Encore! 32™ Series Microcontroller (Z32AN)

High Performance ARM9 SoC

Data Sheet

DS0200-003

www.zilog.com

WARNING: *DO NOT USE IN LIFE SUPPORT*

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer

 Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZILOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZILOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. The information contained within this document has been verified according to the general principles of electrical and mechanical engineering.

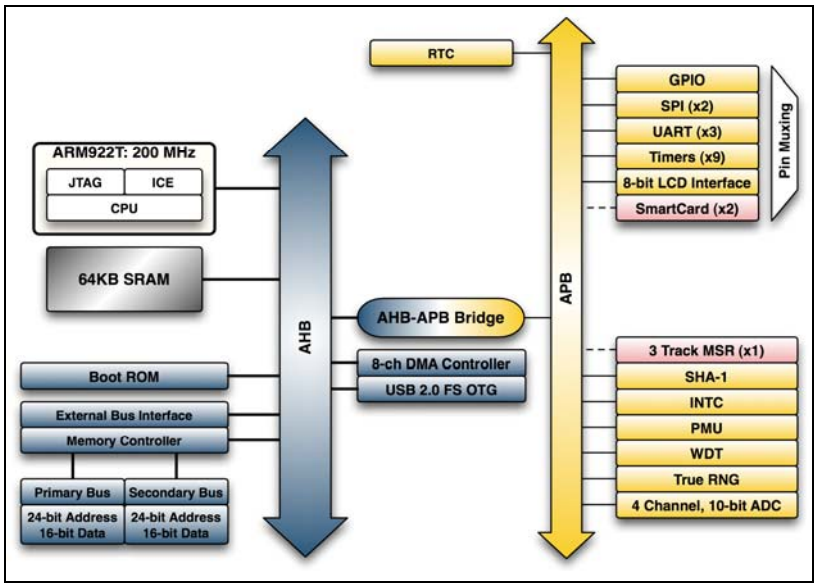
Encore! 32™ is a trademark of Zilog, Inc. ARM® and Thumb® are registered trademarks of ARM Limited in the European Union and other countries. All other product or service names are the property of their respective owners.

Revision History

Each instance in the revision history reflects a change to this document from its previous revision. For more details, refer to the corresponding pages and appropriate links in the table below.

Date	Revision Level	Description	Page Number
August 2008	001	Original issue	All
September 23, 2008	002	Updated part numbers	Chapter 23
February 25, 2009	003	Updated AC Characteristics	Chapter 21, pg. 179

The Z32AN Series is a highly integrated System-on-Chip (SoC) based on the ARM922T core. Available in a 256-BGA package, the Z32AN Series provides a rich set of features on a single chip and enables designers to lower manufacturing costs and reduce time-to-market for embedded applications.



- **200 MHz ARM922T Core**
  - 32/16-bit RISC Core (ARMv4T)
  - 16-bit Thumb Instruction Capable
  - 8k/8k I/D Caches
  - MMU supporting Linux and Windows CE
  - JTAG Embedded ICE Support
- **64 KB Embedded zero-wait-state-SRAM**
- **Vectored Interrupt Controller**
- **FIPS-180-2 compliant SHA-1 Hash Generator**
- **Dual External Bus Interface**
  - 24-bit address, 16-bit data
  - SDRAM in 16-512 MB configurations
  - 10 chip selects
  - External DMA support
- **Power Management Unit**
  - 14-40 MHz oscillator and PLL
  - 32.768 kHz Oscillator for RTC
  - Clock disable per peripheral
  - 3 Modes: Active, Idle, and Battery-Backup
  - Wake from idle capability
- **9 Timer/Counters**
  - Five 32-bit cascadable
  - Four 16-bit with PWM Operation
  - Counter, PWM, Capture and Compare Modes
  - 4 Dedicated I/Os
- **USB 2.0 full speed OTG**
  - 16 endpoints
  - Dedicated DMA
- **3 UARTs**
  - 1 x 8-wire Interface
  - 2 x 4-wire Interface (1 shared with IrDA endec)
- **2 SmartCard Interface Controllers (Optional)**
  - Interfaces directly to ON Semiconductor (NCN6001) SmartCard Controller
- **Magnetic Card Reader (Optional)**
  - Support for ISO 7811-3 and 7811-6 Compliance
  - Direct Interface to Magnetic Head
  - Simultaneous Three Track Reading
- **8 independent DMA Channels**
  - SmartCard, MCR, UARTs, SPIs, LCD, external peripherals and Mem-to-Mem
  - Up to 16 MB transfer capability
- **Voltage: Dual 1.8V and 3.3V supplies**
- **Embedded boot ROM w/external boot option**
- **NIST 800-22 compliant Random Number Generator**
- **Display Controller Interface**
  - Directly Compatible with popular LCD Displays, Text or Graphic Modes
  - Interface to 4/8-bit Data, 3 Control, 1 Contrast
- **Up to 76 General Purpose I/Os**
  - 16 dedicated (with open drain capability)
  - All Configurable as Edge/Level Interrupts
  - Full Input/Output/Tri-state Control
  - GPIO Wake capability
- **4-channel, 10-bit SAR ADC, 45 kSps**
- **Real Time Clock**
- **Watchdog Timer**
- **2 Dedicated SPI Interfaces**
- **3.3V I/Os w/5V tolerant I/O for UART and SPI**
- **256 BGA Package. Boundary scan capable**

Table of Contents

List of Figures

List of Tables

Chapter 1: Pin Description

1.1 System Pins

Pin Name	Dir	Function
nRSTIN	I	**System Reset:** Schmitt trigger input.
nRSTOUT	O	**Reset Out:** 4mA drive.
CLKXI	I	**System Clock Oscillator Input**
CLKXO	O	**System Clock Oscillator Output**

1.2 External Bus Interface

Pin Name	Dir	Function
nCS[9:6]	O	**Memory Chip Selects [9:6]:** 4mA drive. Multiplexed with GPIO_0[19:16] (nCS[9] multiplexed with GPIO_0[19], etc.). After reset, these pins default as chip selects.
nCS[5:0]	O	**Memory Chip Selects [5:0]:** 4mA drive. nCS[0] is used as the external boot memory chip select.
nWEU	O	**Primary Write Strobe:** 8mA drive.
nWEL	O	**Primary Write Strobe:** 8mA drive.
nOE	O	**Primary Output Enable:** 8mA drive.
MA[23:0]	O, I/O	**Primary Address Bus:** 8mA drive. MA[23:20] multiplexed with GPIO_0[23:20] (MA[23] multiplexed with GPIO_0[23], etc.). MA[23:20] are I/O when in GPIO mode. At reset, MA[23:20] default as address pins.
MD[15:0]	I/O	**Primary Data Bus:** 8mA drive.
SDCLK	O	**SDRAM Clock:** 4mA drive.
READY	I	**Ready Input:** 4mA drive. Multiplexed with GPIO_0[26]. At reset, this pin defaults to GPIO.

1.3 Secondary External Bus Interface

Pin Name	Dir	Function
CKE	O	**SDRAM Clock Enable for Secondary Bus:** 4mA drive.
nSWEU	O	**Secondary Write Strobe:** 4mA drive.
nSWEL	O	**Secondary Write Strobe:** 4mA drive.
nSOE	O	**Secondary Output Enable:** 4mA drive.
SA[23:0]	O	**Secondary Bus Address:** 4mA drive.
SD[15:0]	I/O	**Secondary Bus Data:** 4mA drive.
nRAS	O	**SDRAM Row Address Strobe:** 4mA drive.
nCAS	O	**SDRAM Column Address Strobe:** 4mA drive.
nWE	O	**SDRAM Write Enable:** 4mA drive.
DQM[1:0]	O	**SDRAM Data Mask:** 4mA drive.

1.4 External DMA Interface

After reset, all external DMA interface pins default to GPIO.

