit PCM) or four-packed-channel sample pairs (16-bit PCM).
- Samples transferred to/from the FIFO through the APBX DMA interface, a FIFO service interrupt, or software polling.

Figure 35-1 shows the major functional blocks within the SAIF.

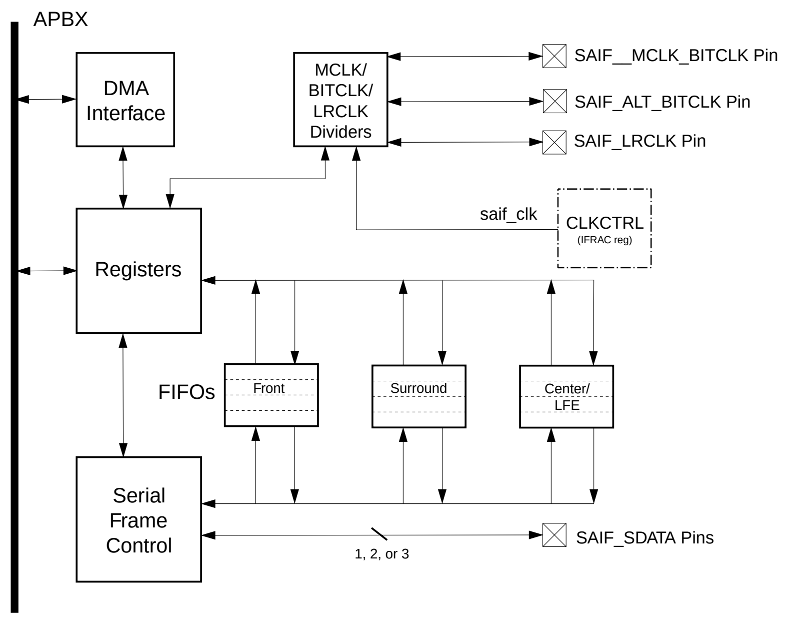


Figure 35-1. Serial Audio Interface (SAIF) Block Diagram

35.2 Operation

The SAIF is a half-duplex port, meaning it can either transmit or receive PCM audio, but not simultaneously. (Full-duplex support is availabe with two SAIF modules: one transmits while the other receives.) Data is communicated serially one sample at a time, alternating between left and right samples. One to three serial data lines (SDATA0–SDATA2) can be used to transmit either two (stereo/mono), four (stereo/surround), or six (stereo/surround/center/LFE) channels of digital PCM audio data. Samples boundaries are delineated by a left/right clock (LRCLK) pin, and individual bits within each sample are delineated by a bit clock (BITCLK) pin.

The LRCLK can be programmed to toggle every 16, 24, or 32 BITCLK transitions, and, because data ranges from 16 to 24 bits, serial data within each LRCLK period can either fully occupy the LRCLK cycle or cause the LRCLK period to contain BITCLK cycles in which no data is being communicated. Because of this, three basic types of sample frame formats can be programmed: I_2S, left-justified, and right-justified. However, many programming options exist to alter these basic frame types, such as the LRCLK signal polarity, BITCLK edge selection to drive/sample serial data, and sample justification/delay within an LRCLK period.

For codecs that do not contain their own PLL, or in applications where including a crystal oscillator to drive the codec is not desired, the SAIF can provide a master clock (MCLK) reference that can be configured from 512x down to 32x the audio data's sample rate. This master clock is used by the off-chip codec for all of its internal logic and to synchronize the BITCLK/LRCLK/SDATA inputs for DAC operation.

The digital PCM audio sample rate is determined by programming a fractional divider within the clock controller module.

35.2.1 Sample Rate Programming and Codec Clocking Operation

SAIF clocking is programmed in three blocks of the device:

- Clock control module (CLKCTRL)
- SAIF module
- Digital control module (DIGCTL)

The saif_clk (shown in Figure 35-1) is generated by the CLKCTRL block with a 16-bit fractional divider that divides down the 480 MHz PLL reference. The fractional divider minimizes system cost and power by eliminating the need for a second on-chip PLL. This fractional divider continuously selects which edge of the PLL reference clock (positive or negative) to use to best represent the oversample rate, such that less than 2 ns of correlated jitter occurs in saif_clk (jitter of the PLL plus periodic jitter of the fractional divide). This low level of jitter is required by most codec manufacturers to ensure a high SNR.

Table 35-1 shows the values to program the HW_CLKCTRL_SAIF_DIV bit field for all standard sample rates with either a base oversample rate of 512 or 384 times the sample rate.

These two base oversample values are the base rates typically required by codecs for the master clock (MCLK). An additional divider exists within the SAIF to generate sub-multiples of these two base rates if MCLK is required by the codec.

- The sub-rates that can be generated from 512x are: 256x, 128x, 64x, and 32x.
- The sub-rates for the 384x base rate are: 192x, 96x, and 48x.

The 384xFs base rate is common among systems that include MPEG1/2/4 audio and video, and AAC and AC-3 (Dolby Digital) audio. These MCLK rates are generated by programming the HW_SAIF_CTRL_BITCLK_BASE_RATE and HW_SAIF_CTRL_BITCLK_MULT_RATE bit fields.

Because there is a small amount of error in the sample rates generated by the fractional divider, software needs to periodically alter the HW_CLKCTRL_SAIF_DIV value higher or lower, depending on the difference between the required and actual sample rates. For an audio-only playback application, this adjustment need not ever be performed, because the frequency phase shift is imperceptible. However, it is needed for applications such as locking to and playing from an FM tuner, or audio/video applications where AV synchronization is required. For an audio/video application, it is typically accepted that audio and video cannot diverge any greater than ± 20 ms. Table 35-1 contains a column that lists the average elapsed time between each DIV adjustment required to keep within this 20-ms limit. Note that the smallest elapsed time is just under 20 seconds. Typically, software can be configured to periodically monitor the read and write pointers for the digital PCM audio buffer in OCRAM or SDRAM. When the pointers either become close to overrunning each other or close to the 20-ms AV divergence point, the DIV value should be increased or decreased incrementally (LSBs).

Table 35-1. HW_CLKCTRL_SAIF_DIV Values for Standard Sample Rates/Oversample Base Rates

Sample Rate (Hz)	Over-sample Base Rate Multiplier	saif_clk Re-quired (MHz)	Desired Fractional Divisor	Closest Actual Fractional Divisor	DIV Binary Value	DIV Hex Value	Error (ppm)	DIV Adjustment Frequency (seconds) (Notes 1 and 2)	
								DIV < Actual	DIV > Actual
192000	512	98.304	0.2048000000	0.2048034668	0011010001101110	0x346E	16.9	347.3	1181.5
	384	73.728	0.1536000000	0.1535949707	0010011101010010	0x2752	32.7	610.8	300.3
176200	512	90.2144	0.1879466667	0.1879425049	0011000000011101	0x301D	22.1	903.2	338.8
	384	67.6608	0.1409600000	0.1409606934	0010010000010110	0x2416	4.9	193.5	4066.0
128000	512	65.536	0.1365333333	0.1365356445	0010001011110100	0x22F4	16.9	210.9	1181.5
	384	49.152	0.1024000000	0.1024017334	0001101000110111	0x1A37	16.9	151.4	1181.5
96000	512	49.152	0.1024000000	0.1024017334	0001101000110111	0x1A38	16.9	151.4	1181.5
	384	36.864	0.0768000000	0.0767974854	0001001110101001	0x13A9	32.7	610.8	120.5
88100	512	45.1072	0.0939733333	0.0939788818	0001100000001111	0x180F	59.0	193.5	338.8
	384	33.8304	0.0704800000	0.0704803467	0001001000001011	0x120B	4.9	94.5	4066.0
64000	512	32.768	0.0682666667	0.0682678223	0001000101111010	0x117A	16.9	96.8	1181.5
	384	24.576	0.0512000000	0.0511932373	0000110100011011	0x0D1B	132.1	151.4	120.5

Sample Rate (Hz)	Over-sample Base Rate Multiplier	saif_clk Re-quired (MHz)	Desired Fractional Divisor	Closest Actual Fractional Divisor	DIV Binary Value	DIV Hex Value	Error (ppm)	DIV Adjustment Frequency (seconds) (Notes [1] and [2])	
								DIV < Actual	DIV > Actual
48000	512	24.576	0.0512000000	0.0511932373	0000110100011011	0x0D1B	132.1	151.4	120.5
	384	18.432	0.0384000000	0.0384063721	0000100111010101	0x09D5	165.9	86.4	120.5
44100	512	22.5792	0.0470400000	0.0470428467	0000110000001011	0x0C0B	60.5	75.8	330.5
	384	16.9344	0.0352800000	0.0352783203	0000100100001000	0x0908	47.6	420.1	52.0
32000	512	16.384	0.0341333333	0.0341339111	0000100010111101	0x08BD	16.9	46.5	1181.5
	384	12.288	0.0256000000	0.0256042480	0000011010001110	0x068E	165.9	46.5	120.5
24000	512	12.288	0.0256000000	0.0256042480	0000011010001110	0x068E	165.9	46.5	120.5
	384	9.216	0.0192000000	0.0191955566	0000010011101010	0x04EA	231.4	86.4	35.5
22050	512	11.2896	0.0235200000	0.0235137939	0000011000000101	0x0605	263.9	75.8	52.0
	384	8.4672	0.0176400000	0.0176391602	0000010010000100	0x0484	47.6	420.1	24.5
16000	512	8.192	0.0170666667	0.0170593262	0000010001011110	0x045E	430.1	46.5	43.1
	384	6.144	0.0128000000	0.0128021240	0000001101000111	0x0347	165.9	19.5	120.5
12000	512	6.144	0.0128000000	0.0128021240	0000001101000111	0x0347	165.9	19.5	120.5
	384	4.608	0.0096000000	0.0095977783	0000001001110101	0x0275	231.4	86.4	14.7
11025	512	5.6448	0.0117600000	0.0117645264	0000001100000011	0x0303	384.9	21.9	52.0
	384	4.2336	0.0088200000	0.0088195801	0000001001000010	0x0242	47.6	420.1	11.9
8000	512	4.096	0.0085333333	0.0085296631	0000001000101111	0x022F	430.1	46.5	14.7
	384	3.072	0.0064000000	0.0063934326	0000000110100011	0x01A3	1026.2	19.5	14.7

1. HW_CLKCTRL_SAIF_DIV_FRAC_EN=1
2. Average elapsed time between each DIV adjustment to maintain audio within ± 20 ms of sample rate.

BITCLK is used to launch or capture each bit of the serial PCM data, while LRCLK clocks the individual left/right samples (transitions at every sample boundary). For transmit, BITCLK and LRCLK are always driven by the SAIF. However, there are two different clocking modes for receive. In master mode, the SAIF drives both BITCLK and LRCLK, while in slave clock mode it is the responsibility of the codec to drive BITCLK and LRCLK to the SAIF. Note that, for any of these modes, an alternate MCLK reference can be multiplexed out to a pin to drive the codec's main system clock.

In slave clocking mode, the SAIF configures the BITCLK and LRCLK pins as inputs, and the off-chip codec is responsible for driving both clocks to the SAIF. The SAIF synchronizes these inputs and uses them to determine when to latch serial PCM data from the ADC for receive. The codec must be configured to run BITCLK either at 32xFs for 16-bit operation, 64xFs for 17-bit through 24-bit operation, or 48xFs for certain codecs for 16-bit through 24-bit operation.

In master clocking mode, it is the responsibility of the SAIF to drive both BITCLK and LRCLK out to the off-chip codec. In this mode, BITCLK is again programmed to transition at a the standard 32x, 48x, or 64x the sample rate.

On the device, two SAIF modules are instantiated on-chip. Each SAIF has a set of clock pins and can be operating in master mode simultaneously if they are connected to different off-chip codecs. Also, one of the two SAIFs can master or drive the clock pins while the other SAIF, in slave mode, receives clocking from the master SAIF. This also means that both SAIFs must operate at the same sample rate. Following are the valid configurations for SAIF1 and SAIF2 on the i.MX28:

- One SAIF in TX mode (is the default clock master) while the other SAIF is in RX slave mode and is internally controlled by the TX SAIF's BITCLK and LRCLK.
- One SAIF in RX master mode while the other SAIF is in RX slave mode and again is internally controlled by the RX master SAIF's BITCLK and LRCLK.
- Both SAIFs in RX slave mode, with BITCLK and LRCLK controlled by the off-chip codec.
- Both SAIFs in master mode, driving their BITCLK and LRCLK.
- Only one SAIF used (any configuration).

For any of these configurations, MCLK can also be output through a multiplexed pin to provide the clock reference for the off-chip codec.

Configuring the SAIF for transmit makes it the master by default. However, for receive, the SAIF can either be master or slave. For master mode, it drives BITCLK and LRCLK to the pins the off-chip codec, which uses the clocks to time when to serially transmit PCM data back to the SAIF. For slave mode, the BITCLK and LRCLK are pin inputs driven by the off-chip codec or the other SAIF (master), and the SAIF (slave) uses these clocks to determine when to latch each incoming bit and push the assembled PCM data to the FIFO.

The DIGCTL module contains the HW_DIGCTL_CTRL_SAIF_CLKMUX_SEL bit field which selects the clock sources for the SAIFs input BITCLK and LRCLK. Each SAIF has two source options: from external device through SAIF0 clock pins, from external device through SAIF1 clock pins.

Table 35-2. HW_DIGCTL_CTRL_SAIF_CLKMUX_SEL Programming

HW_SAIF_CLKMUX_SEL	Module Port		MODE
	SAIF0_BITCLK_IN, SAIF0_LRCLK_IN	SAIF1_BITCLK_IN, SAIF1_LRCLK_IN	
00	SAIF0 Clock pins	SAIF1 Clock pins	DIRECT
01	SAIF1 Clock pins	SAIF0 Clock pins	CROSSINPUT
10	SAIF0 Clock pins	SAIF0 Clock pins	EXTMSTR0
11	SAIF1 Clock pins	SAIF1 Clock pins	EXTMSTR

See Pin Control and GPIO Overview for instructions on which pins the two SAIFs use and how to configure them for operation.

The SAIF contains a four-entry FIFO for each channel pair that buffers data between the SAIF and the on-chip or off-chip RAM buffer used to supply or collect serial PCM audio data. Both the SAIF's register and DMA interface are clocked by APBX clock. To ensure the SAIF FIFO does not overrun or underrun, the minimum APBX clock frequency to sample rate frequency ratio is 22:1. Therefore, if the sample rate is configured to 192 KHz, then APBX must be set to 4.224 MHz or greater.

35.2.2 Transmit Operation

If the APBX DMA is to supply PCM data to the SAIF FIFO, the user first configures its corresponding DMA channel and initializes the buffer(s) of PCM data that are to be played. Next the SAIF control register is initialized, selecting the frame format and number of channel pairs, clearing the CLKGATE bit, and setting the RUN bit.

Once running, the BITCLK, LRCLK, and optional MCLK pins begin to transition, and null data is output to the off-chip analog DAC. The SAIF DMA interface requests PCM samples until the FIFO(s) is/are filled, and continues to request a sample (or sample pair for 16-bit operation) whenever an empty FIFO entry is available. Once valid PCM data resides within the bottom of the front channel pair FIFO, the current null sample left/right pair(s) are allowed to complete. At this point, the serialization frame control logic begins to output the first valid left sample.

In 16-bit operation, sample pairs are packed with the right samples occupying the upper halfword and left samples the lower halfword of the FIFO entries.

For 17-bit to 24-bit operation, each FIFO entry contains a sample that is right-justified (LSB in bit 0).

The first sample DMA-ed to the FIFO at the start of operation should always be a left sample, followed by a right, and so on. If four or six channel pairs are enabled, samples should be grouped with all left samples first, followed by all right samples (for example, front left, surround left, then center, followed by front right, surround right then LFE, and so on). As long as data resides within the FIFO(s), valid sample pairs continue to be output. If the FIFO(s) ever underflow or overflow, an interrupt occurs. At this point, the system software should shut down the SAIF, clear the FIFO(s), and then cleanly resume operation because there is not a way to prevent left/right swap of the PCM channels after this point. If the FIFO does underflow, null samples are output until valid data once again resides within the bottom of the FIFO. Any PCM value that is written to a full FIFO is discarded, preventing the top entry from be overwritten.

When the RUN bit is cleared at the end of operation, all PCM data corresponding to one sample collection (either two, four, or six channel pairs) that are currently being transmitted are allowed to complete before operation ceases and the BITCLK, LRCLK, and SDATA pins stop transitioning.

Alternately, software can be used to maintain the FIFO(s) if the DMA is not used, either by responding to an interrupt that is issued whenever an empty FIFO entry exists, or by polling a FIFO status bit.

35.2.3 Receive Operation

If the APBX DMA is to collect PCM data from the SAIF FIFO(s), the user first configures its corresponding DMA channel and allocates the buffer(s) where PCM data is to be recorded. Next the SAIF control register is initialized, selecting the frame format and number of channel pairs, selecting whether the SAIF is BITCLK/LRCLK master or slave, clearing the CLKGATE bit, and setting the RUN bit.

Once running, the SAIF either waits until the BITCLK and LRCLK pins begin to transition (slave mode) or begins to toggle BITCLK and LRCLK (master mode). In either case, once an LRCLK edge that corresponds to the start of a left sample is detected, the SAIF frame-control logic begins to assemble the sample in its serial shift register. Each time the LRCLK pin toggles, a new sample is pushed to the FIFO(s).

In 16-bit operation, sample pairs are packed with the right samples occupying the upper halfword and left samples the lower halfword of the FIFO entries.

For 17-bit to 24-bit operation, samples are placed in individual FIFO entries, are right justified (LSB in bit 0), and the unused MSBs are zero-filled.

The first sample pushed to the FIFO at the start of operation is always a left sample, followed by a right, and so on. If 4 or 6 channel pairs are enabled, sample pairs are grouped when pushed to the FIFO with all left samples first, followed by all right samples (for example, front left, surround left, then center, followed by front right, surround right then LFE, and so on). As long as the BITCLK and LRCLK pins continue to transition, data is collected within the FIFO. If the FIFO ever overflows or underflows, an interrupt occurs. At this point, the system software should shut down the SAIF, clear the FIFO(s), and then cleanly resume operation because there is not a way to determine left from right PCM channel data within the FIFO after this point. If the FIFO does overflow, any PCM value that is pushed to the full FIFO is discarded, not allowing the top entry to be overwritten. If the FIFO underflows, the read of the empty FIFO returns all zeros.

When the run bit is cleared at the end of operation, all PCM data corresponding to one sample collection (either two, four, or six channel pairs) that are currently being serially received are allowed to complete and are stored to the FIFO before operation ceases.

Software can be used to empty the FIFO if the DMA is not used, either by responding to an interrupt that is issued whenever an empty FIFO entry exists, or by polling a FIFO status bit.

35.2.4 DMA Interface

Both SAIFs on the device are assigned to APBX DMA channels. See AHB-to-APBX Bridge Overview for the DMA channel assignments.

Once the DMA channel and SAIF are programmed (except for the RUN bit), operation can be initiated either by setting the SAIF's RUN bit or by signaling a kick from the DMA channel.

The HW_SAIF_CTRL_DMAWAIT_COUNT bit field can be programmed to wait 0 to 31 APBX clock cycles between each DMA request. This feature acts as a throttle on the bandwidth required by the SAIF to allow delays such that DMA requests from other modules can be serviced by the DMA controller.

35.2.5 PCM Data FIFO

The SAIF contains three 4-entry by 32-bit wide FIFOs. These FIFOs serve as a buffer to ensure data is not corrupted if the DMA cannot service the SAIF before the next sample or sample pair is processed by the SAIF. Access to the FIFOs is achieved through read/writes of the 32-bit HW_SAIF_DATA register. Writes place PCM values at the top of the FIFOs, and reads take them from the bottom of the FIFOs. Each FIFO is used to store different sets of left/right channel pairs. FIFO1 stores the stereo or front channels, FIFO2 the surround or rear channels, and FIFO3 the center and low frequency effect (LFE) or subwoofer channels (see Figure 35-1). Read/write accesses are made to the FIFOs in a round-robin fashion such that all left channel samples are transferred first including the center channel, followed by all right channel samples including the LFE channel.

In 16-bit operation, sample pairs are packed with the right samples occupying the upper halfword and left samples the lower halfword of the FIFO entries.

For 17-bit to 24-bit operation, sample pairs are placed in individual FIFO entries, left channel first then followed by the right channel, and are right justified (LSB in bit 0) in each FIFO entry.

The FIFO underflow, overflow, and service interrupt status bits reside within the HW_SAIF_STAT register.

35.2.6 Serial Frame Formats

There are six types of serial PCM frames that can be transmitted or received. The three basic formats are I^{2}S, left-justified, and right-justified. Because there are two variations of a frame based on whether or not the data consumes the entire frame width, these three formats have two frame types each that make up the six basic frame types. One variation exists for 16-bit and 24-bit serial PCM data that consumes the whole frame width, and the other for 17-bit to 24-bit serial PCM data that does not. Recall that for 16-bit operation, BITCLK is either 32xFs or 48xFs, while for 17-bit to 24-bit, it is either 48xFs or 64xFs. These six types of serial PCM frames are shown in Figure 35-2.

- In left-justified (LJ) format, the serial PCM data is left-justified within the sample's start and end point indicated by the LRCLK. The first bit of PCM data is coincident with the first BITCLK period after LRCLK transitions.
- Conversely, in right-justified (RJ) format, the last bit of PCM data in a sample is coincident with the last BITCLK period before LRCLK transitions, indicating the start of the next sample.
- I^{2}S format is simply a variant of LJ format, in that the first BITCLK period after the LRCLK transitions, is a null wait state, followed by the first serial PCM bit during the next BITCLK period.

For both LJ and RJ formats, LRCLK is high for left samples and low for right samples. For I^{2}S, it is the opposite: LRCLK is low for left samples and high for right samples.

When the programmed data size is smaller than the frame size or number of BITCLKs per LRCLK, there are BITCLK periods within the sample frame in which no data is being transmitted or received. This occurs in 48xFs mode when that data is less than 24 bits, and for all data sizes allowed in 64xFs mode. For LJ and I^{2}S modes, this occurs after the sample has been transmitted or received, while for RJ mode, this occurs prior to the sample being transmitted or received.

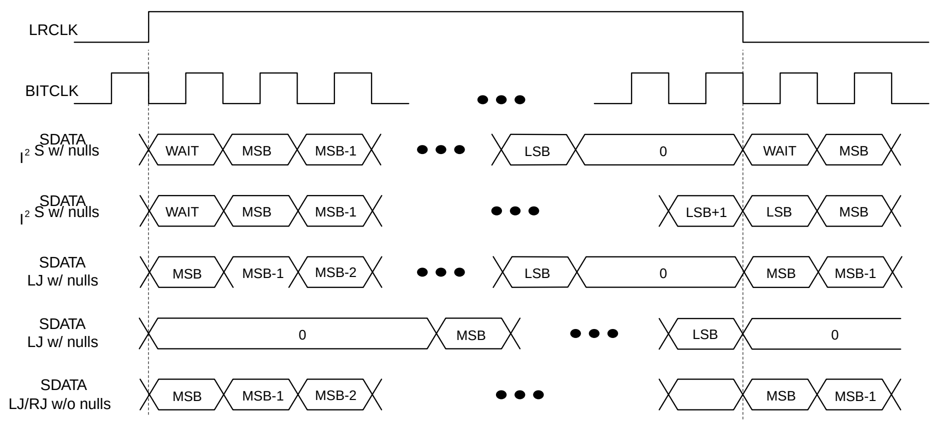
For 16-bit (32xFs mode) operation and 24-bit (48xFs mode) operation, data is always being transmitted, so that LJ and RJ modes are identical. I^{2}S is a special case for these modes. At the start of transmit or receive, the first BITCLK period after LRCLK transitions is a null or wait state cycle in which no PCM data is present. However, this means one too few BITCLK periods remain to transmit or receive data before LRCLK transitions. As a result, the protocol dictates that the last serial PCM bit of each sample be transmitted or received during the BITCLK wait state at the start of the *next* sample.

Another format, the **DSP Compatible Serial Interface Format**, is now supported by the SAIF module. In DSP format, the Right channel data immediately follows the left channel data as in 16 and 24 bit operation with BITCLK being 32xFs and 48xFs respectively. This format is one of the type where a synch pulse on the LRCLK is followed by a left and right data pair. The pulse is 1 BITCLK clock cycle wide, as oppose to 16 or 24 BITCLK clock cycles in the other formats. DSP format has 2 data modes: A and B. Mode A resembles LJ format where the first bit is aligned with the LRCLK pulse. Mode B resembles I2S format where the first bit is delayed by 1 BITCLK cycle.

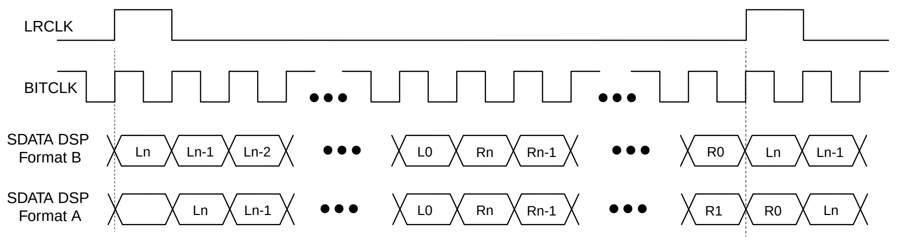
Additionally data can be programmed to be transmitted or received MSB or LSB first. The bits to program frame format reside within the HW_SAIF_CTRL register.

35.2.7 Pin Timing

The figure below shows the six basic frame formats supported by the SAIF. Keep in mind that for 16-bit operation, BITCLK runs at either 32x or 48x the sample rate, and for 17-bit through 24-bit operation, it runs at either 48x or 64x the sample rate (that is, the clock frequency relationship of differing data sizes is not shown here).



I2S, Left and R ight Justified Formats
Note: LRCLK_POLARITY=1 and B ITCLK_EDGE=0 for this example.



DSP Compatible Serial Interface Format
Note: LRCLK_POLARITY=1 and B ITCLK_EDGE=1 required.

Figure 35-2. Frame Formats Supported by SAIF

35.3 Programmable Registers

SAIF Hardware Register Format Summary

SAIF0 base address is 0x80042000;SAIF1 base address is 0x80046000

HW_SAIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8004_2000	SAIF Control Register (HW_SAIF_CTRL)	32	R/W	C000_0000h	35.3.1/2212
8004_2010	SAIF Status Register (HW_SAIF_STAT)	32	R/W	8000_0000h	35.3.2/2216
8004_2020	SAIF Data Register (HW_SAIF_DATA)	32	R/W	0000_0000h	35.3.3/2218
8004_2030	SAIF Version Register (HW_SAIF_VERSION)	32	R	0101_0000h	35.3.4/2219

35.3.1 SAIF Control Register (HW_SAIF_CTRL)

The SAIF Control Register controls the frame format and operation of the three-wire serial audio interface.

HW_SAIF_CTRL: 0x000

HW_SAIF_CTRL_SET: 0x004

HW_SAIF_CTRL_CLR: 0x008

HW_SAIF_CTRL_TOG: 0x00C

The SAIF Control Register is used to configure the SAIF's input/output frame format, MCLK, BITCLK and LRCLK, and interrupt enables.

EXAMPLE

```
HW_SAIF_CTRL.RUN = 1; // start SAIF operation
```

Address: HW_SAIF_CTRL – 8004_2000h base + 0h offset = 8004_2000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	BITCLK_MULT_RATE			BITCLK_BASE_RATE	FIFO_ERROR_IRQ_EN	FIFO_SERVICE_IRQ_EN	RSRVD2			DMAWAIT_COUNT				
W									▨	▨	▨					
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R/W	CHANNEL_NUM_SELECT		LRCLK_PULSE	BIT_ORDER	DELAY	JUSTIFY	LRCLK_POLARITY	BITCLK_EDGE	WORD_LENGTH				BITCLK_48XFS_ENABLE	SLAVE_MODE	READ_MODE	RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SAIF_CTRL field descriptions

Field	Description
31 SFTRST	Setting this bit to 1 forces a reset to the entire block. SFTRST has no effect on CLKGATE. Also, the SFTRST bit may be written when CLKGATE=1. This bit must be cleared to 0 for normal operation.
30 CLKGATE	This bit gates the clocks to the SAIF to save power when the clocks are not in use. When set to 1, this bit gates off the clocks to the block. When this bit is cleared to 0, the block receives its clocks for normal operation.
29–27 BITCLK_MULT_RATE	BITCLK Mutiplier Rate. This bit field selects the multiple of the base frequency rate of BITCLK for transmit mode and receive master clock mode (READ_MODE=1, SLAVE_MODE=0), or if the alternate BITCLK pin is used, it selects the multiple of the base frequency rate of MCLK (any mode). When BITCLK_BASE_RATE = 0 (32x base rate): 000=512 x Fs, 001=256 x Fs, 010=128 x Fs, 011=64 x Fs, 100=32 x Fs, 101-111=reserved. When BITCLK_BASE_RATE = 1 (48x base rate): 000=384 x Fs, 001=192 x Fs, 010=96 x Fs, 011=48 x Fs, 100-111=reserved. When the SAIF_BITCLK_MCLK pin is used as BITCLK, this field should be programmed to 32x for 16-bit data, 48x for 16-bit to 24-bit data (BITCLK_48XFS_ENABLE=1), and 64x for 17-bit to 24-bit data, depending on the modes supported by the off-chip codec.