Pin Name	Dir	Function
TxREQ	I	**External DMA Transmit Request:** Schmitt trigger input. 4mA drive. Multiplexed with GPIO_2[8].
TxACK	O	**External DMA Transmit Acknowledge:** 5V tolerant. 4mA drive. Multiplexed with GPIO_2[9].
RxREQ	I	**External DMA Receive Request:** Schmitt trigger input. 4mA drive. Multiplexed with GPIO_2[10].
RxACK	O	**External DMA Receive Acknowledge:** 5V tolerant. 4mA drive. Multiplexed with GPIO_2[11].

1.5 Timer/Counter

Pin Name	Pins	Function
PWM/TCLK[3:0]	I/O	**PWM/TCLK I/O for counters 3 to 0:** Schmitt trigger input. 4mA drive. Multiplexed with GPIO_0[30:27]. (PWM/TCLK[3] multiplexed with GPIO_0[30], etc.)

1.6 RTC

Pin Name	Type	Function
RTCXI	I	**RTC Oscillator Input:** Expected frequency is 32.768kHz
RTCXO	O	**RTC Oscillator Output:**

1.7 UARTs

At reset, all multiplexed UART pins default to GPIOs.

1.7.1 UART0

Pin Name	Dir	Function
RxD[0]	I	**UART0 Data Input:** 5V tolerant. 4mA drive.
TxD[0]	O	**UART0 Data Output:** 5V tolerant. 4mA drive.
nCTS[0]	I	**UART0 Clear to Send:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[1].
nRTS[0]	O	**UART0 Request to Send:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[4].
nDCD[0]	I	**UART0 Data Carrier Detect:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[7].
nDTR[0]	O	**UART0 Data Terminal ready:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[8].
nDSR[0]	I	**UART0 Data Set Ready:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[9].
nRI[0]	I	**UART0 Ring Indicator:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[10].

1.7.2 UART1

Pin Name	Dir	Function
RxD[1]	I	**UART1 Data Input:** 5V tolerant. 4mA drive.
TxD[1]	O	**UART1 Data Output:** 5V tolerant. 4mA drive.
nCTS[1]	I	**UART1 Clear to Send:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[2].
nRTS[1]	O	**UART1 Request to Send:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[4].

1.7.3 UART2

Pin Name	Type	Function
RxD[2]	I	**UART2 Data Input:** 5V tolerant. 4mA drive. Multiplexed with GPIO_0[31].
TxD[2]	O	**UART2 Data Output:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[0].
nCTS[2]	I	**UART2 Clear to Send:** 5V tolerant. 4mA drive. Mulitiplexed with GPIO_1[3].
nRTS[2]	O	**UART2 Request to Send:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[6].

1.8 SPI

After reset, all SPI pins default to GPIOs. Unless otherwise noted, the drive value of the GPIO is '0'.

1.8.1 Port 0

Pin Name	Dir	Function
MISO[0]	I	**SPI 0 Master DI, Slave DO:** 5V tolerant. 4mA drive. Multiplexed with GPIO[11]
MOSI[0]	O	**SPI 0 Master DO, Slave DI:** 5V tolerant. 4mA drive. Multiplexed with GPIO[13].
SCK[0]	O	**SPI 0 Clock:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[15].
nSS[0]	O	**SPI 0 Slave Select:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[17]. After reset, drives value of '1'.

1.8.2 Port 1

Pin Name	Dir	Function
MISO[1]	I	**SPI 1 Master DI, Slave DO:** 5V tolerant. 4mA drive. Multiplexed with GPIO[12]
MOSI[1]	O	**SPI 1 Master DO, Slave DI:** 5V tolerant. 4mA drive. Multiplexed with GPIO[14].
SCK[1]	O	**SPI 1 Clock:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[16].
nSS[1]	O	**SPI 1 Slave Select:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[18]. After reset, drives value of '1'.

1.9 USB Interface

Pin Name	Type	Function
USB0DN	I/O	**USB Negative Transceiver:**
USB0DP	I/O	**USB Positive Transceiver:**
USB0VBUS	A	**USB VBUS:**
USB0ID	I/O	**USB ID:**
DRIVEBUS0	AO	**USB External Power Enable:** 4mA drive.

1.10 SmartCard Interface

All SmartCard pins default to GPIOs at reset.

1.10.1 Port 0

Pin Name	Type	Function
SC_CLK[0]	O	**SmartCard Port 0 Clock:** 4mA drive. Multiplexed with GPIO_1[19].
SC_IO[0]	I/O	**SmartCard Port 0 I/O:** Internal Pull-up. 4mA drive. Multiplexed with GPIO_1[21].

1.10.2 Port 1

Pin Name	Type	Function
SC_CLK[1]	O	**SmartCard Port 1 Clock:** 4mA drive. Multiplexed with GPIO_1[20].
SC_IO[1]	I/O	**SmartCard Port 1 I/O:** Internal Pull-up. 4mA drive. Multiplexed with GPIO_1[22].

1.10.3 SmartCard SPI Interface

Pin Name	Type	Function
SC_nALARM	I	**SmartCard Alarm Interrupt:** 4mA drive. Multiplexed with GPIO_1[23].
SC_SCK	O	**SmartCard SPI Clock:** 4mA drive. Multiplexed with GPIO_1[24].
SC_MOSI	O	**SmartCard SPI MOSI:** 4mA drive. Multiplexed with GPIO_1[25].
SC_MISO	I	**SmartCard SPI MISO:** 4mA drive. Multiplexed with GPIO_1[26].
SC_nSS[1:0]	O	**SmartCard SPI Slave Select:** 4mA drive. Internal Pull-up. Multiplexed with GPIO_1[28:27] (SC_nSS[1] multiplexed with GPIO_1[28], etc.)

1.11 LCD Display Interface

At reset all LCD pins default to GPIOs, with a drive value of '0'.

Pin Name	Dir	Function
LCD_E	O	**LCD Enable:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[29].
LCD_RS	O	**LCD Register Select:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[30].
LCD_RnW	O	**LCD Read/Write:** 5V tolerant. 4mA drive. Multiplexed with GPIO_1[31].
LCD_D[7:0]	I/O	**LCD Data:** 5V tolerant. 4mA drive. Multiplexed with GPIO_2[7:0]. (LCD_D[7] multiplexed with GPIO_2[7], etc.

1.12 Dedicated General Purpose I/O

Pin Name	Type	Function
GPIO_0[25:24]	I/O	**Dedicated GPIOs [25:24]:**
GPIO_0[15:1]	I/O	**Dedicated GPIOs [15:1]:** 5V tolerant. Open drain capabile. 4mA drive.
GPIO_0[0]	I/O	**Dedicated GPIO [0]:** 5V tolerant. Open drain capabile. 4mA drive. At reset, this is the "download" pin. See TBD for more details.

1.13 ADC

Pin Name	Type	Function
ADC_IN[4:1]	AI	**ADC Analog in**
ADC_VREF	AI	**ADC Connect to External Voltage Reference**

1.14 Magnetic Card Reader

Pin Name	Type	Function
MCR_P[2:0]	AI	**Memory Card Reader Positive Analog Inputs:**
MCR_N[2:0]	AI	**Memory Card Reader Negative Analog Inputs:**
MCR_VREF	A	**Memory Card Voltage Reference:** Connect external 100nF (ceramic) to ground.

1.15 JTAG

Pin Name	Type	Function
ICE_TMS	I	**Test Mode Select:** Internal Pull-up
ICE_TDI	I	**Test Data Input:** Internal Pull-up
ICE_TDO	OZ	**Test Data Output:** 4mA drive. Requires external pull-up
ICE_TCK	I	**Test Clock:** Internal Pull-up
ICE_nTRST	I	**Test Reset:** Internal Pull-up

1.16 Power

Pin Name	Function
VDD[7:0]	1.8V Core Power
VSS[8:0]	Core Ground
VDDIO[11:0]	3.3V I/O Power
VSSIO[12:0]	I/O Ground
VBAT	3.0V Battery Power
AVDD_RTC	3.3V RTC Module Power
AVSS_RTC	RTC Module Ground
AVDD_PLL	1.8V PLL Power
AVSS_PLL	PLL Power
AVDD_MCR	3.3V Mag Reader Power
AVSS_MCR	Mag Reader Ground
AVDD_ADC	3.3V ADC Power
AVSS_ADC	ADC Ground
VDD_USB	3.3V USB Power
VSS_USB	USB Ground

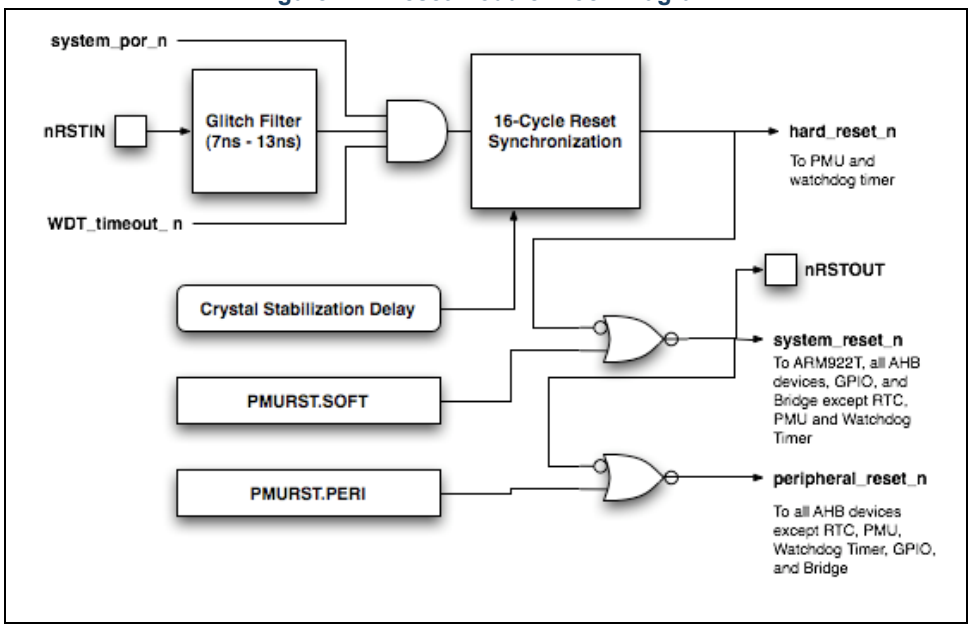
1.17 Pin Assignments, 256 BGA Package

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	MISO[0]	MOSI[0]	SCK[0]	AVSS_ ADC	AVSS_ USB	ADC_ VREF	VSSIO	CLKXI	CLKXO	LCD_ D [3]	LCD_ D [2]	LCD_ D [1]	LCD_ D [0]	PWM_ TCLK[2]	PWM_ TCLK[1]	PWM_ TCLK[0]
B	nSS[0]	MISO[1]	SCK[1]	AVDD_ ADC	AVDD_ USB	USB0 VBUS	USB0 DP	VSSIO	VDDIO	RxACK	LCD_ D [6]	LCD_ D [5]	LCD_ D [4]	PWM_ TCLK[3]	nCTS[2]	nRTS[2]
C	nCS[9]	nOE	MOSI[1]	ADC_ IN[2]	ADC_ IN{1]	ADC_ IN[4]	VDDIO	VSSIO	VDDIO	TxACK	TxREQ	LCD_ RS	LCD_ D [7]	TxD[2]	RxD[2]	nCTS[1]
D	nCS[8]	VSS	VDD	SD[0]	ADC_ IN[3]	USB0ID	USB0DN	AVSS_ PLL	AVDD_ PLL	VSS	RxREQ	LCD_ RnW	LCD_E	VSSIO	VDDIO	nRTS[1]
E	nCS[7]	VDDIO	DRIVE BUS0	SA[22]	SA[21]	nRSTIN	VSSIO	nSS[1]	GPIO_0 [8]	VDD	SC_ CLK[1]	SC_ IO [1]	SC_ MISO	GPIO_0 [14]	GPIO_0 [15]	TxD[1]
F	nCS[6]	VSSIO	SA[20]	SA[19]	SA[18]	SD[2]	SD[1]	ICE_ TMS	ICE_ TCK	GPIO_0 [11]	SC_ IO [0]	SC_n ALARM	VSS	VDD	GPIO_0 [13]	RxD[1]
G	nCS[5]	VSSIO	SA[17]	SA[16]	SA[15]	SD[4]	SD[3]	ICE_ nTRST	ICE_ TDO	GPIO_0 [10]	SC_ CLK[0]	SC_ SCK	SC_ MOSI	GPIO_ [0]	GPIO_0 [2]	nRTS[0]
H	nCS[4]	VDDIO	SA[14]	SA[13]	SA[12]	SD[6]	SD[5]	READY	ICE_ TDI	GPIO_0 [9]	GPIO_0 [12]	SC_ nSS[0]	SC_ nSS[1]	VDDIO	VSSIO	nRI0
J	nCS[3]	VSS	SA[11]	SA[10]	SA[9]	SD[8]	SD[7]	VSS	GPIO_0 [7]	GPIO_0 [1]	GPIO_0 [4]	MCR_ P[0]	MCR_ N[0]	VDD	VSS	nCTS[0]
K	nCS[2]	VDD	SA[8]	SA[7]	SA[6]	SD[10]	SD[9]	VSS	VSS	GPIO_0 [3]	GPIO_0 [5]	MCR_ N[1]	MCR_ P[1]	AVDD_ MCR	AVSS_ MCR	nDTR[0]
L	MA[22]	MA[23]	SA[5]	SA[4]	SD[11]	SD[12]	VSS	VSS	VSS	AVDD_ VBAT	GPIO_0 [6]	MCR_ N[2]	MCR_ P[2]	VSS	MCR_ VREF	RxD[0]
M	MA[20]	MA[21]	SA[3]	SA[2]	SD[13]	SD[14]	VSS	nCS[1]	RTCXO	VDD	VSS	MD[14]	nRST OUT	VDDIO	VSSIO	TxD[0]
N	MA[19]	VDD	SA[1]	SA[0]	GPIO_0 [25]	SD[15]	VDDIO	VSS	RTCXI	AVDD_ RTC	CKE	VSS	MD[10]	MD[6]	MD[2]	nDSR[0]
P	MA[18]	VSS	nCAS	nRAS	VSSIO	GPIO_0 [24]	VSSIO	SDCLK	nCS[0]	AVSS_ RTC	VSS	VSS	MD[11]	MD[7]	MD[3]	nDCD[0]
R	MA[17]	MA[14]	MA[12]	MA[10]	MA[8]	MA[6]	MA[4]	MA[2]	MA[0]	nWE	DQM[0]	nWEU	MD[12]	MD[8]	MD[4]	MD[0]
T	MA[16]	MA[15]	MA[13]	MA[11]	MA[9]	MA[7]	MA[5]	MA[3]	MA[1]	nWEL	DQM[1]	MD[15]	MD[13]	MD[9]	MD[5]	MD[1]

Chapter 2: Reset

The Z32AN Series provides 3 types of reset operations: System Reset, Hard Reset, and Peripheral Reset

Figure 2-1: Reset Module Block Diagram



2.1 System Reset

A system reset is caused by a Hard Reset or a Soft Reset. A system reset resets all modules except the Power Management Unit (PMU) and Watchdog Timer. A system reset is driven out onto the nRSTOUT pin.

2.2 Hard Reset

A hard reset is caused by the reset input pin, an internal Power-on reset or Watchdog time-out. A hard reset will reset all digital modules of the device.

The hard reset is asserted asynchronously by any of its sources. The hard reset is released synchronously to the CLKXI oscillator input after all reset sources have been inactive for 16 clock cycles.