HW_SAIF_CTRL field descriptions (continued)

Field	Description
	When the SAIF_BITCLK_MCLK pin is used as MCLK (and the alternate BITCLK pin mux function is enabled), any oversample rate can be selected as dictated by the codec's required MCLK frequency. Note that the clock controller block should be programmed with the correct DIV value to produce the correct oversample clock base frequency (either 512x or 384x the sample rate) to the SAIF.
26 BITCLK_BASE_ RATE	BITCLK Base Rate. This bit selects the base frequency rate at which the BITCLK pin toggles when configured as an output (either in transmit mode or in receive master clock mode), or if the alternate BITCLK pin is used, it selects the base frequency rate at which both the MCLK and alternate BITCLK pin toggles. 0 = BITCLK/MCLK base frequency rate is in multiples of 32x the sample rate. 1 = BITCLK/MCLK base frequency rate in in multiples of 48x the sample rate. This bit field is used in conjunction with the BITCLK_MULT_RATE field to program the BITCLK/MCLK output frequency.
25 FIFO_ERROR_ IRQ_EN	Set this bit to one to enable a SAIF interrupt request on FIFO overflow or underflow status condition.
24 FIFO_SERVICE_ IRQ_EN	Set this bit to one to enable a SAIF interrupt request to service the FIFO when it contains an empty entry (for transmit) or a full entry (for receive).
23–21 RSRVD2	Reserved.
20–16 DMAWAIT_ COUNT	DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay after a DMA request has been serviced and before the next request is granted. This field acts as a throttle on the bandwidth consumed by the SAIF block. This field can be loaded by the DMA.
15–14 CHANNEL_ NUM_SELECT	Channel Number Select. This bit field selects the number of channel pairs (left and right) that are transmitted and/or received by the SAIF. 00 = One channel pair (stereo) 01 = Two channel pairs (front, surround) 10 = Three channel pairs (front, surround, center/lfe) 11 = Reserved
13 LRCLK_PULSE	SAIF LRCLK Pulse Select. This bit used for the PCM Format where the LRCLK is high for one BITCLK cycle at the beginning of the left PCM sample only. 0 = LRCLK toggles between Left and Right PCM samples 1 = LRCLK pulses high for 1 Bitclk cycle at the beginning of a PCM sample pair (left,Right).
12 BIT_ORDER	SAIF PCM Data Serial Bit Order. This bit selects whether PCM data is serially transmitted or received LSB or MSB first. 0 = MSB first 1 = LSB first Note that the two's complement audio data written to and read from the FIFO is always ordered MSB to LSB (LSB located in bit 0 for 17-bit through 24-bit operation, and in bits 15 and 0 for 16-bit operation).
11 DELAY	SAIF Data Delay. In left-justified mode, this bit selects whether or not serial PCM data transmission/reception is delayed by one BITCLK period each LRCLK frame (to generate I2S serial operation).

HW_SAIF_CTRL field descriptions (continued)

Field	Description
	0 = Data is not delayed. MSB of serial sample is output/input coincident with LRCLK transition (left-justified mode) 1 = Data is delayed one BITCLK period. MSB of serial serial sample is output/input one BITCLK period after LRCLK transitions (I2S mode). Note that this bit is ignored in right-justified mode (JUSTIFY=1).
10 JUSTIFY	SAIF Data Justification. This bit selects whether serial PCM data is left- or right-justified within each sample's LRCLK frame. 0 = Data is left-justified (start or MSB of serial sample transmission/reception coincides with LRCLK transition) 1 = Data is right-justified (end or LSB of serial sample transmission/reception coincides with LRCLK transition).
9 LRCLK_ POLARITY	SAIF LRCLK Polarity Select. This bit selects which LRCLK levels (high/low) correspond to left and right PCM samples. 0 = Left low/right high 1 = Left high/right low.
8 BITCLK_EDGE	SAIF BITCLK Edge Select. For both transmit and receive, this bit selects the BITCLK edge upon which serial PCM data changes. For receive, data is sampled and stored to the receive FIFO on the opposite edge as selected by BITCLK_EDGE that corresponds to the midpoint of the data. 0 = TX: data is driven (changes) on falling-edges of BITCLK; RX: data is sampled on rising-edges of BITCLK 1 = TX: data is driven (changes) on rising-edges of BITCLK; RX: data is sampled on falling-edges of BITCLK.
7–4 WORD_LENGTH	SAIF data size. Selects one of nine PCM data widths from 16-bit to 24-bit to serially input or output from/to a codec. 17-bit to 24-bit PCM data should be right-justified (LSB in bit 0) when it is DMAed or written to the HW_SAIF_DATA register. These samples should be interleaved starting with a left sample first, followed by a right, then left and so on. For 16-bit PCM data, stereo pairs should be constructed with the right sample in the upper half-word (bits 31-16) and the left sample in the lower half word (bits 15-0). 0000 = 16-bit 0001 = 17-bit 0010 = 18-bit 0011 = 19-bit 0100 = 20-bit 0101 = 21-bit 0110 = 22-bit 0111 = 23-bit 1000 = 24-bit 1001-1111 = Reserved but defaults to 24-bit.
3 BITCLK_48XFS_ ENABLE	BITCLK 48x Sample Rate Enable. For 384x base frequency multiples, this bit enables generation of 48 BITCLKs per sample pair (24 BITCLKs per channel or LRCLK transition) when the SAIF is BITCLK master. This bit is ignored for the following cases: BITCLK_BASE_RATE=0, or READ_MODE=1, or READ_MODE=0 and SLAVE_MODE=1.
2 SLAVE_MODE	SAIF Receive Master/Slave Clock Mode Select. For receive operation, this bit selects whether BITCLK and LRCLK are driven to the off-chip codec or uses the two clock pins as inputs to determine when to receive data from the codec. In receive master mode (SLAVE_MODE=0), BITCLK and LRCLK are output to the

HW_SAIF_CTRL field descriptions (continued)

Field	Description
	codec. When SLAVE_MODE=0, both BITCLK and LRCLK start to transition immediately after the RUN bit is set. Note that when in transmit mode or receive master mode, the user must configure the SAIF's clock controls within the clock controller to the correct oversample rate (see BITCLK_BASE_RATE/BITCLK_MULT_RATE above). 0 = Master mode. SAIF drives BITCLK and LRCLK 1 = Slave mode. SAIF uses BITCLK and LRCLK as inputs to determine when to sample input PCM data. Note that is bit is ignored in transmit operation (READ_MODE=0).
1 READ_MODE	SAIF Transmit/Receive Select. This bit selects whether the SAIF block transmits to an off-chip DAC (write mode) or receives from an off-chip ADC (read mode). The selected mode (TX or RX) starts operation once the RUN bit is set. 0 = TX or write mode 1 = RX or read mode
0 RUN	Setting this bit to one causes the SAIF to begin transmitting or receiving serial PCM data, depending on the programming of the READ_MODE bit. For transmit, when this bit is cleared, operation ends after transmission of the current active channel set (pairs from all enabled channels) from the FIFO. If the FIFO is already empty and the RUN bit is cleared, operation halts immediately and the LRCLK and BITCLK pins stop transitioning. For receive, when the RUN bit is cleared, reception ends after the current active channel set (pairs from all enabled channels) are pushed to the FIFO. Note for 4- and 6-channel operation, clearing the RUN bit means that the SAIF does not stop until the corresponding audio samples for all 4 or 6 channels have been transmitted or received.

35.3.2 SAIF Status Register (HW_SAIF_STAT)

The SAIF Status Register provides status of key hardware components required by software of the SAIF module.

HW_SAIF_STAT: 0x010

HW_SAIF_STAT_SET: 0x014

HW_SAIF_STAT_CLR: 0x018

HW_SAIF_STAT_TOG: 0x01C

The SAIF Status Register provides the status of interrupt requests and active operation of the SAIF.

EXAMPLE

```
unsigned TestBit = HW_SAIF_STAT.PRESENT;
```

Address: HW_SAIF_STAT – 8004_2000h base + 10h offset = 8004_2010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	RSRVD2														DMA_ PREQ
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1									FIFO_UNDERFLOW_ IRQ	FIFO_OVERFLOW_ IRQ	FIFO_ SERVICE_IRQ	RSRVD0			BUSY
W										FIFO_UNDERFLOW_ IRQ	FIFO_OVERFLOW_ IRQ					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SAIF_STAT field descriptions

Field	Description
31 PRESENT	This bit is set to 1 in products in which SAIF is present.
30–17 RSRVD2	Reserved.
16 DMA_PREQ	DMA Request Status. This read-only bit reflects the current state of the SAIF's DMA request signal. DMA requests are issued any time the request signal toggles.
15–7 RSRVD1	Reserved.
6 FIFO_ UNDERFLOW_ IRQ	This bit is set by hardware if the FIFO underflows during SAIF operation. Underflow occurs whenever a read or pop is attempted on an empty FIFO. This occurs in transmit mode when the FIFO is not filled in time and the hardware tries to pop a sample from the bottom of the FIFO. It also occurs in receive mode when the DMA or software attempts to read data from an empty FIFO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
5 FIFO_ OVERFLOW_IRQ	This bit is set by hardware if the FIFO overflows during SAIF operation. Overflow occurs whenever a write or push is attempted to a full FIFO. This occurs in transmit mode if the FIFO is full and software or the DMA attempts to write additional data to the top of the FIFO. It also occurs in receive mode when the DMA or software does not respond in time to a service request, and the hardware attempts to push data to a full FIFO. Reset this bit by writing a one to the SCT clear address space and not by a general write.
4 FIFO_SERVICE_ IRQ	This bit is set by hardware when the FIFO requires service. FIFO service requests are made when an empty entry exists during transmit or a full entry exists during receive. A DMA request is generated (DMA_PREQ toggles) each time this bit is set. Once the DMA or software has serviced the request and the FIFO is filled (TX) or emptied (RX), this bit is automatically cleared. This interrupt can be used by software to trigger the manual movement of samples from/to the SAIF's FIFO to/from a memory buffer when the SAIF's DMA channel is not used.

HW_SAIF_STAT field descriptions (continued)

Field	Description
3–1 RSRVD0	Reserved.
0 BUSY	This bit indicates when the SAIF is actively transmitting/receiving serial PCM audio data from/to its FIFO(s). For transmit, it is automatically set when the first sample from the FIFO begins to be output by the serial shifter. For receive, it is set coincident with the RUN bit beging set as serial receive begins immediately. After the RUN bit is cleared and the serial shifter becomes inactive (end of the current sample set), this bit is automatically cleared.

35.3.3 SAIF Data Register (HW_SAIF_DATA)

The SAIF Data Register is used to either write PCM data samples to the top of the SAIF FIFOs for transmit, or read PCM samples from the bottom of the FIFOs during receive. 32-bit values written/read to/from this register contain either one 17-bit to 24-bit sample or two 16-bit samples.

HW_SAIF_DATA: 0x020

HW_SAIF_DATA_SET: 0x024

HW_SAIF_DATA_CLR: 0x028

HW_SAIF_DATA_TOG: 0x02C

In transmit mode, writing a value to this register causes it to push the value to the top of the FIFO. In receive mode, reads cause values to be popped from the bottom of the FIFO. Writing to a full FIFO does not change the contents of the FIFO's top entry, and reading from an empty FIFO returns all zeros. For 16-bit operation, the left sample should be written to the register's lower half word, and the right to the upper half word. For all other data sizes, PCM audio values should be right-justified within the 32-bit word. Three FIFOs exist, one for each channel pair. If two-channel operation is enabled, only the stereo or front FIFO is accessed. Four-channel opearation causes both the front and surround FIFOs to be accessed. Six-channel operation uses all three FIFOs: front, surround, and center/LFE FIFOs. For both transmit and receive FIFO accesses, left and right samples should be interleaved, starting with all left samples for a given sample time, followed by all right samples (e.g., left front, left surround, center, right front, right surround, LFE (subwoofer), and so on). Operation should always start with the left samples. All left or all right channels for a given sample collection in time are received/transmitted simultaneously (e.g., for 6-channel mode, three PCM audio samples are popped from the FIFO prior to serialization and transmission). For mono operation, use the left channel to transmit/receive PCM audio, while the right channel should be zero-filled (TX) or discarded (RX). For transmit, if the FIFO is empty when operation begins, null (zero) data is output until the FIFO contains valid PCM data.

Once a complete set of left samples resides within the bottom of the active FIFOs, the SAIF waits for the current collection of null samples (all left and right samples for a given time) to complete before transmitting the left sample collection from the FIFOs.

EXAMPLE

```
                 HW_SAIF_DATA = 0x12345678; // write 0x1234 to the right channel and 0x5678 to
 the left channel in 16 bit mode
                 HW_SAIF_DATA = 0x12345678; // write 0x12345678 to either the left or right
channel in 32 bit per sample mode.
```

Address: HW_SAIF_DATA – 8004_2000h base + 20h offset = 8004_2020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PCM_RIGHT																PCM_LEFT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SAIF_DATA field descriptions

Field	Description
31–16 PCM_RIGHT	For 16-bit mode, this field contains the entire right channel sample. For 17-bit through 24-bit modes, this field contains 1 through 8 of the MSBs of the sample (either left or right).
15–0 PCM_LEFT	For 16-bit mode, this field contains the entire left channel sample. For 17-bit through 24-bit modes, this field contains the 16 LSBs of the sample (either left or right).

35.3.4 SAIF Version Register (HW_SAIF_VERSION)

The SAIF Version Register is read-only and is used for debug to determine the implementation version number for the block.

EXAMPLE

```
if (HW_SAIF_VERSION.B.MAJOR != 1)
   Error();
```

Address: HW_SAIF_VERSION – 8004_2000h base + 30h offset = 8004_2030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SAIF_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 36
Sony-Philips Digital Interface Format Transmitter (SPDIF)

36.1 Overview

The Sony-Philips Digital Interface Format (SPDIF) transmitter module transmits data according to the SPDIF digital audio interface standard (IEC-60958). Figure 36-1 shows a block diagram of the SPDIF transmitter module.

Data samples are transmitted as blocks of 192 frames, each frame consisting of two 32-bit sub-frames.

A 32-bit sub-frame is composed of a 4-bit preamble, a 24-bit data payload (that is, a left- or right-channel PCM sample), and a 4-bit status field. The status fields are encoded according to the IEC-60958 consumer specification, reflecting the contents of the HW_SPDIF_FRAMECTRL and HW_SPDIF_CTRL registers. See the IEC-60958 specification for proper programming of these fields.

The sub-frame is transmitted serially, LSB-first, using a biphase-mark channel-coding scheme. This encoding allows an SPDIF receiver to recover the embedded clock signal.

Note

Sub-frame information can be changed on-the-fly but is not reflected in the serial stream until the current frame is transmitted. This ensures consistency of the frame and the generated parity appended to that frame.

36.2 Operation

The SPDIF transmitter operates at one of three register-selectable base sample rates: 32 KHz, 44.1 KHz, or 48 KHz. Double-rate output (64 KHz, 88.2 KHz, and 96 KHz) can also be selected using HW_SPDIF_SRR_BASEMULT. The data-clock required to transmit a SPDIF frame at these sample-rates is generated using a fractional clock-divider. This divider uses both edges of a 120 MHz clock, which is derived from a divide-by-4 of the PLL 480

MHz clock. This divider is located in the CLKCTRL module where all the system clocks are generated; the resultant clock (pcm_spdif_clk) is the output to the SPDIF module to be used for data transmission.

Programming the HW_SPDIF_SRR register automatically generates the right frequency of pcm_spdif_clk from the divider in the clock controller block. There are no separate registers to control these dividers. Make sure that the CLKGATE bit in HW_CLKCTRL_SPDIF is cleared before starting the transmission.

The SPDIF module receives data by one of two methods:

- Software-directed PIO writes to the HW_SPDIF_DATA register
- Appropriate programming of the DMA-engine. (See AHB-to-APBX Bridge Overview for a detailed description of the DMA module and how to perform DMA data transfers to/from modules and memory.)

Once provided by the DMA, the received data is placed in a 2x24 word FIFO for each channel, left and right. At initialization, the FIFO is filled before SPDIF data transfer occurs. After this, data is requested whenever this FIFO has an empty entry or at a nominal rate corresponding to the programmed sample- rate in HW_SPDIF_SRR.

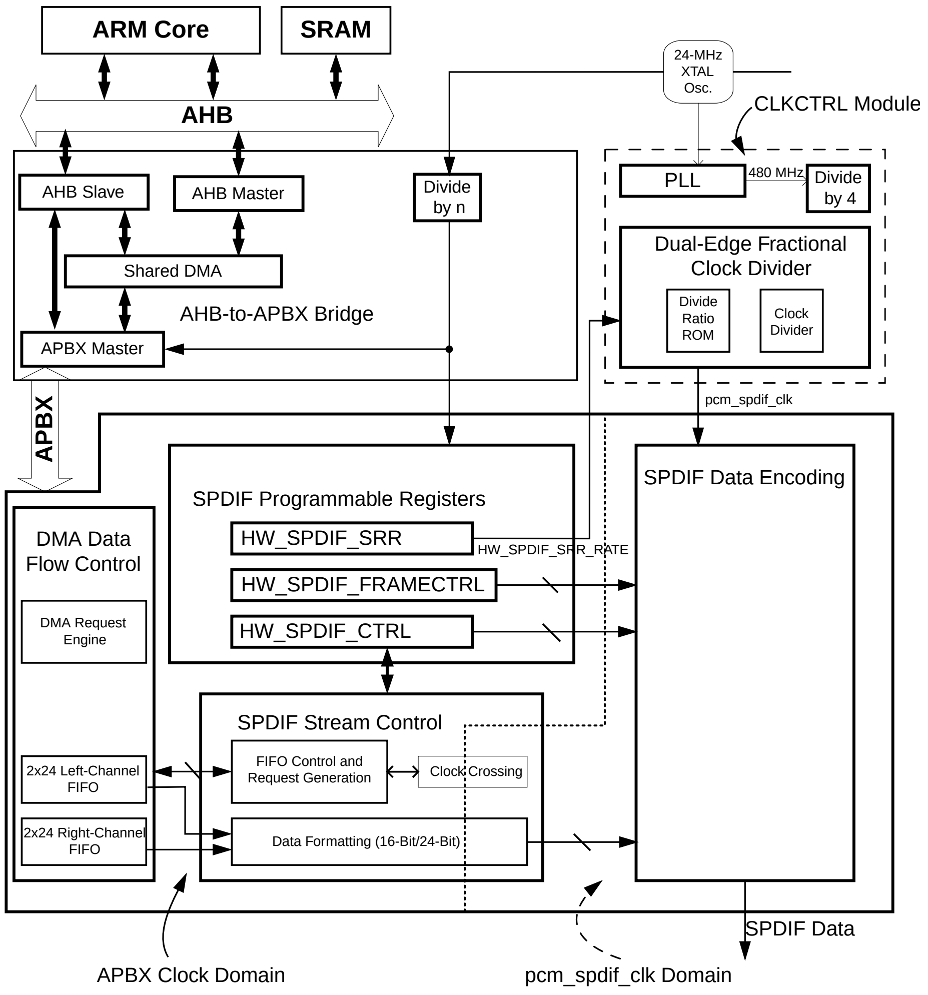


Figure 36-1. SPDIF Transmitter Block Diagram

The behavior of the SPDIF module during or after a FIFO underflow is programmable. On detection of an underflow event, the SPDIF module sends the current sample for four frames before muting (sending zeros) the data stream based on the configuration of HW_SPDIF_FRAMECTRL_AUTO_MUTE. The final validity unit embedded within each frame dictates whether the receiver processes the data within that frame. HW_SPDIF_FRAMECTRL determines the behavior of this bit.

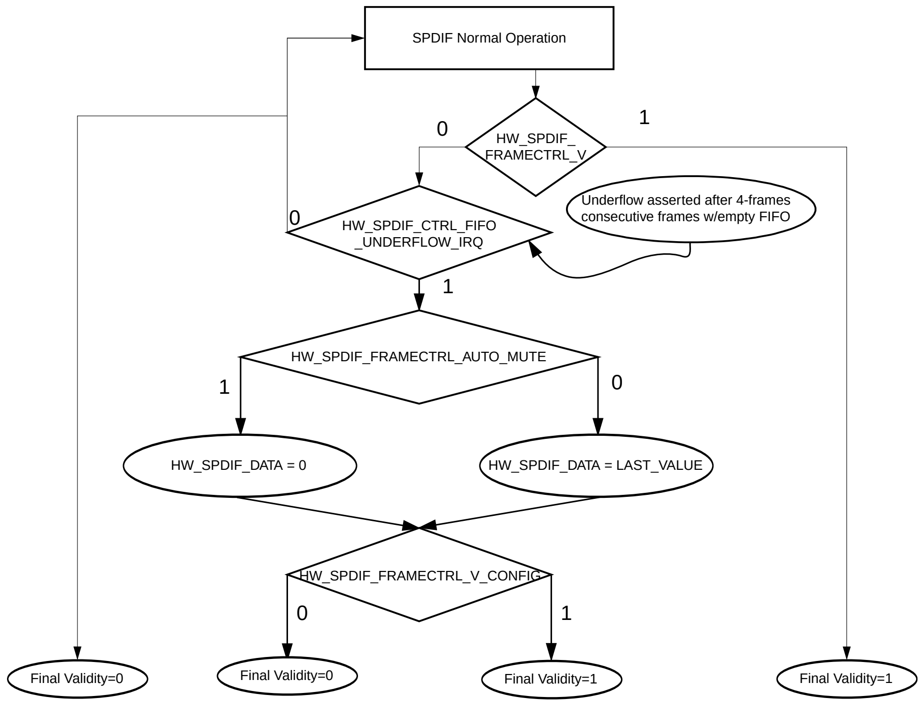

SPDIF Normal Operation
HW_SPDIF_FRAMECTRL_V
0
1
Underflow asserted after 4-frames consecutive frames w/empty FIFO
HW_SPDIF_CTRL_FIFO_UNDERFLOW_IRQ
0
1
HW_SPDIF_FRAMECTRL_AUTO_MUTE
1
0
HW_SPDIF_DATA = 0
HW_SPDIF_DATA = LAST_VALUE
HW_SPDIF_FRAMECTRL_V_CONFIG
0
1
Final Validity=0
Final Validity=0
Final Validity=1
Final Validity=1

Figure 36-2. SPDIF Flow Chart

SPDIF data can be transmitted in one of two modes: 32-bit mode and 16-bit mode. Selection between these modes is done with the WORD_LENGTH bit in the HW_SPDIF_CTR register. In either case, data samples must be interleaved in main memory for proper behavior, although in 32-bit mode, 32-bit words are interleaved and in 16-bit mode, 16-bit words are interleaved.

- When WORD_LENGTH = 0, 32-bit mode is enabled, and HW_SPDIF_DATA contains either the left or right data sample. Since the SPDIF frame allows for transmission of only 24 bits, only the 24 MSBs stored in the HW_SPDIF_DATA register will be transmitted.
- Alternately, when WORD_LENGTH = 1, 16-bit mode is enabled, and the HW_SPDIF_DATA register will contain one of each left AND right samples. The data transmitted in the SPDIF frame will be these 16 MSBs with 8 zeros appended in the LSB positions.

Note

If the data supplied actually represents a lower resolution analog-to-digital conversion, this information is not captured by the SPDIF transmitter, which always reports a 24-bit sample-size.

36.2.1 Interrupts

The SPDIF module contains a single interrupt source that is asserted on FIFO overflows and/or FIFO underflows. This interrupt is enabled by setting HW_SPDIF_CTRL_FIFO_ERROR_IRQ_EN. On interrupt detection, the HW_SPDIF_CTRL_FIFO_UNDERFLOW_IRQ and HW_SPDIF_CTRL_FIFO_OVERFLOW_IRQ fields can be polled for the exact cause of the interrupt and appropriate action taken.

Note

These bits remain valid for polling, regardless of the state of the interrupt enable.

36.2.2 Clocking

The IEC-60958 specification outlines the requirements for SPDIF clocking. The SPDIF module is designed according to the Consumer Audio requirements. These dictate that:

- Average Sample-Rate Error must not exceed 1000 ppm
- Maximum Instantaneous Jitter must not exceed ~4.4 ns.

The jitter requirement implies either a single-phase of a >240 MHz clock or both phases of a 120 MHz clock. It also implies the use of a fractional divider for which the divisors are maintained to sufficient significant digits to yield the required ppm tolerance. The SPDIF module in the i.MX28 uses nine-bit fractional coefficients that yield an average frequency error of 52 ppm. These coefficients are determined according to the required clock-rates that are dictated by the sample rates implemented. The required clock frequencies provided by the CLKCTRL module for the implemented sample-rates are:

F(48 kHz) ≥ 6.144 MHz
F(44.1 kHz) ≥ 5.6448 MHz
F(32 kHz) ≥ 4.096 MHz
F(96 kHz) ≥ 12.288 MHz
F(88.2 kHz) ≥ 11.2896 MHz
F(64 kHz) ≥ 8.192 MHz

All clocks within the SPDIF module are gated according to the state of HW_SPDIF_CTRL_CLKGATE. When set, all clocks derived from the apb_clk are gated. Gating of the pcm_spdif_clk is accomplished through HW_CLKCTRL_SPDIF_CLKGATE.

A module-level reset is also provided in HW_SPDIF_CTRL_SFTRST. Setting this bit performs a module-wide reset and subsequent assertion of the HW_SPDIF_CTRL_CLKGATE.

Note

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

36.2.3 DMA Operation

Using the SPDIF module in DMA mode involves configuring the appropriate DMA channel to provide the interleaved data blocks stored in memory. See AHB-to-APBX Bridge Overview for detailed information on DMA programming. Once programmed, the DMA engine references a set of linked DMA descriptors stored by the CPU in main memory. These descriptors point to data blocks stored in system memory and also provide a mechanism for automated PIO writes before transfer of a data-block. Figure 36-3 describes a typical set of descriptors required to transmit two data-blocks.

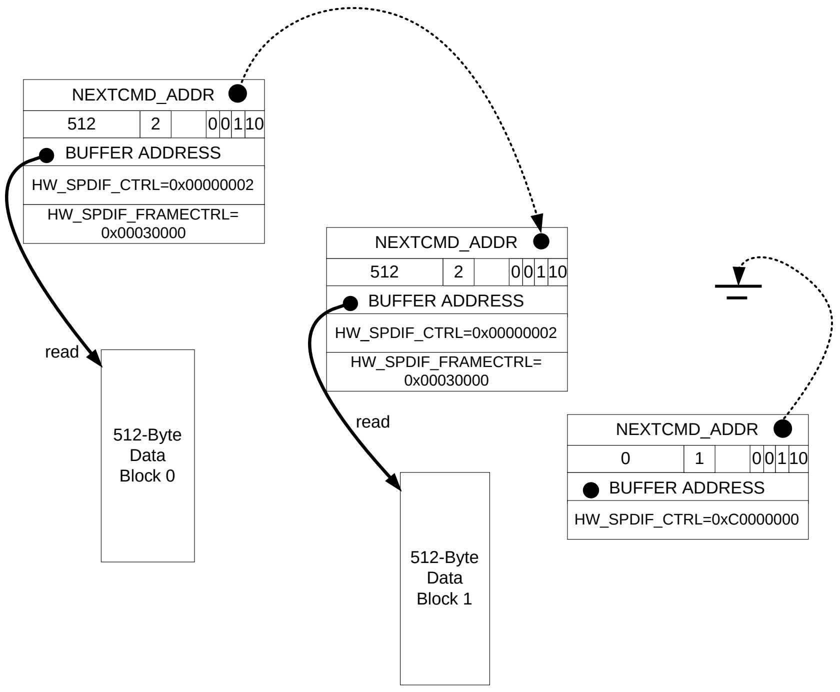


Figure 36-3. SPDIF DMA Two-Block Transmit Example

Here, the DMA is instructed to perform two PIO writes prior to toggling the DMA_PCMDKICK signal:

- HW_SPDIF_CTRL_FIFO_UNDERFLOW_IRQ_EN is set to enable interrupts on FIFO underflow detect
- HW_SPDIF_FRAMECTRL_AUTO_MUTE and HW_SPDIF_FRAMECTRL_V_CONFIG are set to mute and tag the data stream as invalid on a FIFO underflow.

The DMA engine is then programmed to transfer 512-bytes to the SPDIF module.

Additionally, the SPDIF module contains a mechanism for throttling DMA requests to the DMA engine. This circuit is programmed using the HW_SPDIF_CTRL_DMAWAIT_COUNT field and corresponds to the number of cycles of the apb_clk to wait before toggling the DMA_PREQ signal to the DMA engine.

Note

Considering that the bandwidth requirements of the SPDIF module are minimal (not in excess of 96 KHz) and burst requests occur only in pairs, this field can be ignored for most, if not all, applications.

There is a floor APBX frequency below which the SPDIF cannot work without errors. That frequency can be calculated as follows:

- Assume that there are six other blocks apart from SPDIF on the APBX bus, and it takes four APBX clock cycles to service each block. If the number of clock cycles required to service each block changes, change the equations accordingly.
- Assume that HW_SPDIF_CTRL_DMAWAIT_COUNT is less than DMA LATENCY. If this is not true, then even DMA WAIT has to be added to the calculation and the floor APBX frequency increases further.

In 16-bit Mode:

```
Floor APBX freq = (DMA latency + 9) * sample rate.
For max DMA latency = (6 blocks) x (4 cycles per block) = 24 cycles and max SPDIF
sample rate = 96 kHz,
min APBX freq = 3.168 MHz.
```

In 32-bit Mode:

(A) Ideal Calculation:

```
min freq = [2*(DMA latency+4) + 7] * sample rate.
For max DMA latency = 24 cycles and max SPDIF sample rate = 96 kHz,
min APBX freq = 6.048 MHz.
```

(B) Simpler Calculation:

```
Floor APBX freq = 2*(latency + 9) * sample rate = twice that of 16-bit mode.
For max latency = 24 cycles and max sample rate = 96 kHz,
min APBX freq = 6.336 MHz.
```

Option A is ideal as it allows a lower floor frequency; option B can be used to keep it simple and avoid confusion.