2.3 Peripheral Reset

A peripheral reset is caused by a system or peripheral reset. This resets all APB peripherals, except the PMU, Watchdog Timer, and GPIO. This does not reset any AHB device (such as the CPU, Bridge, EBI, DMA, etc.). The source(s) of a reset are logged within PMURESET. **nRSTIN** is run through a glitch filter to improve noise immunity.

Chapter 3: System Clocks and Power Management

The Z32AN Series contains a Power Management Unit (PMU) that controls system clocking and power management. The configuration of the PMU is performed through the PMU registers. Upon a hard reset, the PMU drives the main oscillator clock onto the system clock. All digital clock domains are enabled and most analog circuits are powered down.

The PLL typically locks 100µs after it is enabled. The lock bit should be checked before switching clock source to the PLL. After the PLL has locked, the programmer can write to the PMU registers to control clock frequencies and clock gating. The PMU contains control circuits to allow for glitch-free, dynamic configuration of all clocks. The boot ROM configures the PLL at start up.

Figure 3-1: Simplified PMU Block Diagram

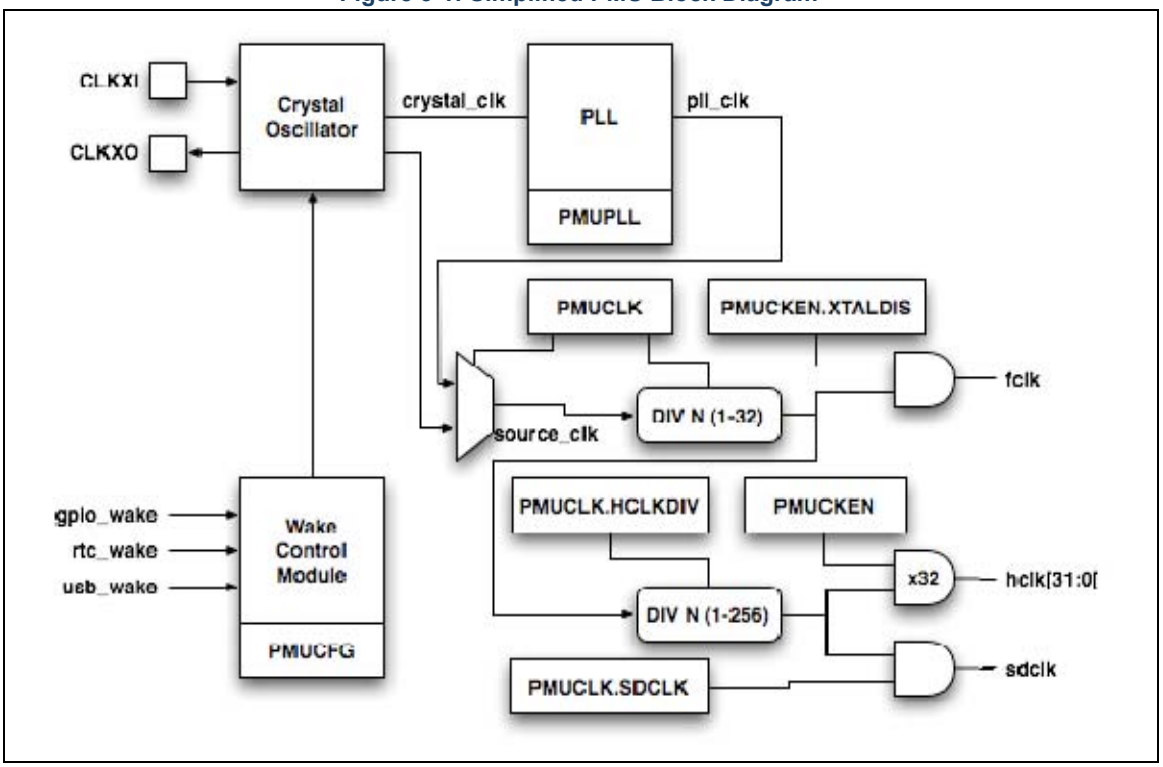


Table 3-1: PMU Module Inputs and Outputs

Signal Name	Dir	Description
CLKXI/CLKXO	I	Input/Output connection to the main external crystal. Any frequency from 14MHz-40MHz may be used. A clock must be present for the device to come out of reset.
fclk	O	Main CPU clock.
hclk[31:0]	O	System clocks. These clocks are driven to the AHB and APB devices as well as the AHB interface of the CPU. This clock can be enabled or disabled on a module-by-module basis. (See PMU Clock Enable Register (ADDR = 0xFFFF_E008)
daaclk	O	Clock driven into the DAA. The DAA requires a 24.000MHz clock.
sdclk	O	Clock eventually driven onto the CLKOUT pin. This signal first goes through the SDRAM Controller module where it may be subject to additional clock gating.
gpio_wake	I	Signal from GPIO to wake system upon any GPIO input transition. See Using the GPIO Wake Function. Though shown as a single signal, there are wake signals from all GPIO modules.
rtc_wake	I	Signal from RTC to wake system upon activation of the RTC alarm. See Real-Time Clock (RTC).

3.1 Power Modes

Mode	Initiation	Wake	Restrictions
HALT ARM922T Wait-for-IRQ	CP15 of ARM922T	IRQ or FIQ active into ARM922T	Clock must remain active to the INTC. The clocks of other modules can be deactivated in the PMU to reduce dynamic power consumption. Clocks for modules expected to generate interrupts should be left enabled.
IDLE	PMU_CFG and PMU_CLK_EN	Transition on selected GPIO inputs or RTC Alarm or USB wake	None. All clocks can be disabled but the PLL and Crystal continue to run.
STOP	PMU_CFG	Transition on selected GPIO inputs or RTC Alarm or USB wake	None. All clocks are disabled and the PLL and Crystal are disabled.
Battery Back Up	Automatic when power is removed	POR	None. Only the RTC remains active. Normal operation resumes when power is returned to the system.

3.1.1 HALT - ARM922 Wait for Interrupt

This is an intrinsic function of the ARM922T. The ARM922T contains a Wait for Interrupt mechanism within the CP15 registers to put the ARM922T in a low power state until the arrival of an IRQ or FIQ. More information can be found in the ARM922T Technical Reference Manual. Prior to placing the CPU in this state, turn off any unnecessary clocks in PMU_CLK_EN to further reduce power. In this mode, the INTC and any peripherals expected to generate interrupts should be left enabled. All other enables may be off.

3.1.1 IDLE

In this mode all clocks are disabled. The PLL and crystal continue to run for fast return to full active operation. The return to active mode is achieved by transitions on selected GPIO, RTC alarm or USB wake. To enter this mode, power down the peripherals and select the wake mechanisms in PMUCFG and turn off the clocks in PMU_CLK_EN.

3.1.2 STOP

In this mode, all the clocks, PLL, and Crystal are disabled. Return to active mode is achieved by transition on selected GPIO, RTC alarm or USB wake. This mode requires an additional delay to active mode while the crystal starts and PLL re-locks. To enter the this mode, power down the peripherals, select the wake mechanisms and disable XTAL with one write to PMUCFG. Once XTAL disabled, the clocks stop. Normal operation is resumed when the one of the selected wake mechanisms occurs and the delay for XTAL stabilization has elapsed.

3.1.3 Battery Back Up

The system is in battery back up mode when all power is removed except the battery. In this mode, the RTC continues to operate. Return to active is achieved by applying valid system power. Resuming normal operation is automatic with the ROM boot sequence occurring.