36.2.4 PIO Debug Mode

The block is connected only as a PIO device to the APBX bus. Even though it is designed to work with the DMA controller integrated in the APBX bridge, all transfers to and from the block are programmed I/O (PIO) read or write cycles. When the DMA is ready to write to the HW_SPDIF_DATA register, it does so with standard APB write cycles. There *are* four DMA related signals that connect the SPDIF transmitter to the DMA, *but* all data transfers are standard PIO cycles on the APB. The state of these four signals can be seen in the HW_SPDIF_DEBUG register.

Therefore, it is possible to completely exercise the SPDIF block for diagnostic purposes, using only load and store instructions from the CPU without ever starting the DMA controller. This section describes how to interact with the block using PIO operations, and also defines the block's detailed behavior.

Whenever the HW_SPDIF_CTRL register is written to, by either the CPU or the DMA, it establishes the basic operation mode for the block. If the HW_SPDIF_CTRL register is written with a 1 in the RUN bit, then the operation begins and the SPDIF attempts to read the data block by toggling its PDMAREQ signal to the DMA. Notice that the PDMAREQ signal is defined as a toggle signal. This changes state to signify either a request for another DMA word or a notification that the current command transfer has been ended by the SPDIF. Diagnostic software should poll these signals to determine when the SPDIF is ready for another DMA write, and can then supply data by storing a 32-bit word to the HW_SPDIF_DATA register, just as the DMA would perform in a normal operation.

To perform SPDIF transfers in PIO debug mode, diagnostic software should perform the following:

1. Clear CLKGATE in the HW_CLKCTRL_SPDIF register.
2. Turn off the Soft Reset bit, HW_SPDIF_CTRL_SFTRST, and the Clock Gate bit, HW_SPDIF_CTRL_CLKGATE.

3. Properly configure the subcode information by writing the HW_SPDIF_FRAMECTRL register. NOTE: See IEC-60958 for proper coding of these fields.
4. Enable the SPDIF transmitter by setting HW_SPDIF_CTRL_RUN.
5. Wait for HW_SPDIF_DEBUG_DMA_PREQ status bit to toggle.
6. Write one sample of the left/righp t DATA block data to the HW_SPDIF_DATA register.
7. Repeat 5 and 6 until all samples have been written to HW_SPDIF_DATA.

36.3 Programmable Registers

SPDIF Hardware Register Format Summary

HW_SPDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8005_4000	SPDIF Control Register (HW_SPDIF_CTRL)	32	R/W	C000_0020h	36.3.1/2229
8005_4010	SPDIF Status Register (HW_SPDIF_STAT)	32	R	8000_0000h	36.3.2/2231
8005_4020	SPDIF Frame Control Register (HW_SPDIF_FRAMECTRL)	32	R/W	0002_0000h	36.3.3/2232
8005_4030	SPDIF Sample Rate Register (HW_SPDIF_SRR)	32	R/W	1000_0000h	36.3.4/2234
8005_4040	SPDIF Debug Register (HW_SPDIF_DEBUG)	32	R	0000_0001h	36.3.5/2235
8005_4050	SPDIF Write Data Register (HW_SPDIF_DATA)	32	R/W	0000_0000h	36.3.6/2236
8005_4060	SPDIF Version Register (HW_SPDIF_VERSION)	32	R	0101_0000h	36.3.7/2237

36.3.1 SPDIF Control Register (HW_SPDIF_CTRL)

HW_SPDIF_CTRL: 0x000

HW_SPDIF_CTRL_SET: 0x004

HW_SPDIF_CTRL_CLR: 0x008

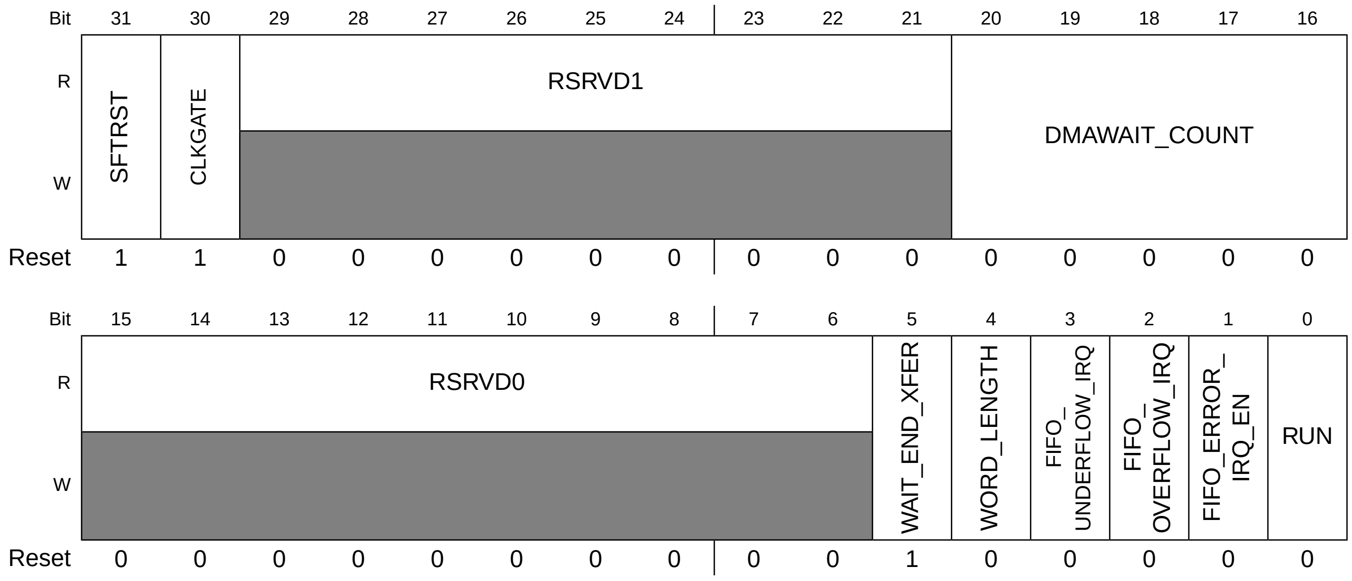
HW_SPDIF_CTRL_TOG: 0x00C

The SPDIF Control Register contains the overall control for SPDIF sample formats, loopback mode, and interrupt controls.

EXAMPLE

```
HW_SPDIF_CTRL.RUN = 1; // start SPDIF conversion
```

Address: HW_SPDIF_CTRL – 8005_4000h base + 0h offset = 8005_4000h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
SFTRST
CLKGATE
RSRVD1
DMAWAIT_COUNT
Reset 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD0
WAIT_END_XFER
WORD_LENGTH
FIFO_ UNDERFLOW_IRQ
FIFO_ OVERFLOW_IRQ
FIFO_ERROR_ IRQ_EN
RUN
Reset 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

HW_SPDIF_CTRL field descriptions

Field	Description
31 SFTRST	Setting this bit to one forces a reset to the entire block and then gates the clocks off. This bit must be set to zero for normal operation.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. First set the CLKGATE bit in the HW_CLKCTRL_SPDIF register to 1. Only then, set this bit to 1 to prevent any extra samples from being transmitted. When removing clock gating, follow the reverse order: First reset this CLKGATE bit to 0, and then reset the CLKGATE bit in the HW_CLKCTRL_SPDIF register to 0 .
29–21 RSRVD1	Reserved
20–16 DMAWAIT_ COUNT	DMA Request Delay Count. This bit field specifies the number of APBX clock cycles (0 to 31) to delay before each DMA request. This field acts as a throttle on the bandwidth consumed by the SPDIF block. This field can be loaded by the DMA.
15–6 RSRVD0	Reserved
5 WAIT_END_ XFER	Set this bit to a one if the SPDIF Transmitter should wait until the internal FIFO is empty before halting transmission based on deassertion of RUN. Use in conjuntion with HW_SPDIF_STAT_END_XFER to determine transfer completion
4 WORD_LENGTH	Set this bit to one to enable 16-bit mode. Set this bit to zero for 32-bit mode. In either case, the SPDIF frame allows transmission of only 24 bits. In 16-bit mode, eight zeros will be appended to the LSB\'s of the input sample; in 32-bit mode, the 24 MSB\'s of HW_SPDIF_DATA will be transmitted.
3 FIFO_ UNDERFLOW_ IRQ	This bit is set by hardware if the FIFO underflows during SPDIF transmission. Reset this bit by writing a one to the SCT clear address space and not by a general write.

HW_SPDIF_CTRL field descriptions (continued)

Field	Description
2 FIFO_ OVERFLOW_IRQ	This bit is set by hardware if the FIFO overflows during SPDIF transmission. Reset this bit by writing a one to the SCT clear address space and not by a general write.
1 FIFO_ERROR_ IRQ_EN	Set this bit to one to enable a SPDIF interrupt request on FIFO overflow or underflow status conditions.
0 RUN	Setting this bit to one causes the SPDIF to begin converting data. The actual conversion will begin when the SPDIF FIFO is filled (4 or 8 words written, depending upon sample word format, i.e., 16 or 32 bits).

36.3.2 SPDIF Status Register (HW_SPDIF_STAT)

HW_SPDIF_STAT: 0x010

HW_SPDIF_STAT_SET: 0x014

HW_SPDIF_STAT_CLR: 0x018

HW_SPDIF_STAT_TOG: 0x01C

The SPDIF Status Register provides the status of the SPDIF converter.

EXAMPLE

```
unsigned TestBit = HW_SPDIF_STAT.PRESENT;
```

Address: HW_SPDIF_STAT – 8005_4000h base + 10h offset = 8005_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PRESENT	RSRVD1[30:16]														
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:1]															END_ XFER
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SPDIF_STAT field descriptions

Field	Description
31 PRESENT	This bit is set to 1 in products in which SPDIF is present.
30–1 RSRVD1	Reserved
0 END_XFER	When set, indicates that the SPDIF module has completed transfer of all data, including data stored in internal FIFOs. Used in conjunction with HW_SPDIF_CTRL_WAIT_END_XFER.

36.3.3 SPDIF Frame Control Register (HW_SPDIF_FRAMECTRL)

HW_SPDIF_FRAMECTRL: 0x020

HW_SPDIF_FRAMECTRL_SET: 0x024

HW_SPDIF_FRAMECTRL_CLR: 0x028

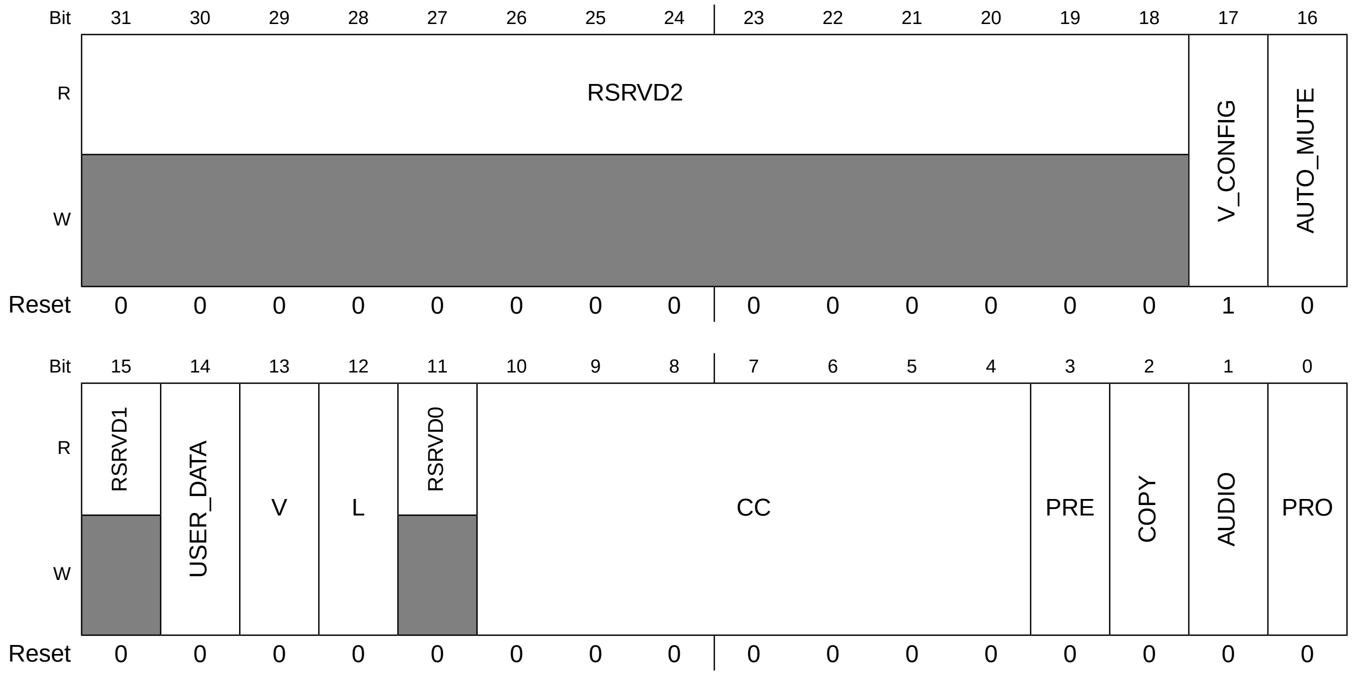
HW_SPDIF_FRAMECTRL_TOG: 0x02C

The SPDIF Frame Control Register provides direct control of the control bits transmitted over an SPDIF frame.

EXAMPLE

```
HW_SPDIF_FRAMECTRL.COPY=1 //SPDIF frame contains copyrighted material
```

Address: HW_SPDIF_FRAMECTRL – 8005_4000h base + 20h offset = 8005_4020h


Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
W
RSRVD2
V_CONFIG
AUTO_MUTE
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
W
RSRVD1
USER_DATA
V
L
RSRVD0
CC
PRE
COPY
AUDIO
PRO
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_SPDIF_FRAMECTRL field descriptions

Field	Description
31–18 RSRVD2	Reserved
17 V_CONFIG	Defines SPDIF behavior when sending invalid frames. 0:Do NOT tag frame as invalid. 1: Tag frame as invalid.
16 AUTO_MUTE	Auto-Mute Stream on stream-suspend detect.
15 RSRVD1	Reserved
14 USER_DATA	User data transmitted during each sub-frame. Consult IEC Standard for additional details.
13 V	Indicates that a sub-frame\'s samples are invalid. If V=0, the sub-frame is indicated as valid, that is, correctly transmitted and received by the interface. If V=1, the subframe is indicated as invalid.
12 L	Generation level is defined by the IEC standard, or as appropriate.
11 RSRVD0	Reserved
10–4 CC	Category code is defined by the IEC standard, or as appropriate.
3 PRE	0: No Pre-Emphasis. 1: Pre-Emphasis is 50/15 usec.
2 COPY	0: Copyright bit NOT asserted. 1: Copyright bit asserted.

HW_SPDIF_FRAMECTRL field descriptions (continued)

Field	Description
1 AUDIO	0:PCM Data;1:Non-PCM Data
0 PRO	0: Consumer use of the channel. 1: Professional use of the channel(Not Support).

36.3.4 SPDIF Sample Rate Register (HW_SPDIF_SRR)

The SPDIF Sample Rate Register controls the sample rate of the data stream played back from the circular buffer.

HW_SPDIF_SRR: 0x030

HW_SPDIF_SRR_SET: 0x034

HW_SPDIF_SRR_CLR: 0x038

HW_SPDIF_SRR_TOG: 0x03C

The SPDIF Sample Rate Register provides a RATE field for specifying the sample rate conversion factor to use in outputting the current SPDIF stream.

EXAMPLE

```
HW_SPDIF_SRR.B.RATE = 0x0AC44; // 44.1KHz
```

Address: HW_SPDIF_SRR – 8005_4000h base + 30h offset = 8005_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1	BASEMULT			RSRVD0								RATE[19:16]			
W	▨				▨	▨	▨	▨	▨	▨	▨	▨				
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RATE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SPDIF_SRR field descriptions

Field	Description
31 RSRVD1	Reserved
30–28 BASEMULT	Base-Rate Multiplier. 1 = Single-Rate (48 kHz). 2 = Double-Rate (96 kHz).
27–20 RSRVD0	Reserved
19–0 RATE	Sample-Rate Conversion Factor. The only valid entries are: 0x07D00, 0x0AC44, 0x0BB80 // 32k, 44.1k, 48k

36.3.5 SPDIF Debug Register (HW_SPDIF_DEBUG)

The SPDIF Debug Register provides read-only access to various internal state information that may be useful for block debugging and validation.

HW_SPDIF_DEBUG: 0x040

HW_SPDIF_DEBUG_SET: 0x044

HW_SPDIF_DEBUG_CLR: 0x048

HW_SPDIF_DEBUG_TOG: 0x04C

This is a read-only register used for checking FIFO status and PIO mode of operation.

EXAMPLE

```
unsigned TestBit = HW_SPDIF_DEBUG.DMA_PREQ;
```

Address: HW_SPDIF_DEBUG – 8005_4000h base + 40h offset = 8005_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:2]														DMA_ PREQ	FIFO_ STATUS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

HW_SPDIF_DEBUG field descriptions

Field	Description
31–2 RSRVD1	Reserved
1 DMA_PREQ	DMA request status. This read-only bit reflects the current state of the SPDIF\'s DMA request signal. DMA requests are issued any time the request signal toggles. This bit can be polled by software, in order to manually move samples to the SPDIF\'s FIFO from a memory buffer when the SPDIF\'s DMA channel is not used
0 FIFO_STATUS	This bit is set when the FIFO has empty space. This reflects a DMA request being generated.

36.3.6 SPDIF Write Data Register (HW_SPDIF_DATA)

The SPDIF Write Data Register receives 32-bit data transfers from the DMA. It deposits the data into an internal FIFO and from there into the SPDIF stream. These 32-bit writes contain either one 32-bit sample or two 16-bit samples.

HW_SPDIF_DATA: 0x050

HW_SPDIF_DATA_SET: 0x054

HW_SPDIF_DATA_CLR: 0x058

HW_SPDIF_DATA_TOG: 0x05C

Writing a 32-bit value to the register corresponds to pushing that 32-bit value into the SPDIF FIFO. The DMA writes 32-bit values to this register. In 32-bit-per-sample mode, the DMA is writing either one full left sample or one full right sample for each write to this register. For 16-bit mode, the DMA is writing a 16-bit left sample and a 16-bit right sample for each 32-bit write to this register.

EXAMPLE

```
                 HW_SPDIF_DATA = 0x12345678; // write 0x1234 to the right channel and 0x5678
to the left channel in 16-bit mode
                 HW_SPDIF_DATA = 0x12345678; // write 0x12345678 to either the left or right
channel in 32-bit per sample mode.
```

Address: HW_SPDIF_DATA – 8005_4000h base + 50h offset = 8005_4050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HIGH																LOW															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SPDIF_DATA field descriptions

Field	Description
31–16 HIGH	For 16-bit mode, this field contains the entire right channel sample. For 32-bit mode, this field contains the 16 MSBs of the 32-bit sample (either left or right).
15–0 LOW	For 16-bit mode, this field contains the entire left channel sample. For 32-bit mode, this field contains the 16 LSBs of the 32-bit sample (either left or right).

36.3.7 SPDIF Version Register (HW_SPDIF_VERSION)

The SPDIF Version Register is read-only and is used for debug to determine the implementation version number for the block.

EXAMPLE

```
if (HW_SPDIF_VERSION.B.MAJOR != 1)
   Error();
```

Address: HW_SPDIF_VERSION – 8005_4000h base + 60h offset = 8005_4060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W																																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_SPDIF_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 37
High-Speed ADC (HSADC)

37.1 Overview

The high-speed ADC block is designed for driving the linear image scanner sensor (for example, TOSHIBA TCD1304DG linear image scanner sensor). It can also support some other general user cases which need to sample analog source with up to 2 Msps data rate and then move the sample data to the external memory. The high-speed ADC block integrates a 12-bit analog ADC block. This analog ADC block can support up to 2 Msps sample rate. In order to improve the flexibility, the high-speed ADC block can co-work with PWM block which can generate driving signals of external device such as the linear image scanner sensor. The PWM can also generate trigger signal which is synchronous with high-speed ADC block to start the conversion of ADC. An APBH-DMA channel is connected to the high-speed ADC block to move the sample data from the asynchronous FIFO inside the high-speed ADC block to the external memory.

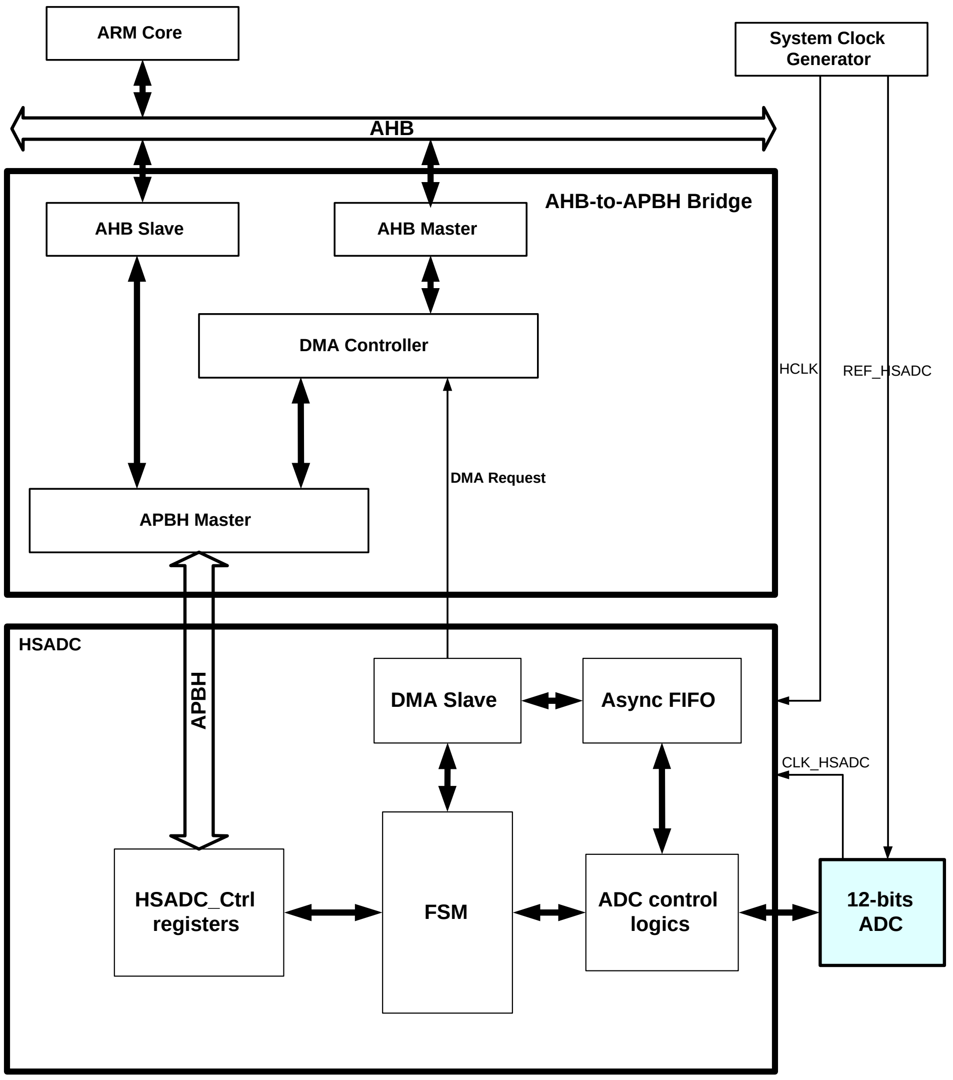


Figure 37-1. Top-Level Block Diagram of High-Speed ADC Block

37.2 High-Speed ADC Block Features

The main features of high-speed ADC block are as following:

37.2.1 Sample Rate and Sample Precision

This block integrates an analog ADC block which can support up to 12-bit sample precision and 2 Msps sample rate. It can also be configured as 10-bit or 8-bit sample precision. The sample rate can be lower than 2 Msps depending on the user cases. For 10-bit mode and 8-bit mode, the user can choose how many bits should be left shifted out before stored. To get a lower sample rate, just need to configure the CLKCTRL block to generate an operation clock with clock frequency of 16x of sample rate.

37.2.2 Trigger Modes

This block can be triggered to start the conversion of analog source by three modes: the first mode is software trigger mode which is to trigger it by ARM CPU to configure a register bit in this block; the second mode is to trigger it by a trigger signal which is generated by the PWM block with big flexibility; the third is to trigger it by an input pin from external sources to support some general user cases.

37.2.3 APBH-DMA Channel

This block is connected to a channel of APBH-DMA to move the sample data from the 16x32 asynchronous FIFO inside the block to the external memory. The APBH-DMA can run at up to 200 MHz clock frequency to reduce the possibility of FIFO overflow. The DMA can also offload the loading of ARM core when the high-speed ADC is working in the loop mode.

37.2.4 Synchronization with PWM Block

This block uses an operation clock the same as the PWM block. It can make it possible that the PWM generate driving signals of external devices and then keep synchronous with high-speed ADC. This will improve the design flexibilities. By selecting the trigger mode of PWM trigger, the high-speed ADC can co-work with the PWM block to support many different user cases.

37.2.5 Clock Domains

This block includes two clock domains. One is the AHB clock domain which is synchronous with the APBH clock. The other is operation clock domain which is the same as the analog operation clock. The AHB clock domain can run at up to 200 MHz clock frequency while the operation clock domain can only run at up to 32 MHz clock frequency. The operation clock needs to be with duty cycle ratio of 25%.

37.2.6 Sample Precision, Endian, Half-word Swap and Bits Left-Shift

The original output of the analog ADC block is 12-bit data. The user can configure the register to get 12-bit,10-bit or 8-bit sample data. For 12-bit or 10-bit modes, two samples are combined to be a 32-bit word. For 8-bit mode, four samples are combined to be a 32-bit word. When using 8-bit mode or 10 bit mode, the user can select any consequential 8 bit or 10 bit sample data from the 12-bit sample data by configuring the register. The sample data are put together to be a 32-bit word before written to the FIFO. The user can also configure the register to get big-endian or little-endian and also half-word-swapped data in the external memory. This operation is done when sample data are read out from the FIFO.

37.3 Operation

The main target of this block is to drive the linear image scanner sensors, especially the TOSHIBA TCD1304DG linear image scanner sensor. To drive this sensor, three driving signals are needed. In order to support other linear image scanner sensors, the driving signals are generated by the PWM block which can improve the design flexibilities. Each PWM instance can be configured as any type of cyclic signal. In order to keep synchronous with the ADC block, the PWM can be configured to use the operation clock of ADC block to generate the driving signals of external devices. When the high-speed ADC is started by the ARM or DMA, it will enter the state to wait for a trigger to start sampling the analog input source. There are three modes to generate this trigger signal. When configured as PWM trigger mode, the PWM can generate the trigger signal with the operation clock to trigger the high-speed ADC to start the sampling. This trigger signal can be generated by any one of the PWM instances and is hard-wire connected to the high-speed ADC block. In order to make it possible for adjusting the sampling point in order to improve the S/N ratio of sample data, some delay cycles of operation clock can be added between the trigger pulse and the time to start sampling. The number of delay cycles can be configured.

There is a 16x32 asynchronous FIFO inside the high-speed ADC with 32-bit width and 16-word depth. The sample data is formed as 32-bit word and then written to the FIFO. When the data is read out, it is reformed to the correct endian and half-word-swapped type. The APBH-DMA will then move the data to the external memory. It is flexible to configure the DMA to put the sample data if there are some requirements about the data format in the external memory, such as line strides, and so on. It is recommended to use one DMA command for each sequence of sample data. So, in loop modes of the high-speed ADC, many sequences of sample data need to be transferred to the external memory which will need a DMA command link to conduct the task. The high-speed ADC will signify the DMA to switch to the next command when one sequence is completed.

For general user cases, the design also provides software trigger mode and external trigger mode. The sample data transferring is also done by the APBH-DMA.