3.2 Wake Mechanisms

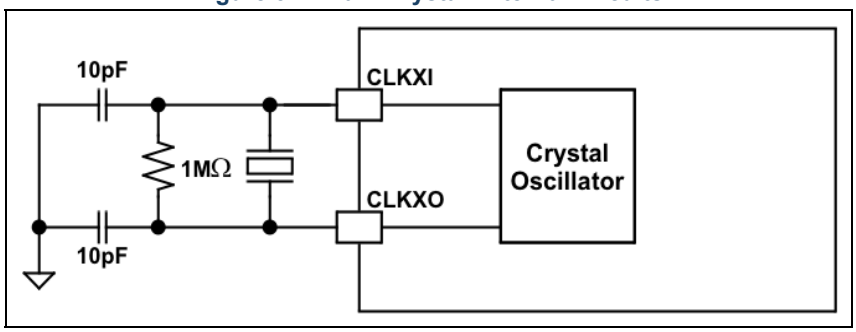
- **GPIO Wake:** This can be used to return to active operation from Idle or Stop Modes. All GPIO modules can be configured to wake in the event of a GPIO input edge by setting GPIO_WAKE_EN. See section Chapter 20: for more details.
- **RTC Wake:** The RTC alarm can also be used to return to active operation from Idle or Stop Modes. The PMU can be configured to “wake” the PMU clocks upon activation of the RTC alarm. See section Chapter 11: for more details.

- **USB Wake:** USB can return the SoC to active operation from Idle or Stop Modes. The lowest power for USB suspend is achieved through “stop” mode. The PMU can be configured to wake when USB activity is detected.

3.3 Main Oscillator External Circuits

The required crystal circuit for the main oscillator is shown below. Supported crystal frequencies are 14MHz - 40MHz..

Figure 3-2: Main Crystal External Circuits



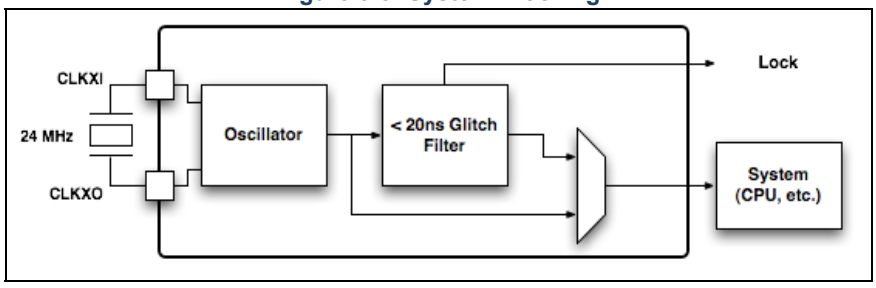
To allow the oscillator to stabilize, the PMU has a 16-bit ripple counter which will block the first 64k crystal clock cycles after a power on reset or when the crystal is re-enabled by a wake function.

3.4 System Clocking Notes

The system outlined in Figure 4-3 isolates sysclk from any glitches or over/under-clocking:

- Fast glitches (< 20ns) are prevented from entering the system by the 20ns Glitch Filter between the oscillator and the rest of the system. This limits the output of the oscillator from generating a signal faster than 50MHz. This is in effect whether or not sysclk is derived from the PLL.
- Fast frequency over-clocking is prevented by the PLL losing lock in the event of sudden changes to the input clock frequency. This is only operate when sysclk is derived from the PLL.
- Under-clocking is prevented by the PLL losing lock when the input clock is below the minimum requirement of the PLL. This will operate only when sysclk is derived from the PLL.

Figure 3-3: System Clocking

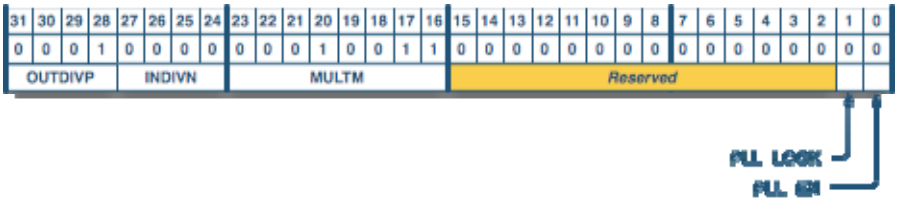


3.5 PMU Registers: (Base → FFFFE000h)

The PMU registers are reset by a hard reset, unless otherwise noted in the register description. Most of the system and registers are reset by a system reset, but the PMU and Watchdog Timer are exceptions.

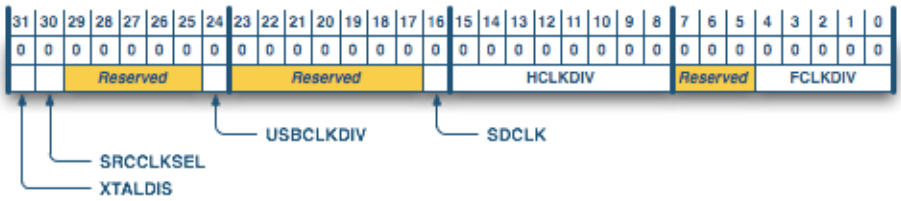
Offset	Register	Description
000h	PMUPLL	PMU PLL Register
004h	PMUCLK	PMU Clock Control Register
008h	PMUCKEN	PMU Clock Enable Register
00Ch	PMURESET	PMU Reset Register
014h	PMUID	PMU ID Register
01Ch	PMUCFG	PMU Configuration Register

3.5.1 Offset 000h: PMUPLL – PMU PLL Register



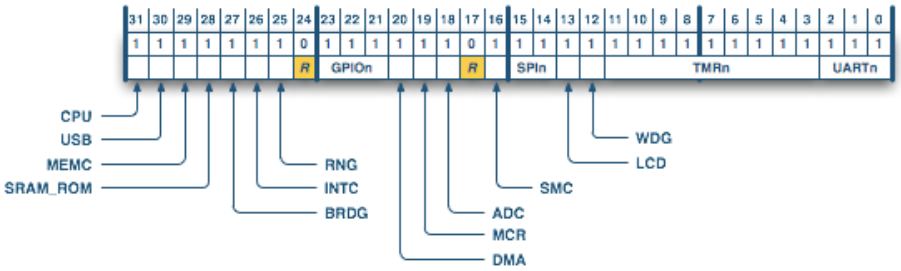
Bits	Type	Reset	Description
31:28	RW	1h	**PLL Output Divider "P" (OUTDIVP):** Determines how much the pll_clk is divided after frequency multiplication. • 0000: pll_clk is divided by 1 • 0001: plll_clk is divided by 2 • ... • 1111: pll_clk is divided by 16
27:24	RW	0h	**PLL Input Divider "N" (INDIVN):** Determines how much the crystal_clk is divided before frequency multiplication. This must result in a frequency above 12 MHz • 0000: crystal_clk is divided by 1 • 0001: crystal_clk is divided by 2 • ... • 1111: crystal_clk is divided by 16
23:16	RW	13h	**PLL Multiplier "M" (MULTM):** Determines how much the frequency is multiplied. This must result in a frequency below 500 MHz and greater than 225 MHz • 00h: ILLEGAL SETTING • 01h: frequency multiplier factor is 2 • 02h: frequency multiplier factor is 3 • ... • FFh: frequency multiplier factor is 256
15:02	*RO*	*0*	*Reserved*
01	RO	0	**PLL Lock (PLL_LOCK):** When cleared, not locked. When set, locked.
00	RW	0	**PLL Power Enable (PLL_ENJ):** When cleared, power down. When set, power up.

3.5.2 Offset 004h: PMUCLK – PMU Clock Control Register



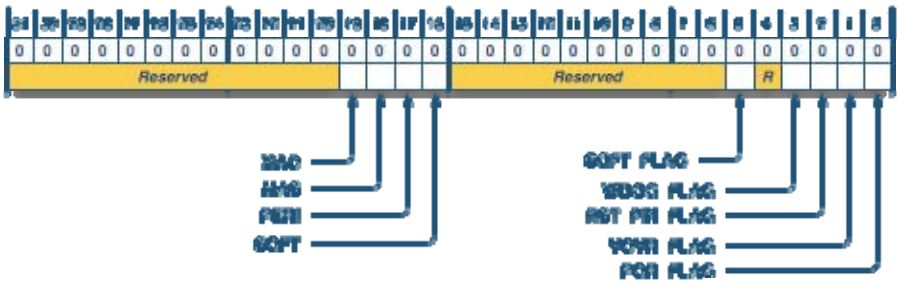
Bits	Type	Reset	DescriptionDS0200.docx
31	RW	0	**Crystal Disable (XTALDIS):** When cleared, the crystal is disabled to put the device into low power sleep mode. This bit is automatically cleared by wake up.
30	RW	0	**Source Clock Select (SRCCLKSEL):** When cleared, select crystal clock as the source clock. When set, select PLL clock as the source clock.
29:25	*RO*	*0*	*Reserved*
24	RW	0	**USB Clock Divider (USBCLKDIV):** When cleared, use HCLK. When set, use HCLK divided by 2.
23:17	*RO*	*0*	*Reserved*
16	RW	0	**External SDRAM Clock Enable (SDCLK):** When cleared, SDCLK is disabled. When set, SDCLK is enabled.
15:08	RW	00h	**HCLK Divider (HCLKDIV):** This specifies how much to divide fclk to produce hclk. • 00h: 1 (hclk = fclk) • 01h: 2 (hclk = fclk / 2) • ... • FFh: 256
07:05	*RO*	*0*	*Reserved*
04:00	RW	00000	**FCLK Divider (FCLKDIV):** This specifies how much to divide source clock to produce fclk. • 00000: 1 (fclk = source clock) • 00001: 2 (fclk = source clock / 2) • ... • 11111: 32

3.5.3 Offset 008h: PMUCKEN – PMU Clock Enable Register



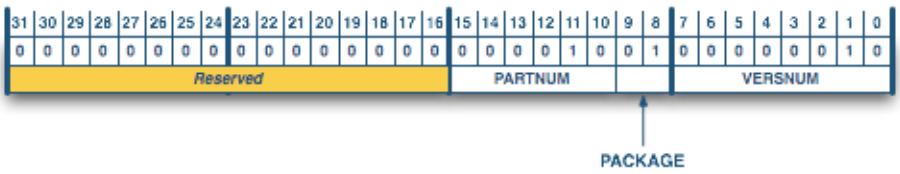
Bits	Type	Reset	Description
31	RW	1	**CPU Clock Enable (CPU):** Enables both CPU clocks (fclk and hclk). Forced to '1' by wake up.
30	RW	1	**USB Clock Enable (USB):** Enables USB clock. Forced to '1' by wake up.
29	RW`	1	**Memory Controller Enable (MEMC):** Enable hclk to the External Memory Controller and SDRAM controller. Forced to '1' by wake up.
28	RW	1	**SRAM/ROM Enable (SRAM_ROM):** Enable hclk to the Internal SRAM and ROM. Forced to '1' by wake up.
27	RW	1	**Bridge Enable (BRDG):** Enable hclk to the AHB bridge. Forced to '1' by wake up.
26	RW	1	**Interrupt Controller Clock Enable (INTC):** Enable hclk to the Interrupt Controller. Forced to '1' by wake up.
25	RW	1	**Random Number Generator Clock Enable (RNG):** Enable hclk to the Random Number Generator.
24	*RO*	*0*	*Reserved*
23:21	RW	111	**GPIO "N" Enable (GPIOn):** Enable hclk to GPIO "N". Forced to '1' by wake up. Bit 23 = GPIO2, bit 21 = GPIO0.
20	RW	1	**DMA Controller Clock Enable (DMA):** Enable hclk to the DMA Controller
19	RW	1	**MCR Enable (MCR):** Enable hclk to MCR (all 3 channels)
18	RW	1	**ADC Enable (ADC):** Enable hclk to the ADC
17	*RW*	*1*	*Reserved*
16	RW	1	**SmartCard Enable (SMC):** Enable hclk to the SmartCard Interface
15:14	RW	11	**SPI "N" Enable (SPIn):** Enable hclk to SPI "n". Bit 15 = SPI1, Bit 14 = SPI0.
13	RW	1	**LCD Enable (LCD):** Enable hclk to LCD Display Controller
12	RW	1	**Watchdog Enable (WDOG):** Enable hclk to the Watchdog Timer
11:03	RW	FFh	**Timer "N" Enable (TMRn):** Enable hclk to Timer "n". Bit 11 = Timer 8, Bit 3 = Timer 0.
02:00	RW	111	**UART "N" Enable (UARTn):** Enable hclk for UART"n". Bit 2 = UART2, Bit0 = UART0.

3.5.4 Offset 00Ch: PMURESET – PMU Reset Register



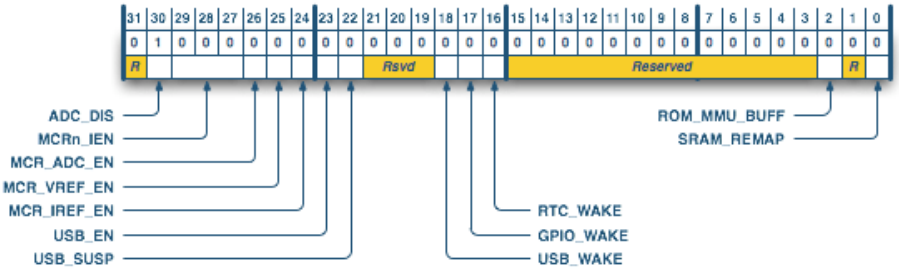
Bits	Type	Reset	Description
31:20	*RO*	*0*	*Reserved*
19	WO	0	**Manufacturing Access Disable (MAD):** Writing to '1' disables manufacturing access. Writing a '0' has no effect.
18	RW	0	**Manufacturing Access Status: (MAS):** When read, this bit indicates the status of manufacturing access. When written as '1', this bit enables manufacturing access. Writes of '0' have no effect. This bit is reset by a system reset (not just a hard reset).
17	WO	0	**Peripheral Reset (PERI):** When set, causes a reset of all APB peripherals except the PMU, WDT and GPIO.
16	WO	0	**Soft Reset (SOFT):** When set, causes a system reset. This includes a reset of all modules except the PMU and the WDT.
15:06	*RO*	*0*	*Reserved*
05	RW1C	0	**Soft Reset Flag (SOFT_FLAG):** When set, system was reset due to a soft reset. Reset only by a POR (not a hard reset).
04	*RO*	*0*	*Reserved*
03	RW1C	0	**Watchdog Reset Flag (WDOG_FLAG):** When set, the system was reset due to a Watchdog time-out. Reset only by a POR (not a hard reset).
02	RW1C	0	**nRSTIN Pin Reset Flag (RST_PIN_FLAG):** When set, the system was reset due to the nRSTIN pin being pulled low. Reset only by a POR (not a hard reset).
01	RW1C	0	**Over-Voltage Reset Flag (VOVR_FLAG):** When set, the system was reset due to a DVDD over voltage detection. Reset only by a POR (not a hard reset).
00	RW1C	0	**POR Reset Flag (POR_FLAG):** When set, the system was reset due to a DVDD Power-on-Reset condition. Reset only by a POR (not a hard reset).