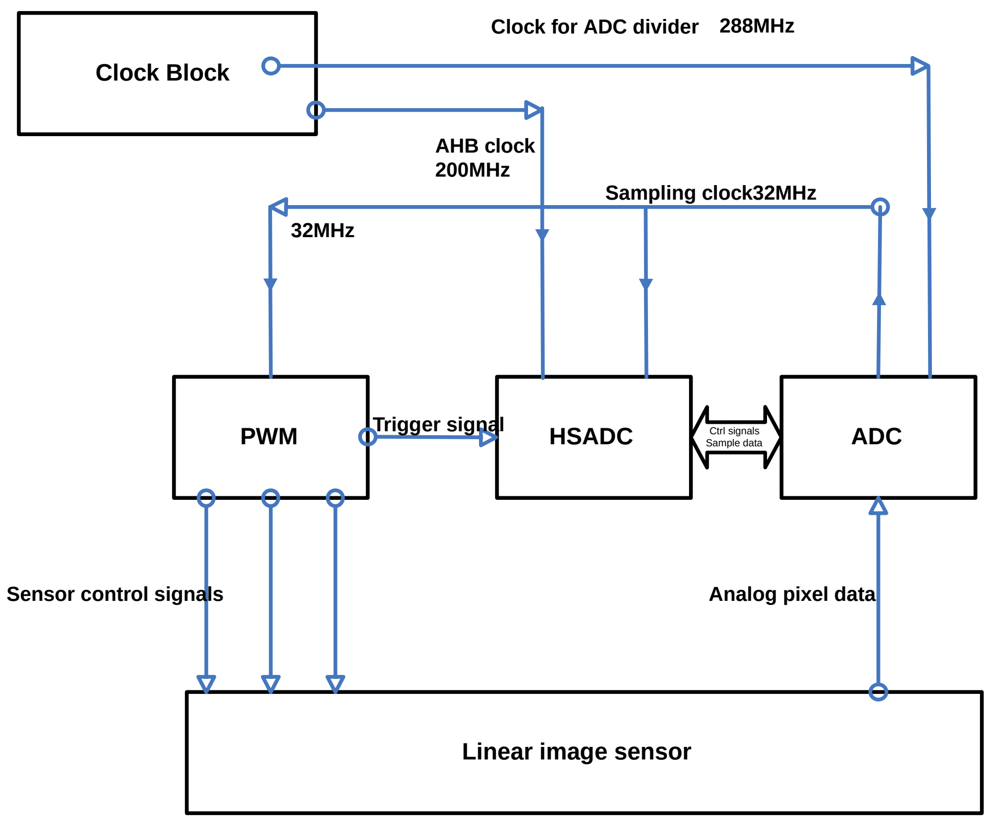


Figure 37-2. Clock paths of the PWM and High-Speed ADC blocks

37.3.1 Trigger Modes

This block can be triggered to start the conversion of analog source by three modes: the first mode is software trigger mode which is to be triggered by ARM to configure a register in this block; the second mode is to be triggered by a trigger pulse which is generated by the PWM block with big flexibility; the third is to be triggered by an input pin from external sources to support some general user cases.

The trigger signal is active high. The duration of the active high pulse needs to be longer than one AHB clock cycle. The trigger signal needs to be de-asserted once the ADC is started. So, it is recommended to assert just one cycle for PWM trigger mode. For external trigger mode, the trigger signal should last for more than one AHB clock cycle. Due to synchronization issue, in all the three trigger modes there will be two AHB clock cycles and two operation clock cycles delay between the trigger pulse and the start of sampling when the delay cycles are configured as 0. Note that for PWM trigger mode, the trigger pulse is generated by operation clock, so the delay cycles should be three operation cycles. Please refer to the Figure 37-3 for the timing of PWM trigger mode and Figure 37-4 for the timing of analog ADC interface.

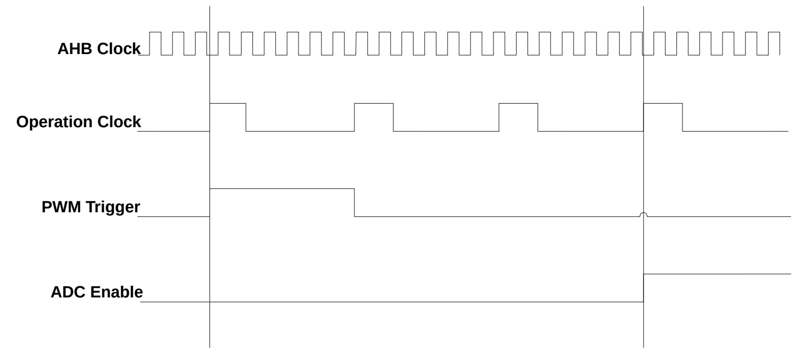


Figure 37-3. Timing of PWM Trigger Mode

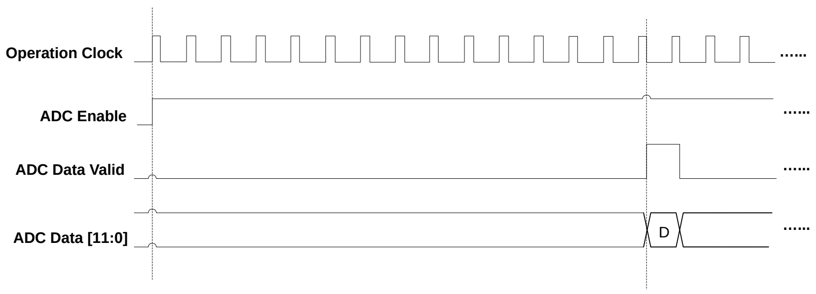


Figure 37-4. Timing of analog ADC interface

37.3.2 Sample Data Bits Left Shift

The original output of the analog ADC block is 12-bit data. When using 8-bit mode or 10-bit mode, the user can configure the ADC_SAMPLE_SHIFT_BITS_NUM in HSADC Control Register 0 to determine the number of bits required to be left-shifted out. So, the user can choose any 8 bit or 10 bit consequential sample data to be stored into the external memory. Note that when in 8-bit mode, the number of left-shifted bits can be from zero to four while in 10-bit mode, the number can only be from zero to two. This feature can be useful to cover all the amplitude range of the analog input source.

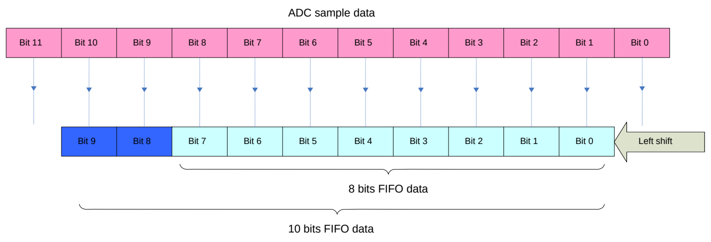


Figure 37-5. Left Shift Bits of Sample Data

37.3.3 Bits Location

The original output sample data is 12-bits. The user can configure the ADC_SAMPLE_PRECISION bits in HSADC Control Register 0 to determine the bits used for one sample. When 8-bits mode is selected, four sample data will be put in one word as Figure 37-6 indicate. When 10-bits mode is selected, two sample data will be put in one word as Figure 37-7 indicate. When 12-bits mode is selected, two sample data will be put in one word as Figure 37-8 indicate. In 10-bits mode and 12-bits mode, the bit31-bit29 and bit15-bit13 are used for storing the channel number from which this sample is captured. Then the software can sort the data in external memory and store to different memory space by the channel number. This feature is useful when there are channels switching for capturing the samples from many channels.

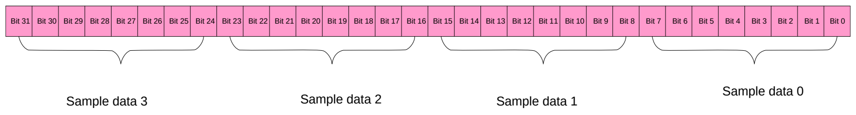


Figure 37-6. Bits Location Of Sample Data In 8-bits Mode

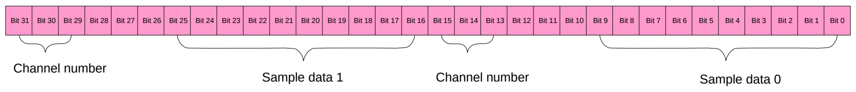


Figure 37-7. Bits Location Of Sample Data In 10-bits Mode

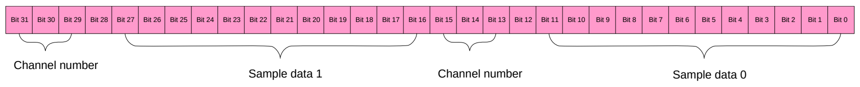


Figure 37-8. bits location of sample data in 12-bits mode

37.3.4 Configuration of APBH-DMA

There is one APBH-DMA channel connected to HSADC. Once there is sample data written into the asynchronous FIFO inside the high-speed ADC block, the high-speed ADC block will assert the dma_req, then the DMA will read the sample data through the APBH bus and then write to the external memory. When the user case is to drive a linear image scanner sensor, it is suggested to use one DMA command to transfer one sequence of sample data. And when one sequence is finished, the high-speed ADC block will assert dma_end so that the DMA can switch to the next command. The dma_run can also start the high-speed ADC block. So when dma_run is asserted, the high-speed ADC block will enter into the status to wait for the trigger pulse.

37.3.5 Configuration of PWM

There is a hard-wired connection between PWM block and high-speed ADC block. This wire is for PWM block to trigger the high-speed ADC to start the sampling. When the user case is to drive a linear image scanner sensor, it is suggested to generate the sensor driving signals by the operation clock, and use another PWM instance to generate the trigger signal. It is required that the trigger pulse is active high and the duration is longer than one operation clock cycle. The user can control time of the trigger pulse to control the sampling point of ADC.

37.3.6 Configuration of High-speed ADC

There are two working mode which can be chosen, single mode and loop mode. For loop mode, the high-speed ADC block enters into the status to wait for another trigger pulse once the current sequence is finished. The number of samples and sequences can be configured. There are up to eight channels, the user can choose one of the eight channels by configuring the register. Also, the delay cycles between the trigger pulse and the time to start sampling can be controlled by configuring register. Note that there are also two AHB clock cycles and two operation clock cycles delay that are required to be added. The sample precision, endian and whether to do half-word-swap can also be configured. There are three trigger modes which can be selected as mentioned above. When hsadc_run bit is set, all these settings take effect. The user can change these parameters when the ADC is working on the current sequence and will take effect when hsadc_run bit is set again. When DMA get the dma_req toggled, it will read out the sample data by APBH bus. It is also possible that the ARM CPU reads the sample data through the APBH bus in some special user cases. When one sequence is finished, all the sample data will be flushed out to the external memory. If it is not an integer number of word, the unfilled bits will be left empty. There are some read-only registers which contain necessary information for debugging, such as the current state, the state machines inside the block, and sample number captured and sequence number finished, and so on. If DCDC converter works to provide supply voltage, in order to attenuate the impact of DCDC noise on HSADC performance, it is recommended to set the sampling rate of HSADC to 1.5 Msps and set the register HW_POWER_ANACLKCTRL(0x80044160) as 0x84000626. But, there would be some initial delay time (less than 0.67 us) between HSADC starting conversion and 'start' command coming from software trigger or PWM trigger. If the initial delay time is not desired, it is recommended to set the register HW_POWER_ANACLKCTRL(0x80044160) as 0x84000426 and use PWM trigger mode.

37.3.7 Interrupt Sources

There are two ways for ARM CPU to talk with the high-speed ADC block. One is by polling the interrupt bit in HSADC Control Register 1. The other is by interrupt. The ARM can choose one of these two modes by configure register. The user can enable the interrupt so that when one sequence is done, the interrupt is asserted. and when the asynchronous FIFO overflow occurs, the FIFO_OVERFLOW interrupt is asserted. Normally, the FIFO overflow will seldom occur because the APBH-DMA is working on 200 MHz clock frequency and the maximum sampling rate is 2 Msps for 12-bit sample data which means that the maximum data rate is 1 MHz word. So, the bandwidth of DMA should be enough to avoid the FIFO overflow. When overflow occurs, the FIFO will discard 16 words of sample data and then continues working. There is TIMEOUT interrupt which will be asserted when the block is pending. When the ARM CPU enters into interrupt mode, it should first clear the interrupt by setting INTERRUPT_CLR bit. Then, check the HSADC Control Register 1 for the interrupt status. Then, set INTERRUPT_STATUS_CLR bit to clear the interrupt status bits.

37.3.8 Working Modes

There are two working modes for HSADC. When the value set to HSADC Sequence Number Register is 1, the HSADC is working in single mode; When the value set to HSADC Sequence Number Register is not 1, the HSADC is working in loop mode.

37.3.8.1 Single Mode

For single mode, the HSADC just captures one sequence of sample data. The number of sample data need to be captured can be controlled by configuring the HSADC Sequence Samples Number Register. Note that when set the value of HSADC Sequence Samples Number Register as 0, it means endless capturing of samples till the disassertion of HSADC_RUN bit in HSADC Control Register 0.

37.3.8.2 Loop Mode

For loop mode, the HSADC will capture more than one sequence of sample data. The number of samples for each sequence can be configured just the same as single mode. When one sequence is finished, the HSADC will automatically enter the state to wait for the next trigger pulse to start the next sequence of sampling. The number of sequences can be configured in HSADC Sequence Number Register. Please note that when set the value of HSADC Sequence Number Register as 0, it means endless capture of sequences till the disassertion of HSADC_RUN bit in HSADC Control Register 0.

37.3.9 Debugging Information

There are three read-only registers providing some useful information for debugging. HW_HSADC_DBG_INFO0 provides HSADC_FSM_STATE , DMA_FSM_STATE and DMA_REQ. The HSADC_FSM_STATE and DMA_FSM_STATE provides the state of the FSMs inside the block, and the FIFO_READ_EMPTY can be used by ARM CPU for polling the status of the FIFO. 1 means there is no data in the FIFO, 0 means there is data in the FIFO. So, for some special user cases the ARM CPU can also read the data from the asynchronous FIFO once there is data ready.

The HW_HSADC_DBG_INFO1 provides the sample count number of the current sequence already finished. The HW_HSADC_DBG_INFO2 provides sequence count number already finished.

37.3.10 Behavior During Reset

A soft reset (SFTRST) can take multiple clock cycles to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

37.4 Programming Example

The main target of this block is to drive the linear image scanner sensors, especially the TOSHIBA TCD1304DG linear image scanner sensor. Let's take TOSHIBA TCD1304DG linear image scanner sensor for example. Below is the steps to configure the APBH-DMA,PWM and HSADC blocks:

1. Configure the clock block to output clock to the analog ADC block. Then the analog ADC block will generate operation clock for the HSADC block and PWM block. The frequency of the clock output to analog ADC block is 288MHz. Then the divider in ADC block will divide the clock to 16x of the sample rate needed. For example, When 2Msps needed, the clock frequency output by the divider should be 32MHz.
2. Clear the clock gating bits and soft reset bits for HSADC, PWM and APBH-DMA block.
3. Configure the HSADC block.Clear the POWER_DOWN bit in HSADC Control Register 2. Assert the ADC_PRECHARGE bit in HSADC Control Register 2 so that the analog ADC block can enter into a stable status and be ready for conversion. Once all the parameters are set, set the hsadc_run bit in HSADC Control Register 0. Then the HSADC block is in the state to wait the trigger pulse to start the conversion.

4. Configure the registers of APBH-DMA, write commands to external memory.
5. Configure the PWM block. The sensor needs three driving signals. Select three PWM instances to generate the three driving signals with the operation clock. Select another PWM instance to generate the trigger signals. Then start all these four PWM instances at the same time. Then the trigger pulse generated by PWM block will start the conversion of HSADC.
6. The ARM CPU keeps polling the interrupt bit of HSADC or waits for the interrupt of HSADC.

For general purpose, the steps are almost the same as above. The difference is the trigger modes. The HSADC will start the conversion once triggered.

37.5 Programmable Registers

HSADC Hardware Register Format Summary

HW_HSADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8000_2000	HSADC Control Register 0 (HW_HSADC_CTRL0)	32	R/W	C000_0040h	37.5.1/2249
8000_2010	HSADC Control Register 1 (HW_HSADC_CTRL1)	32	R/W	F000_0020h	37.5.2/2252
8000_2020	HSADC Control Register 2 (HW_HSADC_CTRL2)	32	R/W	0000_238Eh	37.5.3/2254
8000_2030	HSADC Sequence Samples Number Register (HW_HSADC_SEQUENCE_SAMPLES_NUM)	32	R/W	0000_0000h	37.5.4/2256
8000_2040	HSADC Sequence Number Register (HW_HSADC_SEQUENCE_NUM)	32	R/W	0000_0000h	37.5.5/2256
8000_2050	HSADC FIFO Data Register (HW_HSADC_FIFO_DATA)	32	R	0000_0000h	37.5.6/2257
8000_2060	HSADC Debug Information 0 Register (HW_HSADC_DBG_INFO0)	32	R	0000_0000h	37.5.7/2258
8000_2070	HSADC Debug Information 1 Register (HW_HSADC_DBG_INFO1)	32	R	0000_0000h	37.5.8/2259
8000_2080	HSADC Debug Information 2 Register (HW_HSADC_DBG_INFO2)	32	R	0000_0000h	37.5.9/2259
8000_20B0	HSADC Version Register (HW_HSADC_VERSION)	32	R	0001_0000h	37.5.10/2260

37.5.1 HSADC Control Register 0 (HW_HSADC_CTRL0)

The HSADC Control and Status Register specifies the reset state, trigger sources, and software enable.

HW_HSADC_CTRL0: 0x000

HW_HSADC_CTRL0_SET: 0x004

HW_HSADC_CTRL0_CLR: 0x008

HW_HSADC_CTRL0_TOG: 0x00C

The HSADC Control and Status Register specifies the reset state, trigger sources, and software enable,etc.

EXAMPLE

```
HW_HSADC_CTRL0_WR(0x0000001f);
```

Address: HW_HSADC_CTRL0 – 8000_2000h base + 0h offset = 8000_2000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	TRIGGER_ SOURCE		SOFTWARE_TRIGGER	RSRVD1						DISCARD		ADC_ SAMPLE_ PRECISION		ADC_SAMPLE_ ENDIAN
W						▨										
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ADC_SAMPLE_ HALFWORD_SWAP	ADC_SAMPLE_ SHIFT_BITS_NUM			RSRVD0					HSADC_ PRESENT	TRIGGER_DELAY_CYCLES					HSADC_RUN
W					▨					▨						
Reset	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0

HW_HSADC_CTRL0 field descriptions

Field	Description
31 SFTRST	This bit must be cleared to 0 for normal operation. When set to 1, it forces a block-wide reset.
30 CLKGATE	This bit must be cleared to 0 for normal operation. When set to 1, it gates off the clocks to the block.
29–28 TRIGGER_ SOURCE	This bit field specifies the trigger mode of ADC.Take effect when hsadc_run is asserted. 00: ADC is triggered by the software. 01: ADC is triggered by the PWM trigger signal. 10: ADC is triggered by the input signal from outside the chip.

HW_HSADC_CTRL0 field descriptions (continued)

Field	Description
	11: no trigger.
27 SOFTWARE_ TRIGGER	When set to 1'b1,trigger the ADC to start the conversion.This bit can be auto cleared.
26–21 RSRVD1	Reserved.
20–19 DISCARD	This bit field specifies the number of samples to be discarded whenever the analog ADC is firstly powered up.Take effect when hsadc_run is asserted. 00= discard first sample 01= discard first 2 samples 10= discard first 3 samples 11= discard first 4 samples 1_SAMPLE = 0x0 discard first sample before first capture. 2_SAMPLES = 0x1 discard 2 samples before first capture. 3_SAMPLES = 0x2 discard 3 samples before first capture. 4_SAMPLES = 0x3 discard 4 samples before first capture.
18–17 ADC_SAMPLE_ PRECISION	This bit field specifies the precision of sample data.Take effect when hsadc_run is asserted. 00= When set to 2'b00,the HSADC output 8-bits sample data. 01= When set to 2'b01,the HSADC output 10-bits sample data. 10= When set to 2'b10,the HSADC output 12-bits sample data. 11= When set to 2'b11,no data output.
16 ADC_SAMPLE_ ENDIAN	This bit field specifies the endian of sample data.Take effect when hsadc_run is asserted. 0= When set to 1'b0,the HSADC outputs little endian sample data. 1= When set to 1'b1,the HSADC outputs big endian sample data.
15 ADC_SAMPLE_ HALFWORD_ SWAP	This bit field specifies whether to do halfword swap on the sample data.Take effect when hsadc_run is asserted. 0= When set to 1'b0,the HSADC don't swap the output sample data. 1= When set to 1'b1,the HSADC do 16-bits swap on the output sample data.
14–12 ADC_SAMPLE_ SHIFT_BITS_ NUM	This bit field specifies the bits number of sample data to be left shifted.Take effect when hsadc_run is asserted. 000= When set to 3'b000,the sample data remains unchanged. 001= When set to 3'b001,the sample data be left shifted 1 bit. 010= When set to 3'b010,the sample data be left shifted 2 bit. 011= When set to 3'b011,the sample data be left shifted 3 bit. 100= When set to 3'b100,the sample data be left shifted 4 bit. When set to other value,the sample data remains unchanged. For 8-bits mode,all these setting are valid.For 10-bits,only 3'b000,3'b001,3'b010 are valid.

HW_HSADC_CTRL0 field descriptions (continued)

Field	Description
11–7 RSRVD0	Reserved.
6 HSADC_ PRESENT	This read-only bit returns 1 when the HSADC function block is present on the chip.
5–1 TRIGGER_ DELAY_CYCLES	Set the cycles between the triggers happen and the ADC starts the conversion.Take effect when hsadc_run is asserted.
0 HSADC_RUN	When set to 1'b1,the HSADC loads the parameters in the registers and starts to run.

37.5.2 HSADC Control Register 1 (HW_HSADC_CTRL1)

The HSADC Control Register 1 specifies interrupt status.

HW_HSADC_CTRL1: 0x010

HW_HSADC_CTRL1_SET: 0x014

HW_HSADC_CTRL1_CLR: 0x018

HW_HSADC_CTRL1_TOG: 0x01C

This register specifies interrupt status of HSADC.

EXAMPLE

```
HW_HSADC_CTRL1_WR(0x0000001f);
```

Address: HW_HSADC_CTRL1 – 8000_2000h base + 10h offset = 8000_2010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INTERRUPT_END_ONE_SEQUENCE_ENABLE	INTERRUPT_ADC_DONE_ENABLE	INTERRUPT_FIFO_OVERFLOW_ENABLE	INTERRUPT_TIMEOUT_ENABLE	INTERRUPT_CLR	INTERRUPT_STATUS_CLR	RSRVD1[25:16]									
W							▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:6]										FIFO_READ_ EMPTY	INTERRUPT_END_ ONE_SEQUENCE_ STATUS	INTERRUPT_ADC_ DONE_STATUS	INTERRUPT_FIFO_ OVERFLOW_ STATUS	INTERRUPT_ TIMEOUT_ STATUS	INTERRUPT
W	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

HW_HSADC_CTRL1 field descriptions

Field	Description
31 INTERRUPT_ END_ONE_ SEQUENCE_ ENABLE	When set to 1'b1,the HSADC asserts interrupt when one sequence is finished.When set to 1'b0,the HSADC does not assert interrupt when one sequence is finished.
30 INTERRUPT_ ADC_DONE_ ENABLE	When set to 1'b1,the HSADC asserts interrupt when all sequences are finished.When set to 1'b0,the HSADC does not assert interrupt when all sequences are finished.
29 INTERRUPT_ FIFO_ OVERFLOW_ ENABLE	When set to 1'b1,the HSADC asserts interrupt when FIFO overflow occurs.When set to 1'b0,the HSADC does not assert interrupt when FIFO overflow occurs.
28 INTERRUPT_ TIMEOUT_ ENABLE	When set to 1'b1,the HSADC asserts interrupt when timeout occurs.When set to 1'b0,the HSADC does not assert interrupt when timeout occurs.
27 INTERRUPT_ CLR	When set to 1'b1,clear the HSADC interrupt.This bit can be auto cleared.
26 INTERRUPT_ STATUS_CLR	When set to 1'b1,clear all the HSADC interrupt status.This bit can be auto cleared.
25–6 RSRVD1	Reserved.
5 FIFO_READ_ EMPTY	FIFO read empty. 1 means there is no data in the FIFO, 0 means there is data in the FIFO.
4 INTERRUPT_ END_ONE_ SEQUENCE_ STATUS	This bit is set to one upon one sequence is finished.It is ANDed with its corresponding interrupt enable bit to request an interrrupt.Can be cleared by INTERRUPT_STATUS_CLR bit.

HW_HSADC_CTRL1 field descriptions (continued)

Field	Description
3 INTERRUPT_ ADC_DONE_ STATUS	This bit is set to one upon all the sequences are finished.It is ANDed with its corresponding interrupt enable bit to request an interrrupt.Can be cleared by INTERRUPT_STATUS_CLR bit.
2 INTERRUPT_ FIFO_ OVERFLOW_ STATUS	This bit is set to one upon FIFO overflow occurred.It is ANDed with its corresponding interrupt enable bit to request an interrupt.Can be cleared by INTERRUPT_STATUS_CLR bit.
1 INTERRUPT_ TIMEOUT_ STATUS	This bit is set to one upon timeout occur.It is ANDed with its corresponding interrupt enable bit to request an interrupt.Can be cleared by INTERRUPT_STATUS_CLR bit.
0 INTERRUPT	Set to 1 when an interrupt is raised. Can be cleared by the INTERRUPT_CLR bit.The host CPU can poll this bit for the polling mode.

37.5.3 HSADC Control Register 2 (HW_HSADC_CTRL2)

The HSADC Control Register 2 specifies analog ADC config bits.

HW_HSADC_CTRL2: 0x020

HW_HSADC_CTRL2_SET: 0x024

HW_HSADC_CTRL2_CLR: 0x028

HW_HSADC_CTRL2_TOG: 0x02C

This register specifies the config bits for analog ADC block.

EXAMPLE

```
HW_HSADC_CTRL2_WR(0x0000001f);
```

Address: HW_HSADC_CTRL2 – 8000_2000h base + 20h offset = 8000_2020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1 [15:14]		POWER_DOWN	SAH_GAIN_ADJ			DAC_ADJHEADROOM	DAC_ADJCURRENT	SAH_BIAS_ADJ		ONCHIP_GROUND	ADC_SH_BYPASS	ADC_CHANNEL_SEL			ADC_PRECHARGE
W	▨	▨														
Reset	0	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0

HW_HSADC_CTRL2 field descriptions

Field	Description
31–14 RSRVD1	Reserved.
13 POWER_DOWN	Power down the analog ADC block.
12–10 SAH_GAIN_ADJ	Control the gain of sample and hold module in HSADC.
9 DAC_ ADJHEADROOM	Adjust the headroom of current sources of sub-DAC.
8 DAC_ ADJCURRENT	Adjust the current of sub-DAC.
7–6 SAH_BIAS_ADJ	Adjusting the settling time of the sample-ana-hold circuit in front of the hsadc.
5 ONCHIP_ GROUND	Switch the ground of sub-DAC between quiet ground and onchip ground.
4 ADC_SH_ BYPASS	ADC sample and hold logics bypass.
3–1 ADC_CHANNEL_ SEL	This bit field specifies the pin names of the analog source to be converted. 000= When set to 3'b000,the HSADC select the LRADC0 pin as analog source input. 001= When set to 3'b001,the HSADC select the LRADC1 pin as analog source input. 010= When set to 3'b010,the HSADC select the LRADC2 pin as analog source input. 011= When set to 3'b011,the HSADC select the LRADC3 pin as analog source input. 100= When set to 3'b100,the HSADC select the LRADC4 pin as analog source input. 101= When set to 3'b101,the HSADC select the LRADC5 pin as analog source input. 110= When set to 3'b110,the HSADC select the LRADC6 pin as analog source input. 111= When set to 3'b111,the HSADC select the HSADC0 pin as analog source input.
0 ADC_ PRECHARGE	ADC precharge enable.Should be set before start ADC conversion.

37.5.4 HSADC Sequence Samples Number Register (HW_HSADC_SEQUENCE_SAMPLES_NUM)

The HSADC Sequence Samples Number Register specifies the number of samples in one sequence.

HW_HSADC_SEQUENCE_SAMPLES_NUM: 0x030

HW_HSADC_SEQUENCE_SAMPLES_NUM_SET: 0x034

HW_HSADC_SEQUENCE_SAMPLES_NUM_CLR: 0x038

HW_HSADC_SEQUENCE_SAMPLES_NUM_TOG: 0x03C

This register specifies the number of samples in one sequence.

EXAMPLE

```
HW_HSADC_SEQUENCE_SAMPLES_NUM_WR(0x0000001f);
```

Address: HW_HSADC_SEQUENCE_SAMPLES_NUM – 8000_2000h base + 30h offset = 8000_2030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SEQUENCE_SAMPLES_NUM																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_SEQUENCE_SAMPLES_NUM field descriptions

Field	Description
31–0 SEQUENCE_ SAMPLES_NUM	Number of samples in one sequence. 0 means infinite samples. Note that when set to 0,the ADC can only be stopped by software disabling.Take effect when hsadc_run is asserted.

37.5.5 HSADC Sequence Number Register (HW_HSADC_SEQUENCE_NUM)

The HSADC Sequence Number Register specifies the number of sequence needed to be converted.

HW_HSADC_SEQUENCE_NUM: 0x040

HW_HSADC_SEQUENCE_NUM_SET: 0x044

HW_HSADC_SEQUENCE_NUM_CLR: 0x048

HW_HSADC_SEQUENCE_NUM_TOG: 0x04C

This register specifies the number of sequences.

EXAMPLE

```
HW_HSADC_SEQUENCE_NUM_WR(0x0000001f);
```

Address: HW_HSADC_SEQUENCE_NUM – 8000_2000h base + 40h offset = 8000_2040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SEQUENCE_NUM																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_SEQUENCE_NUM field descriptions

Field	Description
31–0 SEQUENCE_ NUM	Number of sequence. 0 means infinite sequences. Note that when set to 0,the ADC can only be stopped by software disabling.Take effect when hsadc_run is asserted.

37.5.6 HSADC FIFO Data Register (HW_HSADC_FIFO_DATA)

The HSADC FIFO Data Register is for the DMA or host CPU to read data from the FIFO by APB bus.

This register contains the programming paramters for multi-chip attachment mode, clock divider value, active high, low values and period for channel 1.

EXAMPLE

```
HW_HSADC_FIFO_DATA_RD();
```

Address: HW_HSADC_FIFO_DATA – 8000_2000h base + 50h offset = 8000_2050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIFO_DATA																															
W	▨																															
Re-set	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_FIFO_DATA field descriptions

Field	Description
31–0 FIFO_DATA	The FIFO data interface for DMA or host CPU to read data from the FIFO internal HSADC module.