3.5.5 Offset 014h: PMUID – PMU ID Register



Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:10	RO	000010	**Part Number (PARTNUM):** Indicates Z32AN series
09:08	RO	01	**Package Type (PACKAGE):** Indicates BGA
07:00	RO	02h	**Version Number (VERSNUM):** Indicates version AA

3.5.6 Offset 01Ch: PMUCFG – PMU Configuration Register



Bits	Type	Reset	Description
31	*RO*	*0*	*Reserved*
30	RW	1	**ADC Power Disable (ADC_DIS):** When set, enables power to the ADC.
29:27	RW	000	**MCR "N" Power Enable (MCRn_IEN):** Bit 30 = MCR2, bit 28= MCR0. When set, enables power to the MCR "N" analog front end. Note: all MCR power enable bits must be set to use the MCR
26	RW	0	**MCR ADC Power Enable (MCR_ADC_EN):** When set, enables power to the MCR ADC. Note: All MCR Power Enable bits must be set to use the MCR
25	RW	0	**MCR VREF Power Enable (MCR_VREF_EN):** When set, enables power to the MCR voltage reference. Note: All MCR Power Enable bits must be set to use the MCR
24	RW	0	**MCR IREF Power Enable (MCR_IREF_EN):** When set, enables power to the MCR current reference. Note: All MCR Power Enable bits must be set to use the MCR
23	RW	0	**USB Power Enable (USB_EN):** When set, enables power to USB
22	RW	0	**USB Suspend (USB_SUSP):** When set, place USB in suspend mode
21:19	*RO*	*0*	*Reserved*
18	RW	0	**USB Wake Enable (USB_WAKE):** When set, enable wake when USB not IDLE.
17	RW	0	**GPIO Wake Enable (GPIO_WAKE):** When set, enables wake on GPIO wake.
16	RW	0	**RTC Wake Enable (RTC_WAKE):** When set, enables wake on RTC wake.
15:03	*RO*	*0*	*Reserved*
02	RO	0	**ROM MMU Table Write mode (ROM_MMU_BUFF):** When set, disables the buffer-able bit in the ROM MMU table. If the buffer-able bit is set, writes that hit in the cache are marked dirty and written back later. If the buffer-able bit is clear, writes that hit in the cache are written through and memory is updated immediately. The setting of this bit is the inverse of the buffer-able bit that appears in the cacheable region of the ROM MMU table. (See section 5.6.4)
01	*RO*	*0*	*Reserved*
00	RW	0	**SRAM Re-Map (SRAM_REMAP):** When cleared, ROM appears at the bottom of memory. When set, SRAM appears at the bottom of memory

Chapter 4: ARM922T Core and Embedded ICE

The ARM922T core of the Z32AN Series contains a JTAG interface and an embedded In-Circuit Emulator (ICE) interface. For more details on these features, refer to the *ARM922T Technical Reference Manual*.

Big Endian handling is not supported by the Z32AN Series. All data is treated as Little Endian.

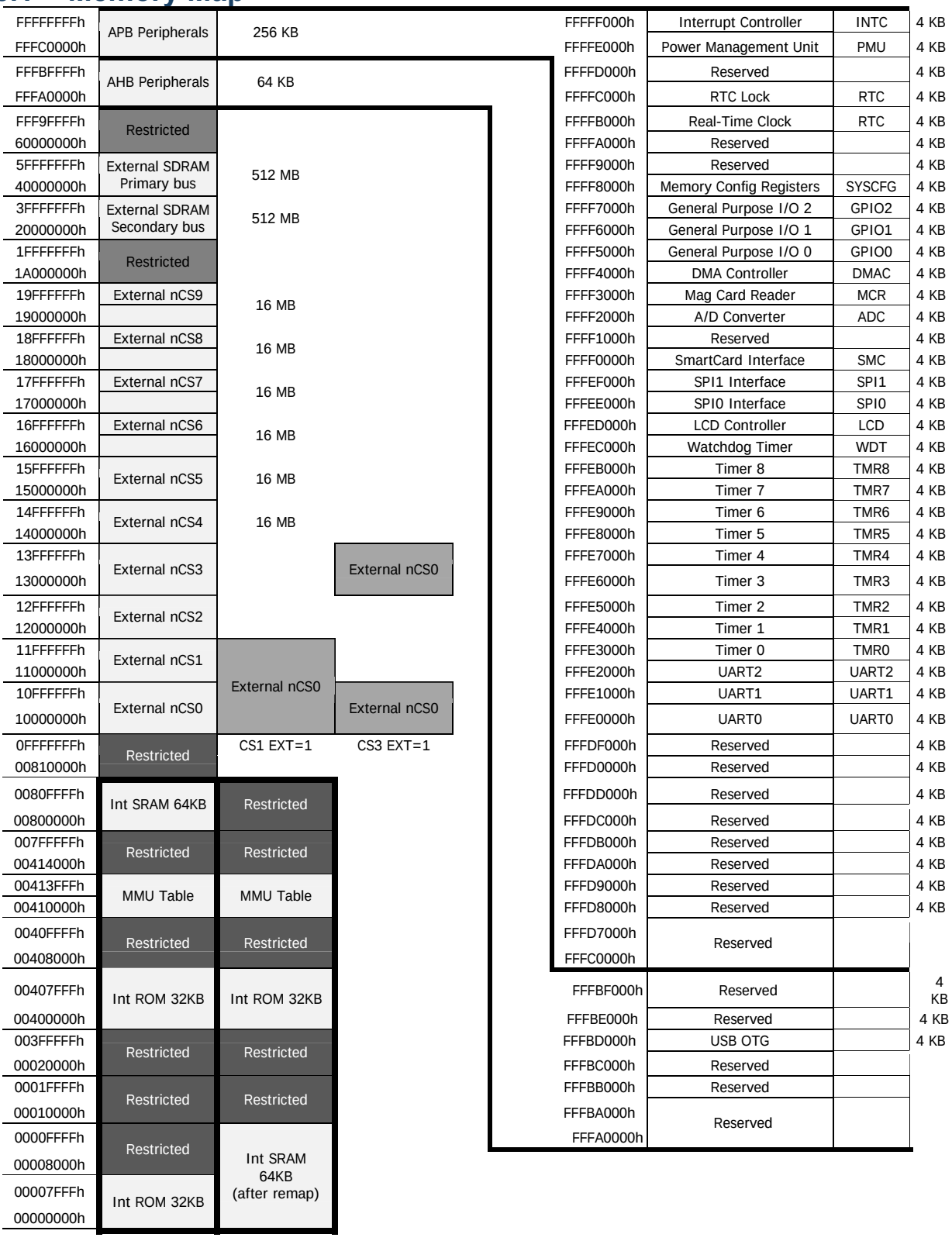
The Z32AN Series supports the following ARM922T clocking modes:

- FastBus (CPUCLK = hclk)
- Synchronous (CPUCLK = fclk, hclk = fclk/N where N = 2, 3, etc.)
- Asynchronous (CPUCLK = fclk, fclk ≥ hclk)

The general restrictions are that fclk ≤ 200 MHz and hclk ≤ 100 MHz

Chapter 5: Memory Organization

5.1 Memory Map



REMAP=0 REMAP=1

5.2 Accesses

All AHB devices can be accessed with byte, halfword or word accesses. APB devices can be read as bytes or halfwords, but should only be written with word accesses. Byte or halfword writes to APB devices are executed as word writes with the defined data being repeated in the other byte lanes to form 32 bits of data. Example: a byte write of AFh results in a word write of AFAFAFAFh). Byte and halfword reads will operate properly on all APB devices, but some data may be lost (when reading a 32-bit wide FIFO register, for example).

5.3 Restricted / Reserved Addresses

Accesses to **restricted** areas will result in a bus error to the requesting device (CPU or DMAC). Addresses shown as **reserved** will have no effect and will not result in a bus error.

5.4 ROM/SRAM Remapping

Remapping is controlled by a bit in the PMU registers. ROM/SRAM remapping is shown below:

Address	REMAP = 0	REMAP = 1
00000000h - 00007FFFh	Internal ROM (32KB)	Internal SRAM (64KB)
00008000h - 0000FFFFh	Restricted	
00400000h - 00407FFFh	Internal ROM (32KB)	Internal ROM (32KB)
00800000h - 0080FFFFh	Internal SRAM (64KB)	Restricted

5.5 Internal SRAM

The internal SRAM is 64 KB of 0 wait-state memory.

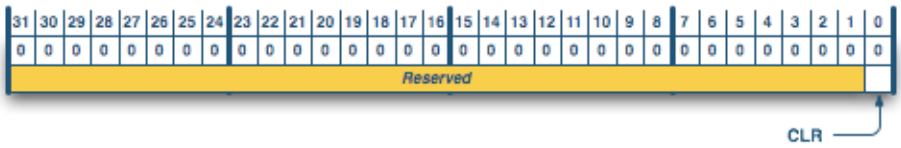
5.5.1 Clock Disable

The clock to the SRAM can be disabled via PMU registers to reduce the power consumption of this module (see System Clocks and Power Management. This clock disable does not disrupt the contents of the memory.