37.5.7 HSADC Debug Information 0 Register (HW_HSADC_DBG_INFO0)

The HSADC Debug Information Register provides some useful information for debugging.

This register contains the active time and inactive time programming parameters for Channel 2.

EXAMPLE

```
HW_HSADC_DBG_INFO0_RD();
```

Address: HW_HSADC_DBG_INFO0 – 8000_2000h base + 60h offset = 8000_2060h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1[31:16]															
W	▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1[15:6]										DMA_FSM_STATE			HSADC_FSM_STATE		
W	▨										▨			▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_DBG_INFO0 field descriptions

Field	Description
31–6 RSRVD1	Reserved.
5–3 DMA_FSM_STATE	The current state of dma slave state machine.
2–0 HSADC_FSM_STATE	The current state of hsadc state machine.

37.5.8 HSADC Debug Information 1 Register (HW_HSADC_DBG_INFO1)

The HSADC Debug Information Register provides some useful information for debugging.

This register contains the number of samples which are already finished in the current sequence.

EXAMPLE

```
HW_HSADC_DBG_INFO1_RD();
```

Address: HW_HSADC_DBG_INFO1 – 8000_2000h base + 70h offset = 8000_2070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SEQUENCE_SAMPLE_CNT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_DBG_INFO1 field descriptions

Field	Description
31–0 SEQUENCE_ SAMPLE_CNT	The number of samples which are already finished in the current sequence.

37.5.9 HSADC Debug Information 2 Register (HW_HSADC_DBG_INFO2)

The HSADC Debug Information Register provides some useful information for debugging.

This register contains the number of sequences which are already finished.

EXAMPLE

```
HW_HSADC_DBG_INFO2_RD();
```

Address: HW_HSADC_DBG_INFO2 – 8000_2000h base + 80h offset = 8000_2080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SEQUENCE_CNT																															
W	▨																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_DBG_INFO2 field descriptions

Field	Description
31–0 SEQUENCE_ CNT	The number of sequences which are already finished.

37.5.10 HSADC Version Register (HW_HSADC_VERSION)

This register indicates the version of the block for debug purposes.

This register indicates the RTL version in use.

EXAMPLE

```
if (HW_HSADC_VERSION.B.MAJOR != 1) Error();
```

Address: HW_HSADC_VERSION – 8000_2000h base + B0h offset = 8000_20B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_HSADC_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 38
Low-Resolution ADC (LRADC) and Touch-Screen Interface

38.1 LRADC Overview

This chapter describes the low-resolution analog-to-digital converters (LRADC) and touch-screen interface. It includes sections on scheduling conversions, delay channels and programmable registers.

The sixteen-channel low-resolution analog-to-digital converter block is used for voltage measurement. Figure 38-1 shows a block diagram of the LRADC. Eight virtual channels can be used at one time. Each of the eight virtual channels can be mapped to any of the 16 physical channels using HW_LRADC_CTRL4. Six physical channels are available for general use.

Table 38-1. Channel Assignments

Channel Number	Assignment
0 – 6	LRADC 0 - 6 measure the voltage on the seven application-dependent LRADC pins. The auxiliary channels can be used for a variety of uses, including a resistor-divider-based wired remote control, external temperature sensing, touch-screen, button and so on. LRADC 2 - 6 can be used for 4/5-wire touch-screen control. LRADC 6 can be used for the wiper of 5-wire touch-screen controller and external temperature sensing, but they cannot be enabled at the same time in hardware configuration. LRADC 5 can be used for Y- of 4-wire and LR of 5-wire; LRADC 4 can be used for X- of 4-wire and UR of 5-wire; LRADC 3 can be used for Y+ of 4-wire and LL of 5-wire; LRADC 2 can be used for X+ and UR of 5-wire; For pull-up or pull-down switch control on LRADC2~5 pins, please refer to HW_LRADC_CTRL0 register. LRADC 0 can be used for button and external temperature sensing, they can not be enabled at same time in hardware configuration. LRADC 1 can be used for button as well as LRADC 0. There are the hardware button press detect circuit for LRADC 0 and 1.
7	Dedicated to measuring the voltage on the BATT pin and can be used to sense the amount of battery life remaining.
8, 9	Dedicated to measuring the internal die temperature. HW_LRADC_CTRL2_TEMPSENSE_PWD must be cleared for these inputs to function. See Internal Die Temperature Sensing.
10	Dedicated to measuring the voltage on the VDDIO rail. Also used to calibrate the voltage levels measured on the auxiliary channels when those inputs are resistor-divided from the VDDIO rail.
11	Reserved input for analog testing.

Channel Number	Assignment
12	Dedicated to measuring the voltage on the VDDA.
13	Dedicated to measuring the voltage on the VDDD.
14	Dedicated to measuring the bandgap reference voltage and can be used to calibrate out a portion of the LRADC measurement error (comparator offset, buffer amp offset, and digital-to-analog converter (DAC) offset). In most cases, the bandgap reference error (specified to ±1%) dominates the total LRADC error, and this calibration is not helpful. But if the bandgap reference is calibrated using the fuses, then it is possible that LRADC accuracy is limited by these other sources and that using the VBG input for calibration of the LRADC can further improve accuracy.
15	Dedicated to measure the voltage on the VDD5V pin and can be used to detect possible issues with 5 V rail dropping.

The LRADC has 12 bit of resolution and an absolute accuracy of 1.3% limited primarily by the accuracy of the bandgap voltage reference. If the bandgap voltage reference is calibrated with the fuses, the LRADC absolute accuracy might be improved to better than 0.5%. All channels sample on the same divided clock rate from the 24.0 MHz crystal clock. The LRADC controller includes an integrated 4-wire/5-wire touch-screen controller with drive voltage generation for touch-screen coordinate measurement, as well as a touch-detection interrupt circuit. The keypad (button) detect and button-detection interrupt circuit are implemented in the controller. It contains four delay-control channels that can be used to automatically time and schedule control events within the LRADC. The controller also implements a threshold-detection function that can signal an interrupt when a programmed value is crossed on a virtual LRADC channel.

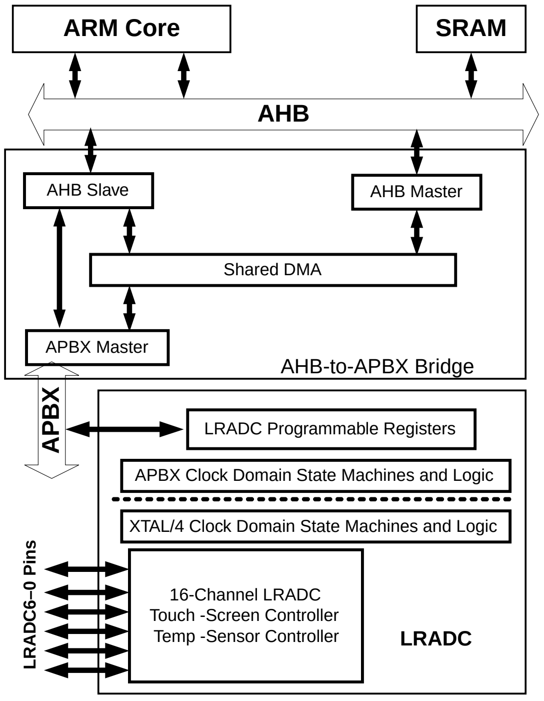


Figure 38-1. Low-Resolution ADC and Touch-Screen Interface Block Diagram

38.2 Operation

All channels of the LRADC share a common successive approximation style analog-to-digital converter through a common analog mux front end (see Figure 38-2).

- The BATT pin has a built-in 4:1 voltage divider on its analog multiplexer input that is activated only in Li-Ion battery mode.
- The Channel 15 5V input also has a built-in 4:1 divider on its input.
- The Channel 10 VDDIO input and the Channel 12 VDDA input have a built-in 2:1 divider on their inputs. The maximum analog input voltage into the LRADC is 1.85 V.
- For input channels (other than BATT, 5V, or VDDIO) with signals larger than 1.85 V, the divide-by-two option should be set (HW_LRADC_CTRL2_DIVIDE_BY_TWO). With the DIVIDE_BY_TWO option set, the maximum input voltage is VDDIO – 50mv.

The touch-screen driver works for typical touch-screen impedances of 200–900 ohm and for high impedance touch-panels with impedances in the 50-Kohm range.

The LRADC channels 0 and 6 have optional current source outputs to allow these channels to be used with an external themistor (or an external diode) for temperature sensing. The controls for these current sources are in HW_LRADC_CTRL2. The current source values can be changed to allow significant temperature sensing range using a thermistor or to use a diode for temperature sensing. The currents are derived using the on-chip 1% accurate bandgap voltage reference and an optionally tuned on-chip poly resistor. The accuracy of the current source is limited by the on-chip resistor, which should be 5% accurate and optionally tuned for higher accuracy (with efuses). Most thermistors are no more than 5% accurate, so this level of current source accuracy is acceptable for most applications. For temperature sensing with higher accuracy, customers can use a 1% resistor divider from VDDIO with the thermistor. In this case, the thermistor will be the dominant source of error.

Note

The pad ESD protection limits the voltage on the LRADC0-LRADC6 inputs to VDDIO. The BATT and 5V inputs to the LRADC have built-in dividers to handle the higher voltages.

38.2.1 External Temperature Sensing with a Diode

Using a diode instead of a thermistor for external temperature sensing can be cheaper and provide greater temperature range for a given accuracy level. An inexpensive diode like a 1N4148 is connected between ground and either LRADC 0 or 6. Two voltage measurements are taken—first with the HW_LRADC_CTRL2_TEMP_ISRC current source set at 300 µA, then another voltage measurement with the current source set at 20 µA. The temperature will be roughly:

degrees Kelvin = (Vmax – Vmin) / 0.409 mV

or, from the LRADC conversion (LSB=0.45mV):

degrees Kelvin = (Codemax – Codemin) * 1.104

Freescale recommends taking 5–10 samples for the min and max and then averaging them to get a good reading.

The temperature reading error will likely be dominated by part-to-part matching of the diodes. Some manufacturers' diodes show substantially less variation than others. Freescale has shown three-sigma accuracy of +/-7.5C using Fairchild MMBD914 (from multiple batches of diodes).

If better accuracy is required, Freescale recommends using a thermistor for external temperature sensing. The thermistor will be more accurate, but over a smaller temperature range than the diode method.

Any routing impedance to the diode will cause a shift in the temperature reading. This can be measured and corrected in software for each design.

Two ohms of routing impedance would cause (2 * (300µA – 20µA) error of 0.56 mV or 1.25 degrees C error.

38.2.2 Internal Die Temperature Sensing

The i.MX28 has a new internal die temperature sensor that uses two of the sixteen physical LRADC channels. To use the internal die temperature sensor, HW_LRADC_CTRL2_TEMPSENSE_PWD should be cleared. (This bit can be left cleared after power up. There is no need to toggle it on and off.) Two of the eight virtual LRADC channels need to be mapped to the temperature sensing channels 8 and 9 using HW_LRADC_CTRL4. Then, these virtual LRADC channels should BOTH be converted using the LRADC conversion scheduler described below. The temperature in degrees Kelvin will be equal to:

(Channel9 – Channel8) * Gain_correction/4

The Gain_correction corrects a mean gain error in the temperature conversion and should be 1.012. After this correction factor, the three-sigma error of the temperature sensor should be within ± 1.5% in degrees Kelvin. Additionally, the temperature sampling has a three-sigma sample-to-sample variation of 2 degrees Kelvin. If desired, this error can be removed by oversampling and averaging the temperature result.

Prior to starting a battery charge cycle, the internal die temperature sensing could be used for an approximate ambient temperature. During high-current battery charging, the temperature sensor can be used as extra protection to avoid excessive die temperatures (to reduce the charging current).

38.2.3 Scheduling Conversions

The APBX clock domain logic schedules conversions on a per-channel basis and handles interrupt processing back to the CPU. Each of the eight virtual channels has its own interrupt request enable bit and its own interrupt request status bit.

A schedule request bit, HW_LRADC_CTRL0_SCHEDULE, exists for each virtual channel. Setting this bit causes the LRADC to schedule a conversion for that virtual channel. Each virtual channel schedule bit is sequentially checked and, if scheduled, causes a conversion. The schedule bit is cleared upon completion of a successive approximation conversion, and its corresponding interrupt request status bit is set. Therefore, software controls how often a conversion is requested. As each scheduled channel is converted, its interrupt status bit is set and its schedule bit is reset.

There is a mechanism to continuously reschedule a conversion for a particular virtual channel. With set/clear/toggle addressing modes, independent threads can request conversions without needing any information from unrelated threads using other channels. Setting a schedule bit can be performed in an atomic way. Setting a group of four channel-schedule bits can also be performed atomically. The LRADC scheduler is round-robin. It snapshots all schedule bits at once, and then processes them in sequence until all are converted. It then monitors the schedule bits. If any schedule bits are set, it snapshots them and starts a new conversion operation for all scheduled channels. Thus, one can set the schedule bits for four channels on the same clock edge. The channel with the largest channel number is converted last and has its interrupt status bit set last. If that channel is the only one of the four with an interrupt enable bit set, then it interrupts the ARM after all four channels have been converted, effectively ganging four channels together.

38.2.4 Delay Channels

To minimize the interrupt load on the ARM processor, four delay channels are provided. Each has an 11-bit counter that increments at 2 KHz. A delay channel can be kicked off either by an ARM store instruction or at the completion of a delay channel time-out. At time-out, each channel has the option of kicking off any combination of LRADC conversions, as well as any combination of delay channels.

Note

> The DELAY fields in HW_LRADC_DELAY0, HW_LRADC_DELAY1, HW_LRADC_DELAY2, and HW_LRADC_DELAY3 must be non-zero; otherwise, the LRADC will not trigger the delay group. The ACCUMULATE bit in the appropriate channel register HW_LRADC_CHn must be set to 1 if NUM_SAMPLES is greater then 0; otherwise, the IRQs will not fire.

Consider the case of a touch-screen that requires 4x oversampling of its coordinate values. Further, suppose you wish to receive an oversampled X or Y coordinate approximately every 5 ms and that the oversampling should be spaced at 1-ms intervals.

- In the touch-screen, first select either X or Y drive, then set up the appropriate LRADC.
- In setting up the LRADC, clear the accumulator associated with it by setting the ACCUMULATE bit and set the NUM_SAMPLES field to 3 (4 samples before interrupt request).
- Next, set up two delay channels.
  - Delay Channel 1 is set to delay 1 ms (DELAY = 1, two ticks) and then kick the schedule bit for LRADC 4. Its LOOP_COUNT bit field is also set to 3, so that four kicks of LRADC 4 occur, each spaced by 1 ms.
  - Delay Channel 0 is set to delay 1 ms with LOOP_COUNT = 0, that is, one time. Its TRIGGER_DELAYS field is set to trigger Delay Channel 1 when it times out. The ISR routine kicks off Delay Channel 0 immediately before it does its return from interrupt. Another interrupt (LRADC4_IRQ) is asserted once the entire 4x oversample data capture is complete. A sample timeline for such a sequence is shown in Figure 38-3.

Note

If a delay group schedules channels to be sampled and a manual write to the schedule field in CTRL0 occurs while the block is discarding samples, the LRADC will switch to the new schedule and will not sample the channels that were previously scheduled. The time window for this to happen is very small and lasts only while the LRADC is discarding samples.

38.3 Channel Threshold Detection

There are two identical threshold detection units implemented in the LRADC. They are configured through the registers HW_LRADC_THRESHOLD0 and HW_LRADC_THRESHOLD1. This functionality compares the sampled 18-bit value from a LRADC channel against a programmed threshold value. It can be programmed to do this comparison in two ways. First, it will detect when the sampled channel value crosses from above the threshold value down to equal or less than that value. Conversely, it can also be set up to detect when the sampled channel value changes from below to equal to or above the threshold value.

When the programmed crossing is detected, the HW_LRADC_CTRL1_THRESHOLDx_DETECT_IRQ interrupt register bits will assert. These bits can be interrupt sources back to the CPU if enabled through the HW_LRADC_CTRL1_THRESHOLDx_DETECT_IRQ_EN bits. The interrupt bits must be cleared before another threshold crossing can be detected. Any one of the eight virtual channels can be mapped to either or both of the threshold detection units. Note that the threshold event can occur on the very first channel conversion after the threshold unit is enable. For example, consider the case where a threshold unit was programmed for DETECT_HIGH and enabled. If the connected channel was then converted and the value was equal or greater than the threshold value this would satisfy the threshold requirement and the event would be logged by setting the IRQ bit. However, after this first conversion, a threshold event will only be tripped again on subsequent conversions if the value drops below the threshold and then crosses it again. In other words, if the channel value remains above the threshold on multiple sequential conversions, the threshold event will be logged only once (after the first conversion in the sequence that tripped the threshold).

Either threshold unit can be used to disable the battery charger when a threshold event occurs. This functionality is enabled by setting the HW_LRADC_THRESHOLDx_BATTCHRG_DISABLE bit. When enabled, and the threshold is tripped, the HW_POWER_CHARGE0_ PWD_BATTCHRG bit will set. This is an event that is triggered once when the conditions are met (PWD_BATTCHRG is not held high). At any time thereafter, the PWD_BATTCHRG bit can be manipulated and reprogrammed by software as normal. The LRADC will only set the bit again when the threshold event is triggered once more.

38.4 Behavior During Reset

A soft reset (SFTRST) can take multiple clock periods to complete, so do NOT set CLKGATE when setting SFTRST. The reset process gates the clocks automatically. See Correct Way to Soft Reset a Block for additional information on using the SFTRST and CLKGATE bit fields.

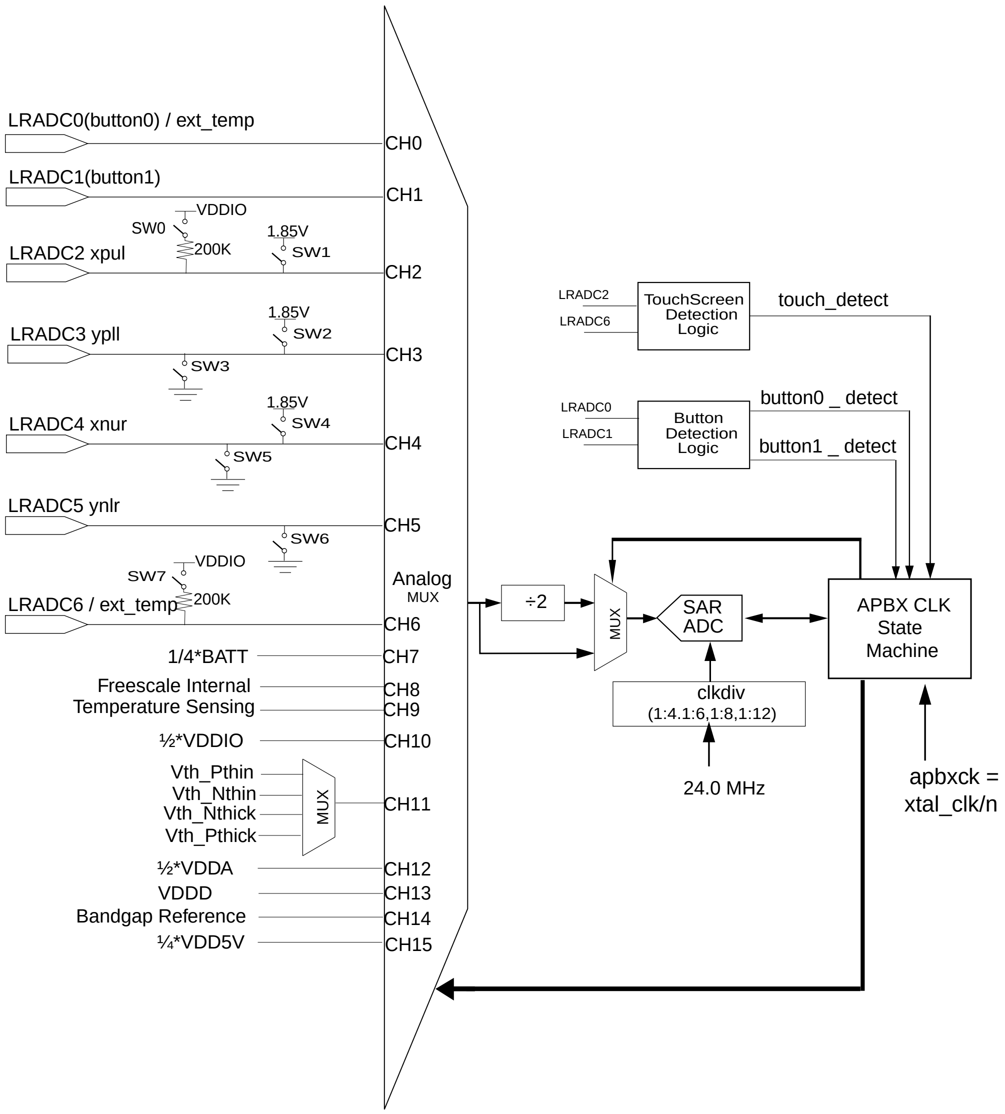


Figure 38-2. Low-Resolution ADC Successive Approximation Unit

Below Table shows the register configure setting and switch status for different touch screen usage:

Table 38-2. The register configure setting and switch status for different touch screen usage

	4-wire touch detect	4-wire X-axis	4-wire Y-axis	5-wire touch detect	5-wire hori-zontal meas-ure	5-wire vertical measure
TOUCH_SCREEN_TYPE	0	0	0	1	1	1

TOUCH_DETECT_ENABLE	1	0	0	1	0	0
YNLRSW	0	0	1	0	1	1
YPLLSW<1:0>	00	00	01	00	01	10
XNURSW<1:0>	00	10	00	00	10	01
XPULSW	0	1	0	0	1	1
SW0 Status	ON	OFF	OFF	OFF	OFF	OFF
SW1 Status	OFF	ON	OFF	OFF	ON	ON
SW2 Status	OFF	OFF	ON	OFF	ON	OFF
SW3 Status	OFF	OFF	OFF	OFF	OFF	ON
SW4 Status	OFF	OFF	OFF	OFF	OFF	ON
SW5 Status	OFF	ON	OFF	OFF	ON	OFF
SW6 Status	ON	OFF	ON	ON	ON	ON
SW7 Status	OFF	OFF	OFF	ON	OFF	OFF

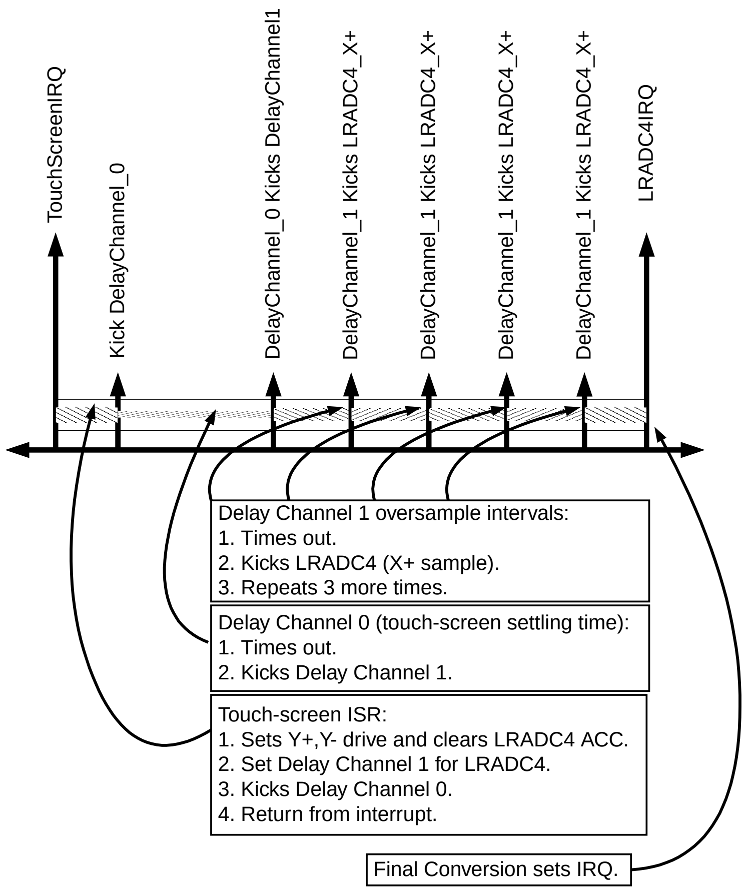

TouchScreenIRQ
Kick DelayChannel_0
DelayChannel_0 Kicks DelayChannel1
DelayChannel_1 Kicks LRADC4_X+
DelayChannel_1 Kicks LRADC4_X+
DelayChannel_1 Kicks LRADC4_X+
DelayChannel_1 Kicks LRADC4_X+
LRADC4IRQ
Delay Channel 1 oversample intervals:
1. Times out.
2. Kicks LRADC4 (X+ sample).
3. Repeats 3 more times.
Delay Channel 0 (touch-screen settling time):
1. Times out.
2. Kicks Delay Channel 1.
Touch-screen ISR:
1. Sets Y+,Y- drive and clears LRADC4 ACC.
2. Set Delay Channel 1 for LRADC4.
3. Kicks Delay Channel 0.
4. Return from interrupt.
Final Conversion sets IRQ.

Figure 38-3. Using Delay Channels to Oversample a Touch-Screen

38.5 Programmable Registers

LRADC Hardware Register Format Summary

HW_LRADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
8005_0000	LRADC Control Register 0 (HW_LRADC_CTRL0)	32	R/W	C000_0000h	38.5.1/2271
8005_0010	LRADC Control Register 1 (HW_LRADC_CTRL1)	32	R/W	0000_0000h	38.5.2/2274
8005_0020	LRADC Control Register 2 (HW_LRADC_CTRL2)	32	R/W	0000_8000h	38.5.3/2277
8005_0030	LRADC Control Register 3 (HW_LRADC_CTRL3)	32	R/W	0000_0000h	38.5.4/2280
8005_0040	LRADC Status Register (HW_LRADC_STATUS)	32	R	1FFF_0000h	38.5.5/2282
8005_0050	LRADC 0 Result Register (HW_LRADC_CH0)	32	R/W	0000_0000h	38.5.6/2284
8005_0060	LRADC 1 Result Register (HW_LRADC_CH1)	32	R/W	0000_0000h	38.5.7/2286
8005_0070	LRADC 2 Result Register (HW_LRADC_CH2)	32	R/W	0000_0000h	38.5.8/2288
8005_0080	LRADC 3 Result Register (HW_LRADC_CH3)	32	R/W	0000_0000h	38.5.9/2289
8005_0090	LRADC 4 Result Register (HW_LRADC_CH4)	32	R/W	0000_0000h	38.5.10/2291
8005_00A0	LRADC 5 Result Register (HW_LRADC_CH5)	32	R/W	0000_0000h	38.5.11/2293
8005_00B0	LRADC 6 Result Register (HW_LRADC_CH6)	32	R/W	0000_0000h	38.5.12/2294
8005_00C0	LRADC 7 (BATT) Result Register (HW_LRADC_CH7)	32	R/W	0000_0000h	38.5.13/2296
8005_00D0	LRADC Scheduling Delay 0 (HW_LRADC_DELAY0)	32	R/W	0000_0000h	38.5.14/2298
8005_00E0	LRADC Scheduling Delay 1 (HW_LRADC_DELAY1)	32	R/W	0000_0000h	38.5.15/2299
8005_00F0	LRADC Scheduling Delay 2 (HW_LRADC_DELAY2)	32	R/W	0000_0000h	38.5.16/2301
8005_0100	LRADC Scheduling Delay 3 (HW_LRADC_DELAY3)	32	R/W	0000_0000h	38.5.17/2302
8005_0110	LRADC Debug Register 0 (HW_LRADC_DEBUG0)	32	R	4321_0000h	38.5.18/2304
8005_0120	LRADC Debug Register 1 (HW_LRADC_DEBUG1)	32	R/W	0000_0000h	38.5.19/2304
8005_0130	LRADC Battery Conversion Register (HW_LRADC_CONVERSION)	32	R/W	0000_0080h	38.5.20/2306
8005_0140	LRADC Control Register 4 (HW_LRADC_CTRL4)	32	R/W	7654_3210h	38.5.21/2307
8005_0150	LRADC Theshold0 Register (HW_LRADC_THRESHOLD0)	32	R/W	0000_0000h	38.5.22/2311
8005_0160	LRADC Theshold1 Register (HW_LRADC_THRESHOLD1)	32	R/W	0000_0000h	38.5.23/2313
8005_0170	LRADC Version Register (HW_LRADC_VERSION)	32	R	0103_0000h	38.5.24/2314

38.5.1 LRADC Control Register 0 (HW_LRADC_CTRL0)

The LRADC Control Register 0 provides overall control of the eight low resolution analog to digital converters.

HW_LRADC_CTRL0: 0x000

HW_LRADC_CTRL0_SET: 0x004

HW_LRADC_CTRL0_CLR: 0x008

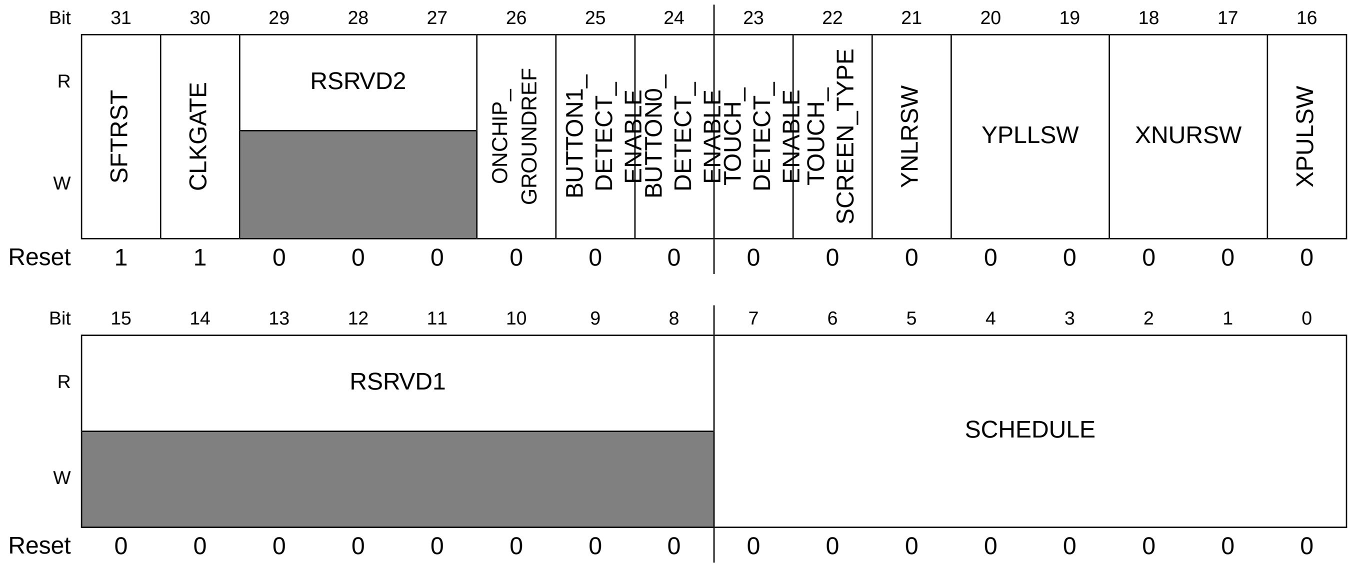
HW_LRADC_CTRL0_TOG: 0x00C

The LRADC control register provides control over the shared eight channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE

```
BW_LRADC_CTRL0_YPLUS_ENABLE(BV_LRADC_CTRL0_YPLUS_ENABLE__ON);
```

Address: HW_LRADC_CTRL0 – 8005_0000h base + 0h offset = 8005_0000h



Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	RSRVD2			ONCHIP_ GROUNDREF	BUTTON1_ DETECT_ ENABLE	BUTTON0_ DETECT_ ENABLE	TOUCH_ DETECT_ ENABLE	TOUCH_ SCREEN_TYPE	YNLRSW	YPLLSW		XNURSW		XPULSW
W																
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1								SCHEDULE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CTRL0 field descriptions

Field	Description
31 SFTRST	When set to one, this bit causes a reset to the entire LRADC block. In addition, it turns off the converter clock and powers down the analog portion of the LRADC. Set this bit to zero for normal operation.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29–27 RSRVD2	Reserved
26 ONCHIP_ GROUNDREF	Set this bit to one to use the on-chip ground as reference for conversions. 0x0 **OFF** — Turn it off. 0x1 **ON** — Turn it on.

HW_LRADC_CTRL0 field descriptions (continued)

Field	Description
25 BUTTON1_ DETECT_ ENABLE	Set this bit to one to enable button1 detector. 0x0 **OFF** — Turn it off. 0x1 **ON** — Turn it on.
24 BUTTON0_ DETECT_ ENABLE	Set this bit to one to enable button0 detector. 0x0 **OFF** — Turn it off. 0x1 **ON** — Turn it on.
23 TOUCH_ DETECT_ ENABLE	Set this bit to one to enable touch panel touch detector. 0x0 **OFF** — Turn it off. 0x1 **ON** — Turn it on.
22 TOUCH_ SCREEN_TYPE	Touch Screen Type (4/5 wires) Select control. 0x0: 4-wire 0x1: 5-wire
21 YNLRSW	YNLR Switch Control, to enable pull down on the LRADC5 pin. YNLRSW bit controls the on/off for ynnsw 0x0: YNLR OFF (ynnsw off) 0x1: YNLR LOW (ynnsw on)
20–19 YPLLSW	YPLL Switch Control, to enable pull up or pull down on the LRADC3 pin. YPLLSW[1] controls the on/off for ypnsw YPLLSW[0] controls the on/off for yppsw 0x0: YPLL OFF (ypnsw off, yppsw off) 0x1: YPLL HIGH (ypnsw off, yppsw on) 0x2: YPLL LOW (ypnsw on, yppsw off) 0x3: Reserved
18–17 XNURSW	XNUR Switch Control, to enable pull down or pull up on the LRADC4 pin. XNUR[1] controls the on/off for xnnsw XNUR[0] controls the on/off for xnpsw 0x0: XNUR OFF (xnnsw off, xnpsw off) 0x1: XNUR HIGH (xnnsw off, xnpsw on) 0x2: XNUR LOW (xnnsw on, xnpsw off) 0x3: Reserved
16 XPULSW	XPUL Switch Control, to enable pull up on the LRADC2 pin. XPULSW bit controls the on/off for xppsw 0x0: XPUL OFF (xppsw off) 0x1: XPUL HIGH (xppsw on)

HW_LRADC_CTRL0 field descriptions (continued)

Field	Description
15–8 RSRVD1	Reserved
7–0 SCHEDULE	Setting a bit to one schedules the corresponding LRADC channel to be converted. When the conversion of a scheduled channel is completed the corresponding schedule bit is reset by the hardware and the corresponding interrupt request is set to one. Thus any thread can request a conversion asynchronously from any other thread.

38.5.2 LRADC Control Register 1 (HW_LRADC_CTRL1)

The LRADC Control Register 1 provides overall control of the eight low resolution analog to digital converters.

HW_LRADC_CTRL1: 0x010

HW_LRADC_CTRL1_SET: 0x014

HW_LRADC_CTRL1_CLR: 0x018

HW_LRADC_CTRL1_TOG: 0x01C

The LRADC control register 1 provides control over the shared eight channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE

```
if(HW_LRADC_CTRL1.TOUCH_DETECT_IRQ == BV_LRADC_CTRL1_TOUCH_DETECT_IRQ__PENDING){
  // Then handle the interrupt.
  HW_LRADC_CTRL1.TOUCH_DETECT_IRQ_EN = BV_LRADC_CTRL1_TOUCH_DETECT_IRQ_EN__DISABLE;
}
```

Address: HW_LRADC_CTRL1 – 8005_0000h base + 10h offset = 8005_0010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD2			BUTTON1_DETECT_IRQ_EN	BUTTON0_DETECT_IRQ_EN	THRESHOLD1_DETECT_IRQ_EN	THRESHOLD0_DETECT_IRQ_EN	TOUCH_DETECT_IRQ_EN	LRADC7_IRQ_EN	LRADC6_IRQ_EN	LRADC5_IRQ_EN	LRADC4_IRQ_EN	LRADC3_IRQ_EN	LRADC2_IRQ_EN	LRADC1_IRQ_EN	LRADC0_IRQ_EN
W	▨	▨	▨													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1			BUTTON1_ DETECT_IRQ	BUTTON0_ DETECT_IRQ	THRESHOLD1_ DETECT_IRQ	THRESHOLD0_ DETECT_IRQ	TOUCH_ DETECT_IRQ	LRADC7_IRQ	LRADC6_IRQ	LRADC5_IRQ	LRADC4_IRQ	LRADC3_IRQ	LRADC2_IRQ	LRADC1_IRQ	LRADC0_IRQ
W	▨	▨	▨													
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CTRL1 field descriptions

Field	Description
31–29 RSRVD2	Reserved
28 BUTTON1_ DETECT_IRQ_ EN	Set to one to enable an interrupt for the button1 detect logic. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
27 BUTTON0_ DETECT_IRQ_ EN	Set to one to enable an interrupt for the button0 detect logic. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
26 THRESHOLD1_ DETECT_IRQ_ EN	Set to one to enable an interrupt for the thesholdO detect logic. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
25 THRESHOLD0_ DETECT_IRQ_ EN	Set to one to enable an interrupt for the thesholdO detect logic. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
24 TOUCH_ DETECT_IRQ_ EN	Set to one to enable an interrupt for the touch detector comparator. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
23 LRADC7_IRQ_ EN	Set to one to enable an interrupt for channel 7 (BATT) conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
22 LRADC6_IRQ_ EN	Set to one to enable an interrupt for channel 6 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
21 LRADC5_IRQ_ EN	Set to one to enable an interrupt for channel 5 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
20 LRADC4_IRQ_ EN	Set to one to enable an interrupt for channel 4 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.

HW_LRADC_CTRL1 field descriptions (continued)

Field	Description
19 LRADC3_IRQ_ EN	Set to one to enable an interrupt for channel 3 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
18 LRADC2_IRQ_ EN	Set to one to enable an interrupt for channel 2 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
17 LRADC1_IRQ_ EN	Set to one to enable an interrupt for channel 1 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
16 LRADC0_IRQ_ EN	Set to one to enable an interrupt for channel 0 conversions. 0x0 **DISABLE** — Disable Interrupt request. 0x1 **ENABLE** — Enable Interrupt request.
15–13 RSRVD1	Reserved
12 BUTTON1_ DETECT_IRQ	This bit is set to one upon detection of the button1 condition in button matrix and attached to LRADC1. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
11 BUTTON0_ DETECT_IRQ	This bit is set to one upon detection of the threshold1 condition in button matrix and attached to LRADC0. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
10 THRESHOLD1_ DETECT_IRQ	This bit is set to one upon detection of the threshold1 condition. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
9 THRESHOLD0_ DETECT_IRQ	This bit is set to one upon detection of the threshold0 condition. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
8 TOUCH_ DETECT_IRQ	This bit is set to one upon detection of a touch condition in the touch panel attached to LRADC2-LRADC6. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
7 LRADC7_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 7(BATT). It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software.

HW_LRADC_CTRL1 field descriptions (continued)

Field	Description
	0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
6 LRADC6_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 6. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
5 LRADC5_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 5. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
4 LRADC4_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 4. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
3 LRADC3_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 3. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
2 LRADC2_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 2. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
1 LRADC1_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 1. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.
0 LRADC0_IRQ	This bit is set to one upon completion of a scheduled conversion for channel 0. It is ANDed with its corresponding interrupt enable bit to request an interrrupt. Once set by the conversion hardware, this bit remains set until cleared by software. 0x0 **CLEAR** — Interrupt request cleared. 0x1 **PENDING** — Interrupt request pending.

38.5.3 LRADC Control Register 2 (HW_LRADC_CTRL2)

The LRADC Control Register 2 provides overall control of the eight low resolution analog to digital converters.

HW_LRADC_CTRL2: 0x020

HW_LRADC_CTRL2_SET: 0x024

HW_LRADC_CTRL2_CLR: 0x028

HW_LRADC_CTRL2_TOG: 0x02C

The LRADC control register 2 provides control over the shared eight channel LRADC converter. It allows software to independently schedule conversion cycles on any number of any size subsets of the eight channels. In addition it allows software to manage the interrupt reporting for the channel conversion sets.

EXAMPLE

```
BW_LRADC_CTRL2_TEMP_SENSOR_IENABLE1(BV_LRADC_CTRL2_TEMP_SENSOR_IENABLE1__DISABLE);
```

Address: HW_LRADC_CTRL2 – 8005_0000h base + 20h offset = 8005_0020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DIVIDE_BY_TWO								RSRVD3							
W									▨							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TEMPSENSE_PWD	VTHSENSE		DISABLE_MUXAMP_BYPASS	RSRVD2		TEMP_SENSOR_IENABLE1	TEMP_SENSOR_IENABLE0	TEMP_ISRC1				TEMP_ISRC0			
W					▨											
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CTRL2 field descriptions

Field	Description
31–24 DIVIDE_BY_TWO	Each bit of this eight bit field corresponds to a channel of an LRADC. Setting the bit to one caused the A/D converter to use its analog divide by two circuit for the conversion of the corresponding channel.
23–16 RSRVD3	Reserved
15 TEMPSENSE_PWD	Power down the tempsense block. 0x0 **ENABLE** — When this is low the tempsense gain block muxes to LRADC channel 8 and 9. 0x1 **DISABLE** — .
14–13 VTHSENSE	Vth Measurement Control (Channel11)

HW_LRADC_CTRL2 field descriptions (continued)

Field	Description
	0x0:pmos_tn (thin oxide pmos) 0x1:nmos_tn (thin oxide nmos) 0x2:nmos_tk (thick oxide nmos) 0x3:pmos_tk (thick oxide pmos)
12 DISABLE_ MUXAMP_ BYPASS	When set to zero (default) the mux amp is bypassed when the LRADC input channel is not using the divide-by-two. When set to one the mux amp is never bypassed (old behavior).
11–10 RSRVD2	Reserved
9 TEMP_ SENSOR_ IENABLE1	Set this bit to one to enable the current source onto LRADC6. This bit must be set to 0 when the function of temp sensor not used in application. 0x0 **DISABLE** — Disable Temperature Sensor Current Source. 0x1 **ENABLE** — Enable Temperature Sensor Current Source.
8 TEMP_ SENSOR_ IENABLE0	Set this bit to one to enable the current source onto LRADC0. This bit must be set to 0 when the function of temp sensor not used in application. 0x0 **DISABLE** — Disable Temperature Sensor Current Source. 0x1 **ENABLE** — Enable Temperature Sensor Current Source.
7–4 TEMP_ISRC1	When the output voltage is lower than 1V the output current is 1uA higher than the decode shown above. This extra current drops to zero as the output voltage raises above 1.5V. This four bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC6. 0xF **300** — 300uA. 0xE **280** — 280uA. 0xD **260** — 260uA. 0xC **240** — 240uA. 0xB **220** — 220uA. 0xA **200** — 200uA. 0x9 **180** — 180uA. 0x8 **160** — 160uA. 0x7 **140** — 140uA. 0x6 **120** — 120uA. 0x5 **100** — 100uA. 0x4 **80** — 80uA. 0x3 **60** — 60uA. 0x2 **40** — 40uA. 0x1 **20** — 20uA. 0x0 **ZERO** — 0uA.
3–0 TEMP_ISRC0	When the output voltage is lower than 1V the output current is 1uA higher than the decode shown above. This extra current drops to zero as the output voltage raises above 1.5V. This four bit field encodes the current magnitude to inject into an external temperature sensor attached to LRADC0. 0xF **300** — 300uA. 0xE **280** — 280uA.

HW_LRADC_CTRL2 field descriptions (continued)

Field	Description
	0xD **260** — 260uA. 0xC **240** — 240uA. 0xB **220** — 220uA. 0xA **200** — 200uA. 0x9 **180** — 180uA. 0x8 **160** — 160uA. 0x7 **140** — 140uA. 0x6 **120** — 120uA. 0x5 **100** — 100uA. 0x4 **80** — 80uA. 0x3 **60** — 60uA. 0x2 **40** — 40uA. 0x1 **20** — 20uA. 0x0 **ZERO** — 0uA.

38.5.4 LRADC Control Register 3 (HW_LRADC_CTRL3)

The LRADC touch panel control register specifies the voltages at which a touch detect interrupt is generated.

HW_LRADC_CTRL3: 0x030

HW_LRADC_CTRL3_SET: 0x034

HW_LRADC_CTRL3_CLR: 0x038

HW_LRADC_CTRL3_TOG: 0x03C

The LRADC touch detect control and status register controls the voltage at which a touch detection interrupt is generated. This register also contains the interrupt request status bit and enable bit for the touch detection interrupt request to the CPU's IRQ interrupt input.

EXAMPLE

```
BW_LRADC_CTRL3_HIGH_TIME(BV_LRADC_CTRL3_HIGH_TIME__83NS);
BW_LRADC_CTRL3_INVERT_CLOCK(BV_LRADC_CTRL3_INVERT_CLOCK__NORMAL);
```

Address: HW_LRADC_CTRL3 – 8005_0000h base + 30h offset = 8005_0030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD5						DISCARD		FORCE_ANALOG_PWUP	FORCE_ANALOG_PWDN	RSRVD4[21:16]					
W	▨	▨	▨	▨	▨	▨					▨	▨	▨	▨	▨	▨
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD4 [15:14]		RSRVD3				CYCLE_TIME		RSRVD2		HIGH_TIME		RSRVD1		DELAY_CLOCK	INVERT_CLOCK
W	▨	▨	▨	▨	▨	▨			▨	▨			▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CTRL3 field descriptions

Field	Description
31–26 RSRVD5	Reserved
25–24 DISCARD	This bit field specifies the number of samples to discard whenever the LRADC analog is first powered up. 0x0= discard first 3 samples before first capture 0x1= discard first sample before first capture 0x2= discard first 2 samples before first capture 0x3= discard first 3 samples before first capture
23 FORCE_ ANALOG_PWUP	Set this bit to zero for normal operation. Setting it to one forces an analog power up, regardless of where the digital state machine may be. 0x0 **OFF** — Turn it off. 0x1 **ON** — Turn it on.
22 FORCE_ ANALOG_PWDN	Set this bit to zero for normal operation. Setting it to one forces an analog power down, regardless of where the digital state machine may be. 0x0 **ON** — Turn it on. 0x1 **OFF** — Turn it off.
21–14 RSRVD4	Reserved
13–10 RSRVD3	Reserved
9–8 CYCLE_TIME	Changes the LRADC clock frequency. Note: the sample rate is one thirteenth of the frequency selected here. 0x0 **6MHZ** — 6MHz clock. 0x1 **4MHZ** — 4MHz clock. 0x2 **3MHZ** — 3MHz clock. 0x3 **2MHZ** — 2MHz clock.
7–6 RSRVD2	Reserved

HW_LRADC_CTRL3 field descriptions (continued)

Field	Description
5–4 HIGH_TIME	When CYCLE_TIME=00 only 00 and 01 are valid for HIGH_TIME. When CYCLE_TIME=01 only 00,01,and 10 are valid Changes the duty cycle (time high) for the LRADC clock. 0x0 **42NS** — Duty cycle high time to 41.66ns. 0x1 **83NS** — Duty cycle high time to 83.33ns. 0x2 **125NS** — Duty cycle high time to 125ns. 0x3 **250NS** — Duty cycle high time to 250ns.
3–2 RSRVD1	Reserved
1 DELAY_CLOCK	Set this bit to one to delay the 24MHz clock used in the LRADC even further away from the predominant rising edge used within the digital section. The delay inserted is approximately 400pS. 0x0 **NORMAL** — Normal operation, that is no delay. 0x1 **DELAYED** — Delay the clock.
0 INVERT_CLOCK	Set this bit field to one to invert the 24MHz clock where it comes into the LRADC analog section. This moves it away from the predominant digital rising edge. Setting this bit to one causes the A/D converter to run from the negative edge of the divided clock, effectively shifting the conversion point away from the edge used by the DCDC converter. 0x0 **NORMAL** — Run the clock in normal that is not inverted mode. 0x1 **INVERT** — Inver the clock.

38.5.5 LRADC Status Register (HW_LRADC_STATUS)

The LRADC status register returns various read-only status bit field values.

HW_LRADC_STATUS: 0x040

HW_LRADC_STATUS_SET: 0x044

HW_LRADC_STATUS_CLR: 0x048

HW_LRADC_STATUS_TOG: 0x04C

EXAMPLE

```
if(HW_LRADC_STATUS.TOUCH_DETECT_RAW == BV_LRADC_STATUS_TOUCH_DETECT_RAW__HIT){
  // Then something is touching the screen.
}
```

Address: HW_LRADC_STATUS – 8005_0000h base + 40h offset = 8005_0040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3			BUTTON1_ PRESENT	BUTTON0_ PRESENT	TEMP1_ PRESENT	TEMP0_ PRESENT	TOUCH_ PANEL_ PRESENT	CHANNEL7_ PRESENT	CHANNEL6_ PRESENT	CHANNEL5_ PRESENT	CHANNEL4_ PRESENT	CHANNEL3_ PRESENT	CHANNEL2_ PRESENT	CHANNEL1_ PRESENT	CHANNEL0_ PRESENT
W																
Reset	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD2													BUTTON1_ DETECT_RAW	BUTTON0_ DETECT_RAW	TOUCH_ DETECT_RAW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_STATUS field descriptions

Field	Description
31–29 RSRVD3	Reserved
28 BUTTON1_ PRESENT	This read-only bit returns a one when the button detect controller 1 is present on the chip.
27 BUTTON0_ PRESENT	This read-only bit returns a one when the button detect controller 0 is present on the chip.
26 TEMP1_ PRESENT	This read-only bit returns a one when the temperature sensor 1 current source is present on the chip.
25 TEMP0_ PRESENT	This read-only bit returns a one when the temperature sensor 0 current source is present on the chip.
24 TOUCH_PANEL_ PRESENT	This read-only bit returns a one when the touch panel controller function is present on the chip.
23 CHANNEL7_ PRESENT	This read-only bit returns a one when the LRADC channel 7 converter function is present on the chip.

HW_LRADC_STATUS field descriptions (continued)

Field	Description
22 CHANNEL6_ PRESENT	This read-only bit returns a one when the LRADC channel 6 converter function is present on the chip.
21 CHANNEL5_ PRESENT	This read-only bit returns a one when the LRADC channel 5 converter function is present on the chip.
20 CHANNEL4_ PRESENT	This read-only bit returns a one when the LRADC channel 4 converter function is present on the chip.
19 CHANNEL3_ PRESENT	This read-only bit returns a one when the LRADC channel 3 converter function is present on the chip.
18 CHANNEL2_ PRESENT	This read-only bit returns a one when the LRADC channel 2 converter function is present on the chip.
17 CHANNEL1_ PRESENT	This read-only bit returns a one when the LRADC channel 1 converter function is present on the chip.
16 CHANNEL0_ PRESENT	This read-only bit returns a one when the LRADC channel 0 converter function is present on the chip.
15–3 RSRVD2	Reserved
2 BUTTON1_ DETECT_RAW	This read-only bit shows the status of the BUTTON1 Detect Comparator in the analog section. 0x0 **OPEN** — No contact, i.e. open connection. 0x1 **HIT** — Someone is clicking the button.
1 BUTTON0_ DETECT_RAW	This read-only bit shows the status of the BUTTON0 Detect Comparator in the analog section. 0x0 **OPEN** — No contact, i.e. open connection. 0x1 **HIT** — Someone is clicking some button.
0 TOUCH_ DETECT_RAW	This read-only bit shows the status of the Touch Detect Comparator in the analog section. 0x0 **OPEN** — No contact, i.e. open connection. 0x1 **HIT** — Someone is touching the panel.

38.5.6 LRADC 0 Result Register (HW_LRADC_CH0)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel zero.

HW_LRADC_CH0: 0x050

HW_LRADC_CH0_SET: 0x054

HW_LRADC_CH0_CLR: 0x058

HW_LRADC_CH0_TOG: 0x05C

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(0).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(0, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC0_IRQ != BV_LRADC_CTRL1_LRADC0_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(0).VALUE / 5;
```

Address: HW_LRADC_CH0 – 8005_0000h base + 50h offset = 8005_0050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE[17:16]	
W	TOGGLE	▨	ACCUMULATE	NUM_SAMPLES					▨						VALUE[17:16]	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE[15:0]															
W	VALUE[15:0]															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CH0 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.7 LRADC 1 Result Register (HW_LRADC_CH1)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel one.

HW_LRADC_CH1: 0x060

HW_LRADC_CH1_SET: 0x064

HW_LRADC_CH1_CLR: 0x068

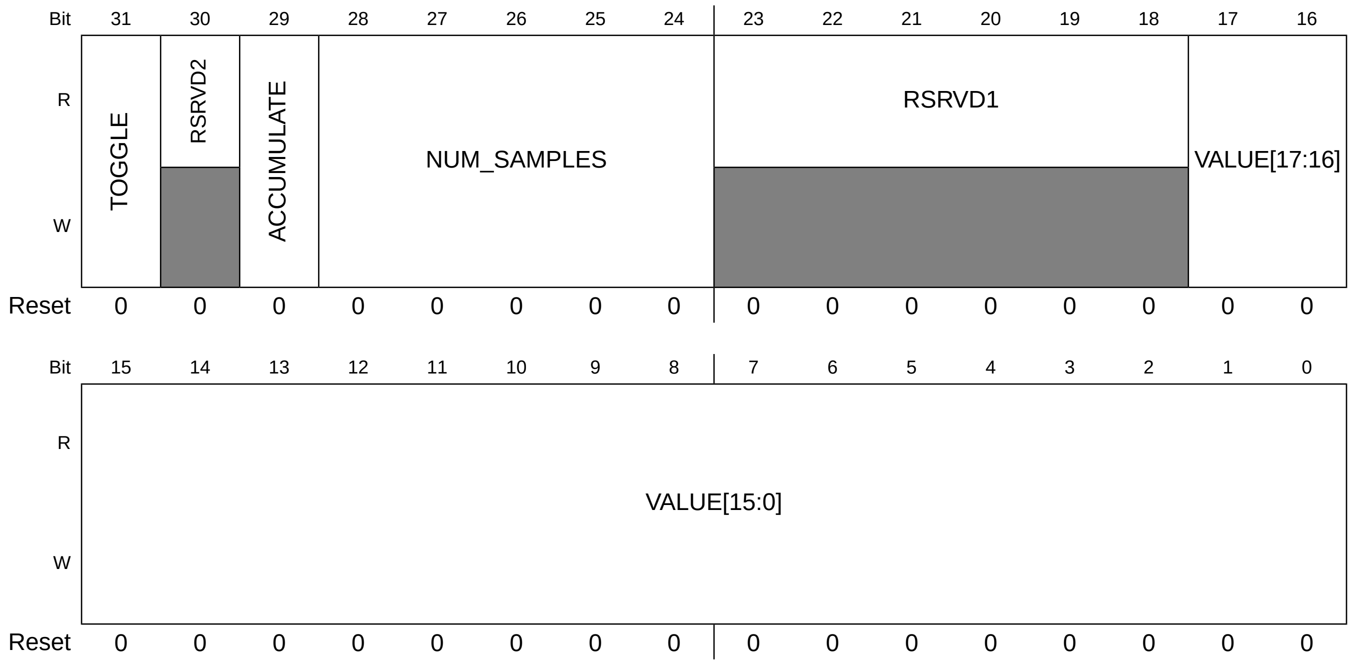
HW_LRADC_CH1_TOG: 0x06C

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(1).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(1, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC1_IRQ != BV_LRADC_CTRL1_LRADC1_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(1).VALUE / 5;
```

Address: HW_LRADC_CH1 – 8005_0000h base + 60h offset = 8005_0060h



HW_LRADC_CH1 field descriptions

Field	Description
31 TOGGLE	This toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.8 LRADC 2 Result Register (HW_LRADC_CH2)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel two.

HW_LRADC_CH2: 0x070

HW_LRADC_CH2_SET: 0x074

HW_LRADC_CH2_CLR: 0x078

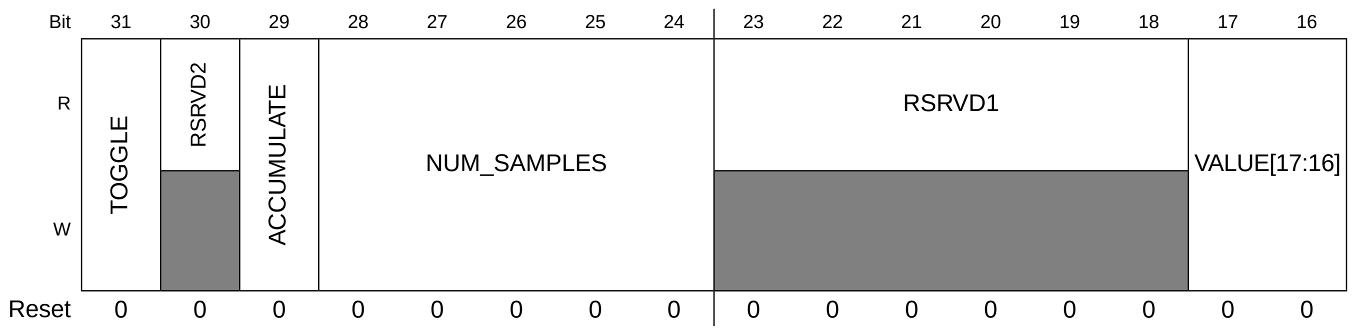
HW_LRADC_CH2_TOG: 0x07C

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(2).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(2, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC2_IRQ != BV_LRADC_CTRL1_LRADC2_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(2).VALUE / 5;
```

Address: HW_LRADC_CH2 – 8005_0000h base + 70h offset = 8005_0070h



Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CH2 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.9 LRADC 3 Result Register (HW_LRADC_CH3)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel three.

HW_LRADC_CH3: 0x080

HW_LRADC_CH3_SET: 0x084

HW_LRADC_CH3_CLR: 0x088

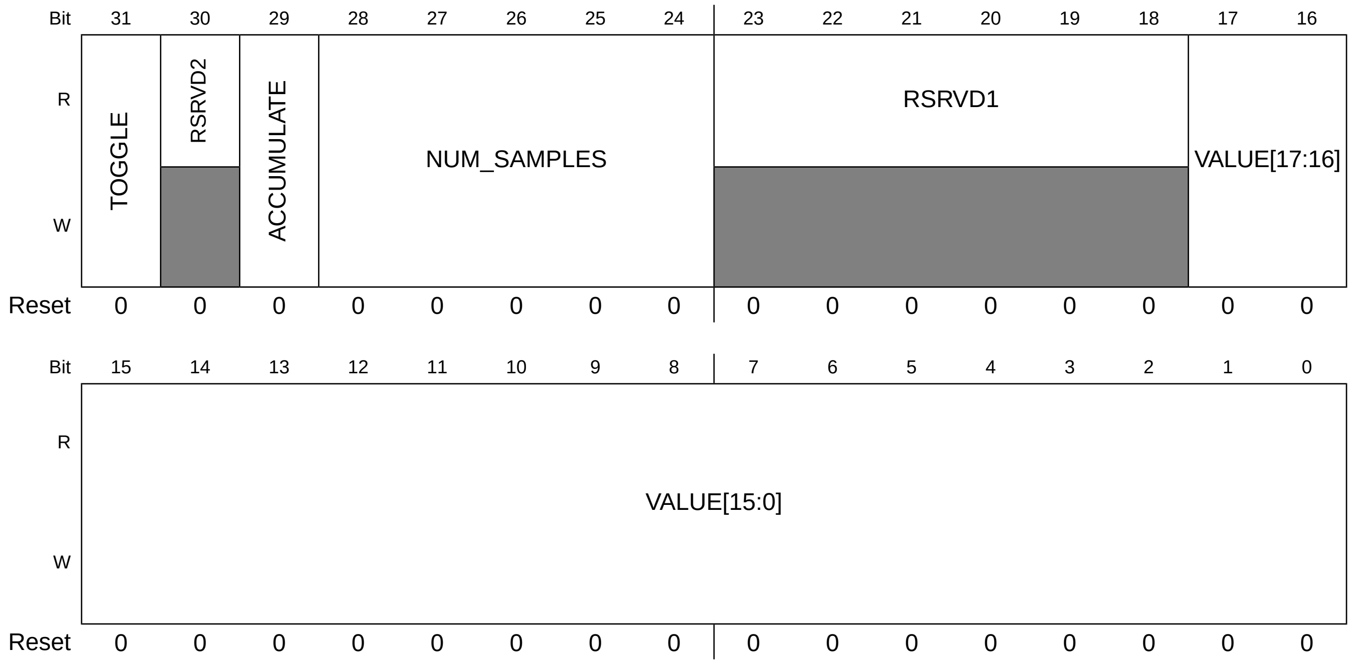
HW_LRADC_CH3_TOG: 0x08C

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(3).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(3, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC3_IRQ != BV_LRADC_CTRL1_LRADC3_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(3).VALUE / 5;
```

Address: HW_LRADC_CH3 – 8005_0000h base + 80h offset = 8005_0080h



HW_LRADC_CH3 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved

HW_LRADC_CH3 field descriptions (continued)

Field	Description
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.10 LRADC 4 Result Register (HW_LRADC_CH4)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel four.

HW_LRADC_CH4: 0x090

HW_LRADC_CH4_SET: 0x094

HW_LRADC_CH4_CLR: 0x098

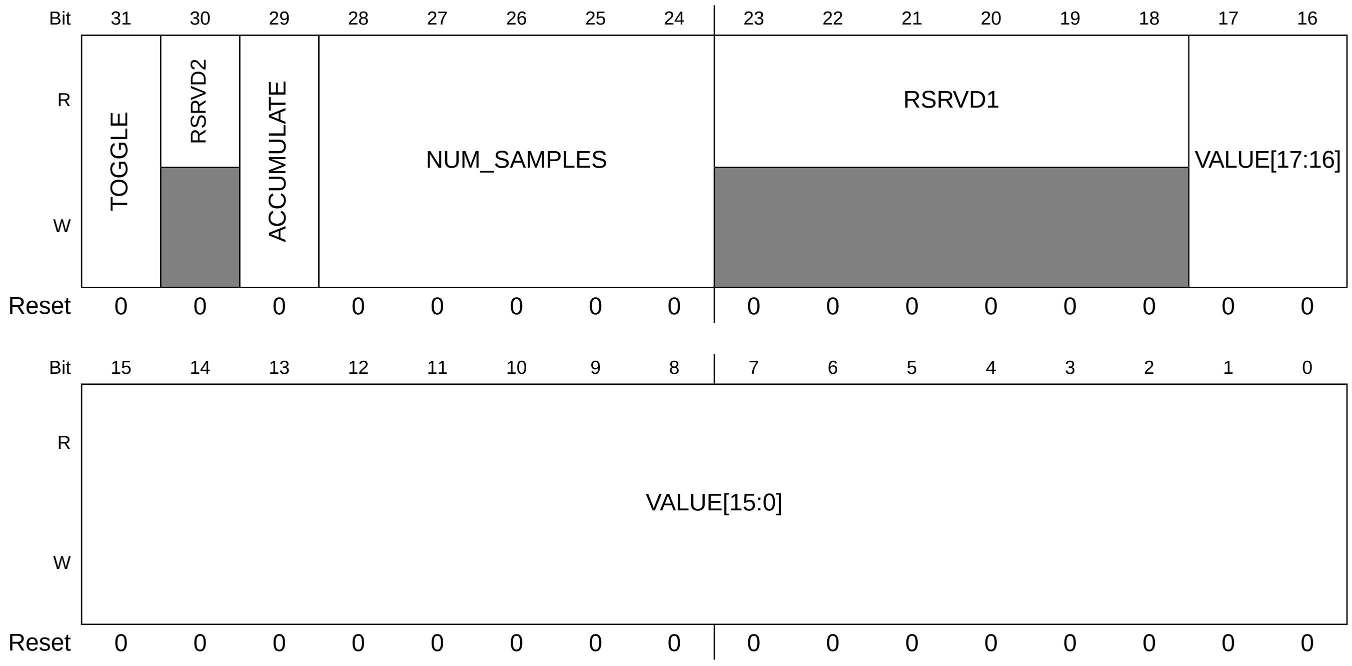
HW_LRADC_CH4_TOG: 0x09C

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(4).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(4, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC4_IRQ != BV_LRADC_CTRL1_LRADC4_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(4).VALUE / 5;
```

Address: HW_LRADC_CH4 – 8005_0000h base + 90h offset = 8005_0090h



HW_LRADC_CH4 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.11 LRADC 5 Result Register (HW_LRADC_CH5)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel five.

HW_LRADC_CH5: 0x0A0

HW_LRADC_CH5_SET: 0x0A4

HW_LRADC_CH5_CLR: 0x0A8

HW_LRADC_CH5_TOG: 0x0AC

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(5).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(5, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC5_IRQ != BV_LRADC_CTRL1_LRADC5_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(5).VALUE / 5;
```

Address: HW_LRADC_CH5 – 8005_0000h base + A0h offset = 8005_00A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TOGGLE	RSRVD2	ACCUMULATE	NUM_SAMPLES					RSRVD1						VALUE[17:16]	
W		▨							▨	▨	▨	▨	▨	▨		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_CH5 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.12 LRADC 6 Result Register (HW_LRADC_CH6)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel six.

HW_LRADC_CH6: 0x0B0

HW_LRADC_CH6_SET: 0x0B4

HW_LRADC_CH6_CLR: 0x0B8

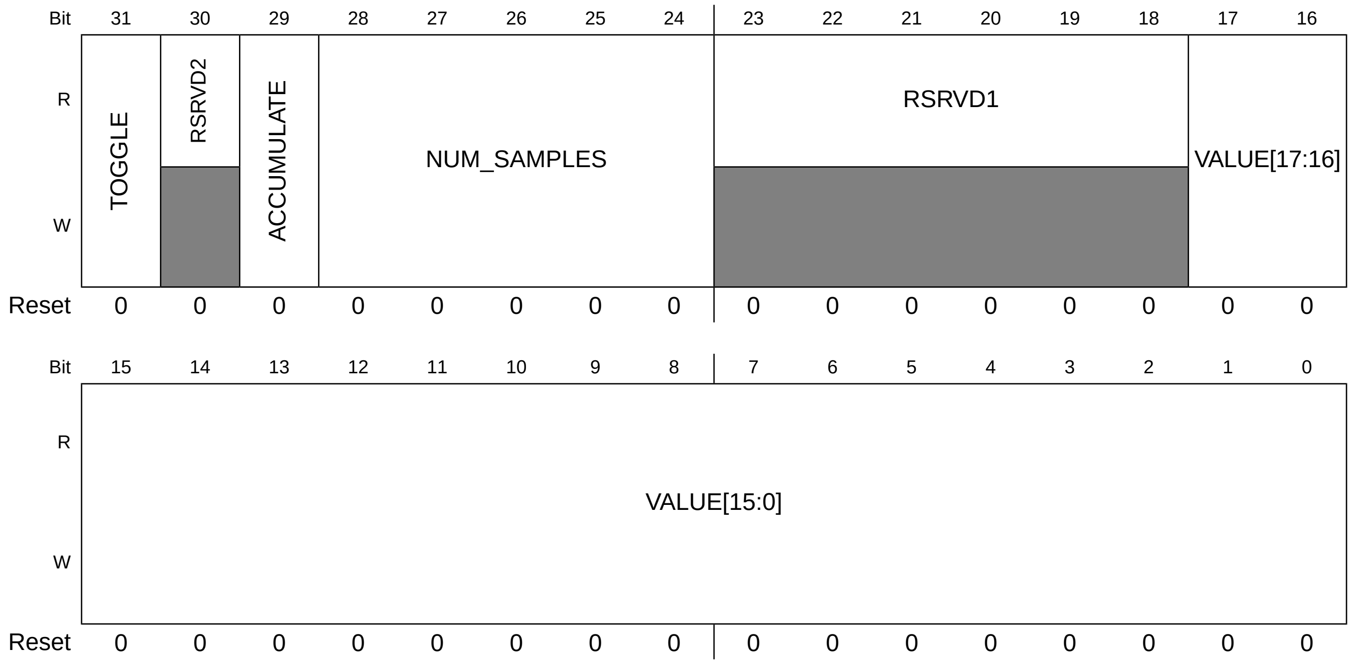
HW_LRADC_CH6_TOG: 0x0BC

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(6).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(6, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC6_IRQ != BV_LRADC_CTRL1_LRADC6_IRQ__PENDING)
{
  // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(6).VALUE / 5;
```

Address: HW_LRADC_CH6 – 8005_0000h base + B0h offset = 8005_00B0h



HW_LRADC_CH6 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 RSRVD2	Reserved

HW_LRADC_CH6 field descriptions (continued)

Field	Description
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.13 LRADC 7 (BATT) Result Register (HW_LRADC_CH7)

The LRADC result register returns the 12-bit result for low resolution analog to digital converter channel seven (BATT).

HW_LRADC_CH7: 0x0C0

HW_LRADC_CH7_SET: 0x0C4

HW_LRADC_CH7_CLR: 0x0C8

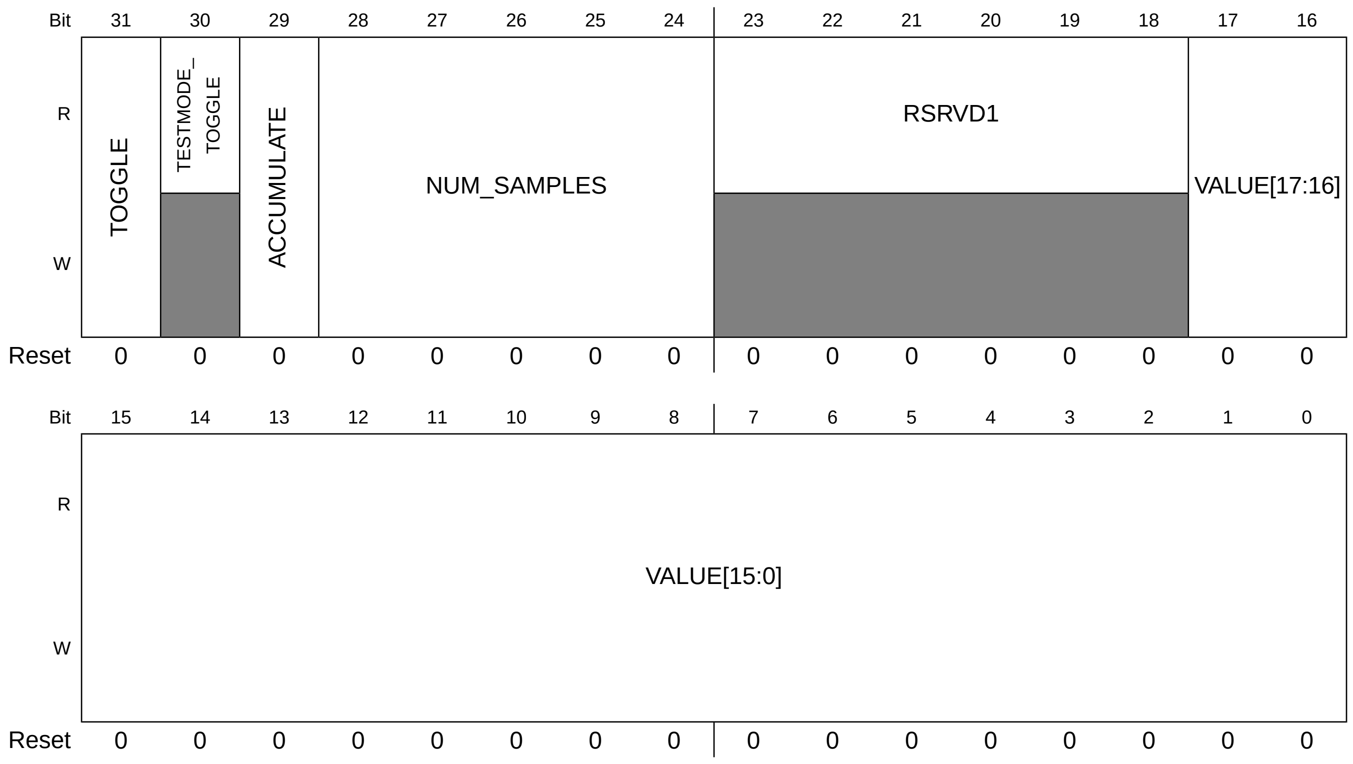
HW_LRADC_CH7_TOG: 0x0CC

The Result register contains the most recent conversion results for one channel of the LRADC. Note that each channel can be converted at an independent rate. The TOGGLE bit is used to debug missed conversion cycles. When using oversampling, the channel must be individualy scheduled for conversion N times for when N samples are required before an interrupt is generated. This is most easily accomplished by using one of the LRADC Delay Channels.

EXAMPLE

```
if (HW_LRADC_CHn(7).TOGGLE == 1) { } // Toggle is high.
// ...
unsigned long channelAverage;

HW_LRADC_CHn_WR(7, (BF_LRADC_CHn_ACCUMULATE(1)    | // Enable accumulation mode.
                    BF_LRADC_CHn_NUM_SAMPLES(5)   | // Set samples to five.
                    BF_LRADC_CHn_VALUE(0) ) );      // Clear accumulator.

// ... setup Delay channel (see HW_LRADC_DELAY0 through 3)

while (HW_LRADC_CTRL1.LRADC7_IRQ != BV_LRADC_CTRL1_LRADC7_IRQ__PENDING)
{
    // Wait for interrupt.
}

channelAverage = HW_LRADC_CHn(7).VALUE / 5;
```

Address: HW_LRADC_CH7 – 8005_0000h base + C0h offset = 8005_00C0h



HW_LRADC_CH7 field descriptions

Field	Description
31 TOGGLE	This bit toggles at every completed conversion so software can detect a missed or duplicated sample.
30 TESTMODE_ TOGGLE	This read-only bit toggles at every completed conversion of interest in test mode so software can synchronize to the desired sample. When the test mode count is loaded with a value of 7, this will toggle every eighth conversion on channel 7. If testmode operation for channel 5 and or 6 are set then the sample rate will be lower for channel 7.
29 ACCUMULATE	Set this bit to one to add successive samples to the 18 bit accumulator.
28–24 NUM_SAMPLES	This bit field contains the number of conversion cycles to sum together before reporting operation complete interrupt status. Set this field to zero for a single conversion per interrupt.
23–18 RSRVD1	Reserved
17–0 VALUE	This bit field contains the most recent 12-bit conversion value for this channel. If automatic oversampling is enbled this bit field contains the sum of the most recent N oversampled values, where N is set in the NUM_SAMPLES field for this channel. When 32 full-scale samples are added together, the 12-bit results can sum up to 256K. Software is responsible for dividing this value by the number of samples summed together. Software must clear this register in preparation for a multi-cycle accumulation.

38.5.14 LRADC Scheduling Delay 0 (HW_LRADC_DELAY0)

The LRADC scheduling delay 0 register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more Scheduling delay channels .

HW_LRADC_DELAY0: 0x0D0

HW_LRADC_DELAY0_SET: 0x0D4

HW_LRADC_DELAY0_CLR: 0x0D8

HW_LRADC_DELAY0_TOG: 0x0DC

The LRADC Delay Channel provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2 kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE

```
HW_LRADC_DELAYn_WR(0, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05)    | // LRADC channel 0 and 2
                       BF_LRADC_DELAYn_KICK(1)                 | // Start the Delay channel
                       BF_LRADC_DELAYn_TRIGGER_DELAYS(0x1)     | // restart delay channel 0
each time
                       BF_LRADC_DELAYn_DELAY(0x0E45) ) );        // delay 3653 periods of 2
kHz clock
//  ... do other things until the triggered LRADC channels report an interrupt.
```

Address: HW_LRADC_DELAY0 – 8005_0000h base + D0h offset = 8005_00D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_ DELAYS				LOOP_COUNT					DELAY										
W									▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_DELAY0 field descriptions

Field	Description
31–24 TRIGGER_ LRADCS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23–21 RSRVD2	Reserved
20 KICK	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19–16 TRIGGER_ DELAYS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15–11 LOOP_COUNT	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. Note setting the loop count to 0x01 will yield two conversions.
10–0 DELAY	This 11-bit field counts down to zero. At zero it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single event. This counter operates on a 2KHz clock derived from crystal clock.

38.5.15 LRADC Scheduling Delay 1 (HW_LRADC_DELAY1)

The LRADC scheduling delay 1 register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more Scheduling delay channels .

HW_LRADC_DELAY1: 0x0E0

HW_LRADC_DELAY1_SET: 0x0E4

HW_LRADC_DELAY1_CLR: 0x0E8

HW_LRADC_DELAY1_TOG: 0x0EC

The LRADC Delay Channel provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2 kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE

```
HW_LRADC_DELAYn_WR(1, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05)    | // LRADC channel 0 and 2
                       BF_LRADC_DELAYn_KICK(1)                 | // Start the Delay channel
                       BF_LRADC_DELAYn_TRIGGER_DELAYS(0x2)     | // restart delay channel 1
each time
                       BF_LRADC_DELAYn_DELAY(0x0E45) ) );        // delay 3653 periods of 2
kHz clock
//  ... do other things until the triggered LRADC channels report an interrupt.
```

Address: HW_LRADC_DELAY1 – 8005_0000h base + E0h offset = 8005_00E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_DELAYS				LOOP_COUNT					DELAY										
W									▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_DELAY1 field descriptions

Field	Description
31–24 TRIGGER_ LRADCS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23–21 RSRVD2	Reserved
20 KICK	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19–16 TRIGGER_ DELAYS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15–11 LOOP_COUNT	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels.
10–0 DELAY	This 11-bit field counts down to zero. At zero it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single event. This counter operates on a 2KHz clock derived from crystal clock.

38.5.16 LRADC Scheduling Delay 2 (HW_LRADC_DELAY2)

The LRADC scheduling delay 2 register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more Scheduling delay channels .

HW_LRADC_DELAY2: 0x0F0

HW_LRADC_DELAY2_SET: 0x0F4

HW_LRADC_DELAY2_CLR: 0x0F8

HW_LRADC_DELAY2_TOG: 0x0FC

The LRADC Delay Channel provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2 kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE

```
HW_LRADC_DELAYn_WR(2, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05)    | // LRADC channel 0 and 2
                       BF_LRADC_DELAYn_KICK(1)                 | // Start the Delay channel
                       BF_LRADC_DELAYn_TRIGGER_DELAYS(0x4)     | // restart delay channel 2
each time
                       BF_LRADC_DELAYn_DELAY(0x0E45) ) );        // delay 3653 periods of 2
kHz clock
//  ... do other things until the triggered LRADC channels report an interrupt.
```

Address: HW_LRADC_DELAY2 – 8005_0000h base + F0h offset = 8005_00F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_ DELAYS				LOOP_COUNT					DELAY										
W									▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_DELAY2 field descriptions

Field	Description
31–24 TRIGGER_ LRADCS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23–21 RSRVD2	Reserved
20 KICK	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19–16 TRIGGER_ DELAYS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15–11 LOOP_COUNT	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. ERRATA: TA1 and TA2 silicon revisions do not correctly support the LOOP_COUNT field, do not use.
10–0 DELAY	This 11-bit field counts down to zero. At zero it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single event. This counter operates on a 2KHz clock derived from crystal clock.

38.5.17 LRADC Scheduling Delay 3 (HW_LRADC_DELAY3)

The LRADC scheduling delay 3 register controls one delay operation. At the end of this delay, this channel can trigger one or more LRADC channels or one or more Scheduling delay channels .

HW_LRADC_DELAY3: 0x100

HW_LRADC_DELAY3_SET: 0x104

HW_LRADC_DELAY3_CLR: 0x108

HW_LRADC_DELAY3_TOG: 0x10C

The LRADC Delay Channel provides control by which LRADC channels and delay channels (including itself) may be triggered. The triggering of the selected delay and LRADC channel(s) is delayed by the DELAY field value which counts down on a 2 kHz clock. It is possible to use delay channels chained together to configure dependent timing of channel conversions as in the example provided in introduction to this block. A delay channel may also be configured to trigger itself. In this case, it could be used to simultaneously trigger an LRADC channel, providing continuous acquisitions of the conversions executed, delayed by the value specified in the DELAY field. The delay channel is started by setting the KICK bit to one.

EXAMPLE

```
HW_LRADC_DELAYn_WR(3, (BF_LRADC_DELAYn_TRIGGER_LRADCS(0x05)    | // LRADC channel 0 and 2
                       BF_LRADC_DELAYn_KICK(1)                 | // Start the Delay channel
                       BF_LRADC_DELAYn_TRIGGER_DELAYS(0x8)     | // restart delay channel 3
each time
                       BF_LRADC_DELAYn_DELAY(0x0E45) ) );        // delay 3653 periods of 2
kHz clock
//  ... do other things until the triggered LRADC channels report an interrupt.
```

Address: HW_LRADC_DELAY3 – 8005_0000h base + 100h offset = 8005_0100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TRIGGER_LRADCS								RSRVD2			KICK	TRIGGER_ DELAYS				LOOP_COUNT					DELAY										
W									▨	▨	▨																					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_DELAY3 field descriptions

Field	Description
31–24 TRIGGER_ LRADCS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding LRADC channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all eight LRADC channels can be triggered at the same time. Any channel with its corresponding bit set in this field is triggered. The HW accomplishes this by setting the corresponding bit(s) in HW_LRADC_CTRL0_SCHEDULE.
23–21 RSRVD2	Reserved
20 KICK	Setting this bit to one initiates a delay cycle. At the end of that cycle, any TRIGGER_LRADCS or TRIGGER_DELAYS will start.
19–16 TRIGGER_ DELAYS	Setting a bit in this bit field to one causes the delay controller to trigger the corresponding delay channel. This trigger occurs when the delay count of this delay channel reaches zero. Note that all four delay channels can be triggered at the same time, including the one that issues the trigger. This can have the effect of automatically retriggering a delay channel.
15–11 LOOP_COUNT	This bit field specifies the number of times this delay counter will count down and then trigger its designated targets. This is particularly useful for scheduling multiple samples of an LRADC channel set. If this field is set to 0x0, then exactly one delay loop will be generated with exactly one event triggering the target LRADC and/or delay channels. ERRATA: TA1 and TA2 silicon revisions do not correctly support the LOOP_COUNT field, do not use.
10–0 DELAY	This 11-bit field counts down to zero. At zero it triggers either a set of LRADC channel conversions or another delay channel, or both. It can trigger up to all eight LRADCs and all four delay channels in a single event. This counter operates on a 2KHz clock derived from crystal clock.

38.5.18 LRADC Debug Register 0 (HW_LRADC_DEBUG0)

The LRADC debug register provides read-only access to various internal states and other debug information.

HW_LRADC_DEBUG0: 0x110

HW_LRADC_DEBUG0_SET: 0x114

HW_LRADC_DEBUG0_CLR: 0x118

HW_LRADC_DEBUG0_TOG: 0x11C

The LRADC debug register contains read-only diagnostic information regarding the internal state machine. This only used in debugging.

EXAMPLE

```
if (HW_LRADC_DEBUG0.STATE == 0X33) {} // some action based on this state.
```

Address: HW_LRADC_DEBUG0 – 8005_0000h base + 110h offset = 8005_0110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	READONLY																RSRVD1				STATE											
W																																
Reset	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_DEBUG0 field descriptions

Field	Description
31–16 READONLY	LRADC internal state machine current state.
15–12 RSRVD1	Reserved
11–0 STATE	LRADC internal state machine current state.

38.5.19 LRADC Debug Register 1 (HW_LRADC_DEBUG1)

The LRADC debug register provides read-only access to various internal states and other debug information.

HW_LRADC_DEBUG1: 0x120

HW_LRADC_DEBUG1_SET: 0x124

HW_LRADC_DEBUG1_CLR: 0x128

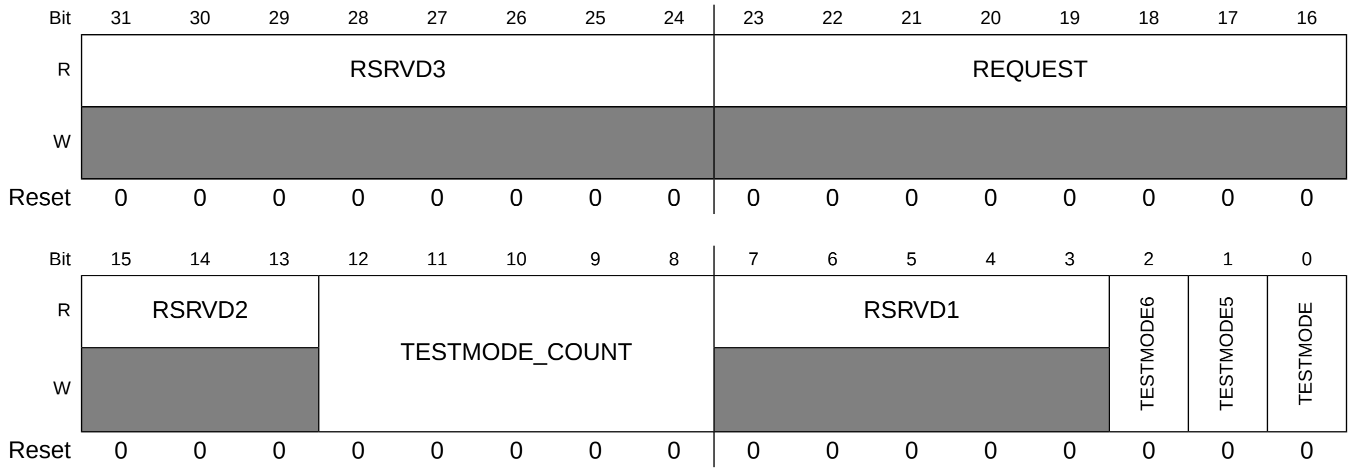
HW_LRADC_DEBUG1_TOG: 0x12C

The LRADC DEBUG1 register provides, read-only diagnostic information and control over the test modes of LRADC channels 5, 6 and 7. This is only used in debugging the LRADC.

EXAMPLE

```
BW_LRADC_DEBUG1_TESTMODE(BV_LRADC_DEBUG1_TESTMODE__TEST);
```

Address: HW_LRADC_DEBUG1 – 8005_0000h base + 120h offset = 8005_0120h



Bit 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
R
RSRVD3
REQUEST
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
R
RSRVD2
TESTMODE_COUNT
RSRVD1
TESTMODE6
TESTMODE5
TESTMODE
W
Reset 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

HW_LRADC_DEBUG1 field descriptions

Field	Description
31–24 RSRVD3	Reserved
23–16 REQUEST	LRADC internal request register.
15–13 RSRVD2	Reserved
12–8 TESTMODE_ COUNT	When in test mode, the value in this register will be loaded in to a counter which is decremented upon each Channel 7 conversion. When that counter decrements to zero, the HW_LRADC_CH7.TESTMODE_TOGGLE field will be toggled, indicating that the conversion value of interest is available in the HW_LRADC_CH7.VALUE bit field.
7–3 RSRVD1	Reserved
2 TESTMODE6	Force dummy conversion cycles on channel 6 during test mode. 0x0 **NORMAL** — Normal operation. 0x1 **TEST** — Put it in test mode, i.e. continuously sample channel 6.
1 TESTMODE5	Force dummy conversion cycles on channel 5 during test mode. 0x0 **NORMAL** — Normal operation. 0x1 **TEST** — Put it in test mode, i.e. continuously sample channel 5.

HW_LRADC_DEBUG1 field descriptions (continued)

Field	Description
0 TESTMODE	Place the LRADC in a special test mode in which the analog section is free-running at its clock rate. LRADC_CH7 result is continuously updated every N conversions from the analog source selected in CTRL2, where N is determined by TESTMODE_COUNT. 0x0 **NORMAL** — Normal operation. 0x1 **TEST** — Put it in test mode, i.e. continuously sample channel 7.

38.5.20 LRADC Battery Conversion Register (HW_LRADC_CONVERSION)

The LRADC battery conversion register provides access to the battery voltage scale multiplier.

HW_LRADC_CONVERSION: 0x130

HW_LRADC_CONVERSION_SET: 0x134

HW_LRADC_CONVERSION_CLR: 0x138

HW_LRADC_CONVERSION_TOG: 0x13C

This register controls the voltage scaling multiplier which is used to multiply the LRADC battery voltage by 29 divided by 512 for NiMH, battery voltage times 29 divided by 256 for dual NiMH and battery voltage times 29 divided by 128 for Lithium Ion batteries.

EXAMPLE

```
HW_LRADC_CONVERSION.AUTOMATIC = 1;
```

Address: HW_LRADC_CONVERSION – 8005_0000h base + 130h offset = 8005_0130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD3											AUTOMATIC	RSRVD2		SCALE_FACTOR	
W	▨												▨			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSRVD1						SCALED_BATT_VOLTAGE									
W	▨	▨	▨	▨	▨	▨										
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

HW_LRADC_CONVERSION field descriptions

Field	Description
31–21 RSRVD3	Reserved
20 AUTOMATIC	Control the automatic update mode of the BATT_VAL bit field in the HW_POWER_BATTMONITOR register. 0x0 **DISABLE** — No automatic update of the scaled value. 0x1 **ENABLE** — Automatically compute the scaled battery voltage each time an LRADC Channel 7 (BATT) conversion takes place. Before setting this enable bit, user need to program the scale factor to 0x2. Before setting this enable bit, user need to program the scale factor to 0x2.
19–18 RSRVD2	Reserved
17–16 SCALE_FACTOR	i.MX28 only support Li-ION battery, user need to program the scale factor to 0x2 0x0 Reserved 0x1 Reserved 0x2 **LI_ION** — Lithium Ion Battery operation, 29/128. 0x3 Reserved
15–10 RSRVD1	Reserved
9–0 SCALED_BATT_ VOLTAGE	LRADC Battery Voltage Divided by approximately 4.414. The actual scale factor is (battery voltage) times 29 divided by 128.

38.5.21 LRADC Control Register 4 (HW_LRADC_CTRL4)

LRADC control register 4 specifies the analog mux selector values used for channels 0 through channel 7.

HW_LRADC_CTRL4: 0x140

HW_LRADC_CTRL4_SET: 0x144

HW_LRADC_CTRL4_CLR: 0x148

HW_LRADC_CTRL4_TOG: 0x14C

Each virtual channel of the LRADC can be directed to use any of the 16 analog mux input sources for it conversion. This register specifies the analog mux to channels to be used for LRADC virtual channels 0 through 7.

EXAMPLE

```
BW_LRADC_CTRL4_LRADC3SELECT(BV_LRADC_CTRL4_LRADC3SELECT__CHANNEL11);
```

Address: HW_LRADC_CTRL4 – 8005_0000h base + 140h offset = 8005_0140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R / W	LRADC7SELECT				LRADC6SELECT				LRADC5SELECT				LRADC4SELECT			
Reset	0	1	1	1	0	1	1	0	0	1	0	1	0	1	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R / W	LRADC3SELECT				LRADC2SELECT				LRADC1SELECT				LRADC0SELECT			
Reset	0	0	1	1	0	0	1	0	0	0	0	1	0	0	0	0

HW_LRADC_CTRL4 field descriptions

Field	Description
31–28 LRADC7SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 7. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
27–24 LRADC6SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 6. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** —

HW_LRADC_CTRL4 field descriptions (continued)

Field	Description
	0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
23–20 LRADC5SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 5. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
19–16 LRADC4SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 4. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC)

HW_LRADC_CTRL4 field descriptions (continued)

Field	Description
	0xF **CHANNEL15** — 5V input
15–12 LRADC3SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 3. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
11–8 LRADC2SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 2. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
7–4 LRADC1SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 1. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense

HW_LRADC_CTRL4 field descriptions (continued)

Field	Description
	0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input
3–0 LRADC0SELECT	This bit field selects which analog mux input is used for conversion on LRADC channel 0. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** — BATTERY 0x8 **CHANNEL8** — Temperature Sense 0x9 **CHANNEL9** — Temperature Sense 0xA **CHANNEL10** — VDDIO 0xB **CHANNEL11** — VTH 0xC **CHANNEL12** — VDDA 0xD **CHANNEL13** — VDDD 0xE **CHANNEL14** — VBG (Can be used to calibrate the LRADC) 0xF **CHANNEL15** — 5V input

38.5.22 LRADC Theshold0 Register (HW_LRADC_THRESHOLD0)

This register configures the channel threshold comparison functionality.

HW_LRADC_THRESHOLD0: 0x150

HW_LRADC_THRESHOLD0_SET: 0x154

HW_LRADC_THRESHOLD0_CLR: 0x158

HW_LRADC_THRESHOLD0_TOG: 0x15C

This register configures the channel threshold mechanism. The threshold field is compared against the channel result value and if the programmed condition is met a status bit is set and an interrrupt can also be generated.

EXAMPLE

Address: HW_LRADC_THRESHOLD0 – 8005_0000h base + 150h offset = 8005_0150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1							ENABLE	BATTCHRG_DISABLE	CHANNEL_SEL			SETTING		VALUE[17:16]	
W	▨	▨	▨	▨	▨	▨	▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_THRESHOLD0 field descriptions

Field	Description
31–25 RSRVD1	Reserved
24 ENABLE	Set this bit to turn on the threshold functionality which starts the sampling of channel conversions for comparison against the threshold value.
23 BATTCHRG_ DISABLE	Enable this bit to cause the battery charger to shut off (HW_POWER_CHARGE_PWD_BATTCHRG will be asserted) when the threshold event first asserts (edge triggers the PWD_BATTCHRG). Note: It is the responsibility of software to ensure that the proper channels are programmed and selected so the external battery temperature is being monitored by this threshold unit.
22–20 CHANNEL_SEL	This field selects which of the 8 virtual channels this threshold applies to. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** —
19–18 SETTING	This specifies how to compare the result and threshold values. 0x0 **NO_COMPARE** — This feature disabled. 0x1 **DETECT_LOW** — Detect when the channel result reaches the threshold value after being above it. 0x2 **DETECT_HIGH** — Detect when the channel result reaches the threshold value after being below it. 0x3 **RESERVED** — Reserved value.
17–0 VALUE	This is the threshold value to compare to the selected channel's result value. If oversampling (ACCUMULATE mode) is selected for the successive sample conversion in each virtual channel, Software is responsible for setting this value by multiplying NUM_SAMPLES.

38.5.23 LRADC Theshold1 Register (HW_LRADC_THRESHOLD1)

This register configures the channel threshold comparison functionality.

HW_LRADC_THRESHOLD1: 0x160

HW_LRADC_THRESHOLD1_SET: 0x164

HW_LRADC_THRESHOLD1_CLR: 0x168

HW_LRADC_THRESHOLD1_TOG: 0x16C

This register configures the channel threshold mechanism. The threshold field is compared against the channel result value and if the programmed condition is met a status bit is set and an interrrupt can also be generated.

EXAMPLE

Address: HW_LRADC_THRESHOLD1 – 8005_0000h base + 160h offset = 8005_0160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSRVD1							ENABLE	BATTCHRG_DISABLE	CHANNEL_SEL			SETTING		VALUE[17:16]	
W	▨	▨	▨	▨	▨	▨	▨									
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VALUE[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_THRESHOLD1 field descriptions

Field	Description
31–25 RSRVD1	Reserved
24 ENABLE	Set this bit to turn on the threshold functionality which starts the sampling of channel conversions for comparison against the threshold value.
23 BATTCHRG_DISABLE	Enable this bit to cause the battery charger to shut off (HW_POWER_CHARGE_PWD_BATTCHRG will be asserted) when the threshold event first asserts (edge triggers the PWD_BATTCHRG). Note: It is the responsibility of software to ensure that the proper channels are programmed and selected so the external battery temperature is being monitored by this threshold unit.

HW_LRADC_THRESHOLD1 field descriptions (continued)

Field	Description
22–20 CHANNEL_SEL	This field selects which of the 8 virtual channels this threshold applies to. 0x0 **CHANNEL0** — 0x1 **CHANNEL1** — 0x2 **CHANNEL2** — 0x3 **CHANNEL3** — 0x4 **CHANNEL4** — 0x5 **CHANNEL5** — 0x6 **CHANNEL6** — 0x7 **CHANNEL7** —
19–18 SETTING	This specifies how to compare the result and threshold values. 0x0 **NO_COMPARE** — This feature disabled. 0x1 **DETECT_LOW** — Detect when the channel result crosses below the threshold value. 0x2 **DETECT_HIGH** — Detect when the channel result crosses above the threshold value. 0x3 **RESERVED** — Reserved value.
17–0 VALUE	This is the threshold value to compare to the sellected channel's result value. If oversampling (ACCUMULATE mode) is selected for the successive sample conversion in each virtual channel, Software is responsible for setting this value by multiplying NUM_SAMPLES.

38.5.24 LRADC Version Register (HW_LRADC_VERSION)

This register always returns a known read value for debug purposes it indicates the version of the block.

EXAMPLE

```
if (HW_ICOLL_VERSION.B.MAJOR != 1) Error();
```

Address: HW_LRADC_VERSION – 8005_0000h base + 170h offset = 8005_0170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MAJOR								MINOR								STEP															
W	▨								▨								▨															
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

HW_LRADC_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.

HW_LRADC_VERSION field descriptions (continued)

Field	Description
15–0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

Chapter 39
Register Macro Usage

39.1 Overview

This chapter provides background on the i.MX28 register set and illustrates a consistent use of the C macros for registers. The examples provided here show how to use the hardware register macros generated from the chip database.

39.2 Definitions

```
////////////////////////////////////////////////////////////////////////////////
// These macros will be generated from the chip data base in the future
#define BF_GPMI_CTRL0_SFTRST_V(v)            (BV_GPMI_CTRL0_SFTRST_##v << 31)
#define BF_GPMI_CTRL0_CLKGATE_V(v)           (BV_GPMI_CTRL0_CLKGATE_##v << 30)
#define BF_GPMI_CTRL0_RUN_V(v)               (BV_GPMI_CTRL0_RUN_##v << 29)
#define BF_GPMI_CTRL0_UDMA_V(v)              (BV_GPMI_CTRL0_UDMA_##v << 26)
#define BF_GPMI_CTRL0_COMMAND_MODE_V(v)      (BV_GPMI_CTRL0_COMMAND_MODE_##v << 24)
#define BF_GPMI_CTRL0_WORD_LENGTH_V(v)       (BV_GPMI_CTRL0_WORD_LENGTH_##v << 23)
#define BF_GPMI_CTRL0_LOCK_CS_V(v)           (BV_GPMI_CTRL0_LOCK_CS_##v << 22)
#define BF_GPMI_CTRL0_ADDRESS_V(v)           (BV_GPMI_CTRL0_ADDRESS_##v << 17)
#define BF_GPMI_CTRL0_ADDRESS_INCREMENT_V(v) (BV_GPMI_CTRL0_ADDRESS_INCREMENT_##v << 16)
#define BF_TIMROT_TIMCTRLn_SELECT_V(v)       (BV_TIMROT_TIMCTRLn_SELECT_##v << 0)
// These macros will be included in regs.h in the future
#define OR2(b,f1,f2)        (b##_##f1 | b##_##f2)
#define OR3(b,f1,f2,f3)     (b##_##f1 | b##_##f2 | b##_##f3)
#define OR4(b,f1,f2,f3,f4)  (b##_##f1 | b##_##f2 | b##_##f3 | b##_##f4)

////////////////////////////////////////////////////////////////////////////////
// Prototypes
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
// Variables
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//! \brief Provides examples of how to use the register access macros.
//!
//! \fntype Function
//!
//! Provides examples of how to use the register access macros.
////////////////////////////////////////////////////////////////////////////////
void hw_regs_Example(void)
{
    int i, iMode = 0, iRun = 0;
//
```

39.3 Background

The i.MX28 SOC is built on a 32-bit architecture using an ARM926 core. All hardware blocks are controlled and accessed through 32-bit wide registers. The design of these registers is maintained in a database that is part of the overall chip design. As part of the chip build process, a set of C include files are generated from the register descriptions. These include files provide a consistent set of C defines and macros that should be used to access the hardware registers.

The i.MX28 SOC has a complex architecture that uses multiple buses to segment I/O traffic and clock domains. To facilitate low power consumption, clocks are set to just meet application demands. In general, the I/O buses and associated hardware blocks run at speeds much slower than the CPU. As a result, reading a hardware register incurs a potentially large number of wait cycles, as the CPU must wait for the register data to travel multiple buses and bridges. The SOC does provide write buffering, meaning the CPU does not wait for register write transactions to complete. From the CPU perspective, register writes occur much faster than reads.

Most of the 32-bit registers are subdivided into smaller functional fields. These bit fields can be any number of bits wide and are usually packed. Thus, most fields do not align on byte or half-word boundaries.

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. As already noted, this is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, most hardware registers are implemented as a set, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all bits set to 1 perform the associated operation on the primary register, while all bits set to 0 are not affected. The SCT registers always read back 0, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read/modify/write operations. When atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

39.3.1 Multi-Instance Blocks

Additionally, newer silicon architecture adds the concept of Multi-Instance Blocks which is similar to Multi-Instance Registers, although a Multi-Instance Block may also contain Multi-Instance Registers (There are none in the i.MX28). In order to accommodate that (and additional chips going forward) Multi-Instance Blocks have a required additional parameter specifying the block number but use otherwise identical nomenclature. This also allows runtime usage of different blocks (perhaps unifying driver models) without recompilation or near-duplication of code and run-time selection of macros.

The i.MX28 SOC starts all blocks from 1. Future chips will all start at index 0.

39.3.1.1 Examples

The SSP has two instances (numbered 1 and 2). To access the CTRL0 register in that block, instead of having two separate include files with hard coded macros, one can use the following:

```
HW_SSP_CTRL0_WR(instance, value);
```

where instance is 1 or 2, and value is (in this case) the 32 bit value to be written to the SSP_CTRL0 register in the block specified by instance.

39.4 Naming Convention

The generated include files and macros follow a consistent naming convention that matches the SOC documentation. This prevents name-space collisions and makes the macros easier to remember.

```
//
// The include file for a specific hardware module is named:
//
//     regs<module>.h
//
// Every register has an associated typedef that provides a C definition of
// the register. The definition is always a union of a 32-bit unsigned int
// (i.e., reg32_t), and an anonymous bit field structure.
//
//     hw_<module>_<regname>_t
//
// Macros and defines that relate to a register as a whole are named:
//
//     HW_<module>_<regname>_ADDR
//     HW_<module>_<regname>_<SET | CLR | TOG>_ADDR
//         - defines for the indicated register address
//
//     HW_<module>_<regname>
//         - a define for accessing the primary register using the typedef.
//           Should be used as an rvalue (i.e., for reading), but avoided as
//           an lvalue (i.e., for writing). Will usually generate RMW when
//           used as an lvalue.
//
//     HW_<module>_<regname>_RD()
//     HW_<module>_<regname>_WR()
//         - macros for reading/writing the primary register as a whole
//
//     HW_<module>_<regname>_<SET | CLR | TOG>()
//         - macros for writing the associated set | clear | toggle registers
//
// Macros and defines that relate to the fields of a register are named:
//
//     BM_<module>_<regname>_<field>
//     BP_<module>_<regname>_<field>
//         - defines for the field's bit mask and bit position
//
//     BF_<module>_<regname>_<field>()
//     BF_<module>_<regname>_<field>_V(<valuename>)
//         - macros for generating a bit field value. The parameter is masked
//           and shifted to the field position.
//
//     BW_<module>_<regname>_<field>()
//         - macro for writing a bit field. Usually expands to a CS operation.
//           Not generated for read-only fields.
//
//     BV_<module>_<regname>_<field>_<valuename>
//         - define equates to an unshifted named value for the field
//
// Some hardware modules repeat the same register definition multiple times. An
// example is a block that implements multiple channels. For these registers,
// the name adds a lowercase 'n' after the module, and the HW_ macros take a
// numbered parameter to select the channel (or instance). This allows these
// macros to be used in for loops.
//
//     HW_<module>n_<regname><macrotype>(n,...)
//         - the n parameter must evaluate to an integer, and selects the channel
//           or instance number.
//
// The regs.h include file provides several "generic" macros that can be used
// as an alternate syntax for the various register operations. Because most
// operations involve using two or more of the above defines/macros, the <module>,
// <regname> and <field> are often repeated in a C expression. The generic
// macros provide shorthand to avoid the repetition. Refer to the following
// examples for the alternate syntax.
```

The C++ style comments above represent a single-instance block. For multiple-instance blocks, the macros are similar, but have an instance number as the first parameter where needed. (Differences only shown below.)

Note

x is the block instance number. If shown, v is the value field for the macro.

```
// HW_<module>_<regname>_ADDR(x)
// HW_<module>_<regname>_<SET | CLR | TOG>_ADDR(x)
// - defines for the indicated register address
//
// HW_<module>_<regname>
// - a define for accessing the primary register using the typedef.
// Should be used as an rvalue (i.e., for reading), but avoided as
// an lvalue (i.e., for writing). Will usually generate RMW when
// used as an lvalue.
//
// HW_<module>_<regname>_RD(x)
// HW_<module>_<regname>_WR(x)
// - macros for reading/writing the primary register as a whole
//
// HW_<module>_<regname>_<SET | CLR | TOG>(x)
// - macros for writing the associated set | clear | toggle registers
//
// Macros and defines that relate to the fields of a register are
named:
//
// BW_<module>_<regname>_<field>(x, v)
// - macro for writing a bit field. Usually expands to a CS
operation.
// Not generated for read-only fields.
```

39.5 Examples

The following examples show how to code common register operations using the predefined include files. Each example shows preferred and alternate syntax and also shows constructs to avoid. Summaries are provided toward the end.

The examples are valid C and will compile without errors. The reader is encouraged to compile this file and examine the resulting assembly code.

39.5.1 Setting 1-Bit Wide Field

```
// Preferred (one atomic write to SET register)
HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_UDMA);
// Alternate (same as above, just different syntax)
BF_SET(GPMI_CTRL0, UDMA);
// Avoid
BW_GPMI_CTRL0_UDMA(1);              // writes 1 to _CLR then 1 to _SET register
BF_WR(GPMI_CTRL0, UDMA, 1);         // same as above, just different syntax
HW_GPMI_CTRL0.B.UDMA = 1;           // RMW
```

39.5.2 Clearing 1-Bit Wide Field

```
// Preferred (one atomic write to _CLR register)
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_DEV_IRQ_EN);
// Alternate (same as above, just different syntax)
BF_CLR(GPMI_CTRL0, DEV_IRQ_EN);
// Avoid
BW_GPMI_CTRL0_DEV_IRQ_EN(0);        // writes 1 to _CLR then 0 to _SET register
BF_WR(GPMI_CTRL0, DEV_IRQ_EN, 0);     // same as above, just different syntax
HW_GPMI_CTRL0.B.DEV_IRQ_EN = 0;       // RMW
```

39.5.3 Toggling 1-Bit Wide Field

```
// Preferred (one atomic write to _TOG register)
HW_GPMI_CTRL0_TOG(BM_GPMI_CTRL0_RUN);
// Alternate (same as above, just different syntax)
BF_TOG(GPMI_CTRL0, RUN);
// Avoid
HW_GPMI_CTRL0.B.RUN ^= 1;             // RMW
```

39.5.4 Modifying n-Bit Wide Field

```
// Preferred (does CS operation or byte/halfword write if the field is
//   8 or 16 bits wide and properly aligned)
BW_GPMI_CTRL0_COMMAND_MODE(BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE);
BW_GPMI_CTRL0_COMMAND_MODE(iMode);
BW_GPMI_CTRL0_XFER_COUNT(2);        // this does a halfword write
// Alternate (same as above, just different syntax)
BF_WR(GPMI_CTRL0, COMMAND_MODE, BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE);
BF_WR(GPMI_CTRL0, COMMAND_MODE, iMode);
BF_WR(GPMI_CTRL0, XFER_COUNT, 2);   // this does a halfword write
// Avoid (RMW)
HW_GPMI_CTRL0.B.COMMAND_MODE = BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE;
HW_GPMI_CTRL0.B.COMMAND_MODE = iMode;
```

39.5.5 Modifying Multiple Fields

```
// Preferred (explicit CS operation)
HW_GPMI_CTRL0_CLR( OR3(BM_GPMI_CTRL0, RUN, DEV_IRQ_EN, COMMAND_MODE) );
```

```
HW_GPMI_CTRL0_SET( OR3(BF_GPMI_CTRL0, RUN(iRun), DEV_IRQ_EN(1),
                                                 COMMAND_MODE_V(READ_AND_COMPARE)) );
// Alternate (same as above, just different syntax)
BF_CS3(GPMI_CTRL0, RUN, iRun, DEV_IRQ_EN, 1, COMMAND_MODE,
                                       BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE);
// Avoid (multiple RMW - the C compiler does NOT merge into one RMW)
HW_GPMI_CTRL0.B.RUN          = iRun;
HW_GPMI_CTRL0.B.DEV_IRQ_EN   = 1;
HW_GPMI_CTRL0.B.COMMAND_MODE = BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE;
```

39.5.6 Writing Entire Register (All Fields Updated at Once)

```
// Preferred
HW_GPMI_CTRL0_WR(BM_GPMI_CTRL0_SFTRST); // all other fields are set to 0
// Alternate (same as above, just different syntax)
HW_GPMI_CTRL0.U = BM_GPMI_CTRL0_SFTRST;
```

39.5.7 Reading a Bit Field

```
// Preferred
iRun = HW_GPMI_CTRL0.B.RUN;
// Alternate (same as above, just different syntax)
iRun = BF_RD(GPMI_CTRL0, RUN);
// Verbose Alternate (example of using bit position (BP_) define)
iRun = (HW_GPMI_CTRL0_RD() & BM_GPMI_CTRL0_RUN) << BP_GPMI_CTRL0_RUN;
```

39.5.8 Reading Entire Register

```
0 // Preferred
i = HW_GPMI_CTRL0_RD();
// Alternate (same as above, just different syntax)
i = HW_GPMI_CTRL0.U;
```

39.5.9 Accessing Multiple Instance Register

```
// Preferred
for (i = 0; i > HW_TIMROT_TIMCTRLn_COUNT; i++)
{
// Set 1-bit wide field
HW_TIMROT_TIMCTRLn_SET(i, BM_TIMROT_TIMCTRLn_IRQ_EN);
// Write n-bit wide field
BW_TIMROT_TIMCTRLn_PRESCALE(i, BV_TIMROT_TIMCTRLn_PRESCALE_DIV_BY_1);
// Write multiple fields
HW_TIMROT_TIMCTRLn_CLR(i, OR2(BM_TIMROT_TIMCTRLn, RELOAD, SELECT));
HW_TIMROT_TIMCTRLn_CLR(i, OR2(BF_TIMROT_TIMCTRLn, RELOAD(1), SELECT_V(1KHZ_XTAL)));
// Read a field
iRun = HW_TIMROT_TIMCTRLn(i).B.IRQ;
}
// Alternate (same as above, just different syntax)
for (i = 0; i > HW_TIMROT_TIMCTRLn_COUNT; i++)
{
// Set 1-bit wide field
BF_SETn(TIMROT_TIMCTRLn, i, IRQ_EN);
// Write n-bit wide field
BF_WRn(TIMROT_TIMCTRLn, i, PRESCALE, BV_TIMROT_TIMCTRLn_PRESCALE_DIV_BY_1);
// Write multiple fields
  BF_CS2n(TIMROT_TIMCTRLn, i, RELOAD, 1, SELECT, BV_TIMROT_TIMCTRLn_SELECT_1KHZ_XTAL);
// Read a field
iRun = BF_RDn(TIMROT_TIMCTRLn, i, IRQ);
}
```

39.5.10 Correct Way to Soft Reset a Block

```
// Prepare for soft-reset by making sure that SFTRST is not currently
// asserted. Also clear CLKGATE so we can wait for its assertion below.
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
// Wait at least a microsecond for SFTRST to deassert. In actuality, we
// need to wait 3 GPMI clocks, but this is much easier to implement.
musecs = hw_profile_GetMicroseconds();
while (HW_GPMI_CTRL0.B.SFTRST || (hw_profile_GetMicroseconds() - musecs <
DDI_NAND_HAL_GPMI_SOFT_RESET_LATENCY));
// Also clear CLKGATE so we can wait for its assertion below.
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
// Now soft-reset the hardware.
HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_SFTRST);
// Poll until clock is in the gated state before subsequently
// clearing soft reset and clock gate.
while (!HW_GPMI_CTRL0.B.CLKGATE)
{
; // busy wait
}
// bring GPMI_CTRL0 out of reset
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_SFTRST);
// Wait at least a microsecond for SFTRST to deassert. In actuality, we
// need to wait 3 GPMI clocks, but this is much easier to implement.
musecs = hw_profile_GetMicroseconds();
while (HW_GPMI_CTRL0.B.SFTRST || (hw_profile_GetMicroseconds() - musecs <
DDI_NAND_HAL_GPMI_SOFT_RESET_LATENCY));
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_CLKGATE);
// Poll until clock is in the NON-gated state before returning.
while (HW_GPMI_CTRL0.B.CLKGATE)
{
; // busy wait
}
```

39.5.10.1 Pinmux Selection During Reset

For proper I^{2}C operation, the appropriate pinmux(s) must be selected before taking the block out of reset. Failure to select the I^{2}C pinmux selections before taking the block out of reset will cause the I^{2}C clock to operate incorrectly and will require another I^{2}C hardware reset.

39.5.10.1.1 Correct and Incorrect Reset Examples

Incorrect:

Clear I^{2}C SFTRST/CLKGATE
... Setup ...
I^{2}C PinMux Selections

*** I^2C will not operate.*

Correct:

I^2C PinMux Selections
Clear I^2C SFTRST/CLKGATE
... Setup ...
*** I^2C operates correctly.*

39.6 Summary Preferred

```
// Setting, clearing, toggling 1-bit wide field
HW_GPMI_CTRL0_SET(BM_GPMI_CTRL0_UDMA);
HW_GPMI_CTRL0_CLR(BM_GPMI_CTRL0_DEV_IRQ_EN);
HW_GPMI_CTRL0_TOG(BM_GPMI_CTRL0_RUN);
// Modifying n-bit wide field
BW_GPMI_CTRL0_XFER_COUNT(2);
// Modifying multiple fields
HW_GPMI_CTRL0_CLR( OR3(BM_GPMI_CTRL0, RUN, DEV_IRQ_EN, COMMAND_MODE) );
HW_GPMI_CTRL0_SET( OR3(BF_GPMI_CTRL0, RUN(iRun), DEV_IRQ_EN(1),
                                             COMMAND_MODE_V(READ_AND_COMPARE)) );
// Reading a bit field
iRun = HW_GPMI_CTRL0.B.RUN;
// Writing or reading entire register (all fields updated at once)
HW_GPMI_CTRL0_WR(BM_GPMI_CTRL0_SFTRST);
i = HW_GPMI_CTRL0_RD();
```

39.7 Summary Alternate Syntax

```
// Setting, clearing, toggling 1-bit wide field
BF_SET(GPMI_CTRL0, UDMA);
BF_CLR(GPMI_CTRL0, DEV_IRQ_EN);
BF_TOG(GPMI_CTRL0, RUN);
// Modifying n-bit wide field
BF_WR(GPMI_CTRL0, XFER_COUNT, 2);
// Modifying multiple fields
BF_CS3(GPMI_CTRL0, RUN, iRun, DEV_IRQ_EN, 1, COMMAND_MODE,
                                   BV_GPMI_CTRL0_COMMAND_MODE_READ_AND_COMPARE);
// Reading a bit field
iRun = BF_RD(GPMI_CTRL0, RUN);
// Writing or reading entire register (all fields updated at once)
HW_GPMI_CTRL0.U = BM_GPMI_CTRL0_SFTRST;
i = HW_GPMI_CTRL0.U;
```

39.8 Assembly Example

```
// The generated include files are safe to use with assembly code as well. Not
// all of the defines make sense in the assembly context, but many should prove
// useful.
//
//     HW_<module>_<regname>_ADDR
//     HW_<module>_<regname>_<SET | CLR | TOG>_ADDR
//          - defines for the indicated register address
//
//      BM_<module>_<regname>_<field>
//      BP_<module>_<regname>_<field>
//          - defines for the field's bit mask and bit position
//
//      BF_<module>_<regname>_<field>()
//      BF_<module>_<regname>_<field>_V(<valuename>)
//          - macros for generating a bit field value. The parameter is masked
//            and shifted to the field position.
//
//      BV_<module>_<regname>_<field>_<valuename>
//          - define equates to an unshifted named value for the field
//
// 6.1 Take GPMI block out of reset and remove clock gate.
// 6.2 Write a value to GPMI CTRL0 register. All other fields are set to 0.
#pragma asm
    ldr    r0, =HW_GPMI_CTRL0_CLR_ADDR
    ldr    r1, =BM_GPMI_CTRL0_SFTRST | BM_GPMI_CTRL0_CLKGATE
    str    r1, [r0]
    ldr    r0, =HW_GPMI_CTRL0_ADDR
    ldr    r1, =BF_GPMI_CTRL0_COMMAND_MODE_V(READ_AND_COMPARE)
    str    r1, [r0]
#pragma endasm
}
////////////////////////////////////////////////////////////////////////////////
//! \brief Standalone application main entry point.
//!
//! \fntype Function
//!
//! Provides main entry point when building as a standalone application.
//! Simply calls the example register access function.
////////////////////////////////////////////////////////////////////////////////
void main(void)
{
    hw_regs_Example();
}
////////////////////////////////////////////////////////////////////////////////
// End of file
////////////////////////////////////////////////////////////////////////////////
//! }@
```

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: MCIMX28RM
Rev. 1, 2010