5.5.2 Zeroization

The internal SRAM can be automatically cleared by writing 1 to bit 0 in INT_SRAM_CLR.CLR. Clearing takes 64k hclks. During this time, reads of the SRAM are stalled.

5.5.3 Address FFFF8068h: INT_SRAM_CLR – Internal SRAM Clear Register



Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	RW	0	**Clear (CLR):** When written to '1', hardware writes 0's to internal SRAM from top address to zero. When read as '0', the internal clear-to-zero is done. While still set to '1', the clear-to-zero is still in progress. A write of '0' has no effect.

5.6 Internal ROM and Boot Program

After reset, the ARM CPU begins executing code out of the internal ROM at address 0h.

5.6.1 Boot locations

After reset, boot code will search both the primary secondary busses for a valid image. It searches for a fixed constant at a specific memory location to determine if a valid image exists on that memory bus and chip select

To facilitate initial programming and code updates, boot code is also capable of downloading an application using one of the UARTs. It checks GPIO[0] at startup. If '1', it searches for an application to download using one of the UARTs.

5.6.2 External Memory Image Format

Offset	Contents		Comments
...	Application Code	Code	Program to be executed
Application Start			
...	User defined		The space between the application info and application code can be any user defined data
	Application Info	Application Length	Length of the application
		Application Start	First non-zero word above the ECB
1Ch	External Control Block (ECB)	00000000h	Reserved
18h		00000000h	Reserved
14h		00000000h	Reserved
10h		00000000h	Reserved
0Ch		D3C2B1A0h	Fixed constant for quick checking of the image
08h		MEMC_GCFG	Register value loaded by Boot ROM into MEMC
04h		MEMC_TIM0	Register value loaded by Boot ROM into MEMC
00h		MEMC_CFG0	Register value loaded by Boot ROM into MEMC

The External Control Block contains:

- Memory bus width in MEMC_CFG0. The location of byte 0 appears at the same location in both 8-bit and 16-bit wide memories.
- Fixed constant with the value of D3C2B1A0h. This is used to check that a valid boot image exists in this external memory.
- MEMC and PMU register values. These are loaded into their corresponding registers by the boot code. This is to allow for optimal memory access time for the remainder of the load process.

5.6.3 Boot Sequence

5.6.3.1 Download Pin (GPIO[0])

Before boot code begins searching external memory, it checks GPIO[0]. If this pin is high, boot code skips external memory checks and tries to download an image from a UART.

5.6.3.2 External Memory

- **Bus Search:** Boot code searches for code in external memory on nCS[0]. It searches the secondary bus first followed by the primary bus.
- **Bus Width:** Boot code initially sets up the memory controller for the maximum timings. The boot code will first read the byte at offset 0 to determine the memory bus width. The boot code programs this value into MEMC_CFG0.
- **Fixed Constant:** Boot code then reads the data word at offset 0Ch for the fixed constant D3C2B1A0h. If the boot code finds it, it reads the rest of the External Control Block (ECB) parameters and sets up the memory controller timings and PLL based upon the ECB settings.
- **Starting Address:** If the External Control Block was found, the boot code will then search for the application start address immediately after the ECB. The application start address is the first non-zero word following the ECB. Once the application start pointer is found, boot code jumps to this application start address. Since ARM code must be word aligned, this address must be word aligned. The CPU is placed in its reset state (the MMU and caches are disabled) before branching to the start address.

5.6.3.3 UART Download

If an image was not found in external memory, or if the download pin GPIO[0] was asserted, the boot code tries to load an image from one of the UARTs. The boot code will search all of the UARTs for download activity.

Since the operating frequency of the device is unknown, boot code uses the timers to perform automatic baud rate detection. It expects the first character received to be the ASCII carriage return character '\r' 013. The timers will measure the length from the end of the start bit to the beginning of the stop bit to determine the baud rate. The UART is setup using 8 data bits, no parity,1 stop bit. The maximum baud rate is limited by the capability of the UART and system clock. A rate of 115k baud is achievable using a 24MHz clock.

Once the baud rate is setup, boot code displays a prompt as confirmation that the baud rate is setup correctly. The boot code then expects an Intel hex file to be sent. This file is loaded at address 0. Once download is complete (end of file record found), boot code executes the application at address 0.

5.6.4 Boot ROM MMU Table

A simple MMU table is implemented starting at address 00410000h. The Virtual to Physical mapping as well as the Cacheable/Bufferable settings are shown below.

Virtual Address	Physical Address	Cacheable/Bufferable	Comments
FFFFFFFFh ... C0000000h	FFFFFFFFh ... C0000000h	None	APB Devices
FFFFFFFFh ... C0000000h	FFFFFFFFh ... C0000000h	Non-Cacheable & Non-Bufferable	Peripherals
BFFFFFFFh ... 80000000h	3FFFFFFFh ... 00000000h	Non-Cacheable & Non-Bufferable	Non-cacheable view of lowest quadrant.
7FFFFFFFh ... 00000000h	7FFFFFFFh ... 00000000h	Cacheable & Bufferable (write-back mode)	

Chapter 6: Interrupt Controller (INTC)

The interrupt controller is an APB device that prioritizes and routes all interrupt channels from internal peripherals and external devices to the CPU. Features:

- IRQ or FIQ generation for each interrupt source (programmable)
- Unique vectors for each interrupt channel
- Programmable priority for each channel (8 priority levels)
- Support for nesting and preemption by higher priority interrupts (IRQ only)

Interrupts are first fed into a Priority Encoder. The highest priority enabled interrupt is passed on to either the IRQ or FIQ Processor. These are nearly identical and fully independent of each other – the INTC does not prioritize FIQ delivery over IRQ delivery. These blocks pass interrupt requests to the CPU and provide the proper vector when software reads the vector register. Additionally, the IRQ processor includes support for interrupt nesting (interruption of ISRs).

6.1 Interrupt Channels and Sources

Table 6-1: Interrupt Source to Channel Mapping

#	Source	#	Source	#	Source	#	Source
0	TMR6	8	TMR5	16	GPIO2 A	24	SPI1
1	WDT	9	SmartCard Alarm	17	GPIO2 B	25	UART1
2	UART0	10	SmartCard 0	18	DMAC	26	UART2
3	TMR0	11	SmartCard 1	19	MCR	*27*	*Reserved*
4	TMR1	12	GPIO0 A	20	USB	28	RNG
5	TMR2	13	GPIO0 B	*21*	*Reserved*	29	RTC
6	TMR3	14	GPIO1 A	22	ADC	30	TMR7
7	TMR4	15	GPIO1 B	23	SPI0	31	TMR8

6.2 Interrupt Priority

Each interrupt channel can be assigned a priority level from 0 (highest) to 7 (lowest). This is done by programming the INTC_CFGN. The interrupt priority level is only effective if the channel is configured as an IRQ interrupt (not FIQ). All FIQ interrupts are considered to have a priority of 0.

The priority has two effects:

- If two or more enabled interrupts of the same type (IRQ/FIQ) arrive at the same time, the higher priority interrupt will be serviced first. If two or more interrupts of the same priority arrive, the interrupt with the lower channel number will have priority.
- When using nested interrupts, only an interrupt of a higher programmed priority will generate a new (nested) interrupt to the CPU. Interrupts of the same or lower priority are masked until the CPU has indicated the completion of the ISR by writing to the INTC_IEND (or INTC_FEND for FIQ)

6.3 Configuring the Interrupt Controller

Prior to configuring a particular channel, a number of settings must be made:

1. The ARM exception table should be set up. This typically is done by remapping the SRAM to appear at the bottom of memory and placing the following ARM instruction at locations 0x00000018 (IRQ Vector) and 0x0000001C (FIQ Vector) as shown in the example below:
   - LDR PC, [PC,#0xFFFFFF00 - (0x18+8)] ; IRQ Vector (located at A=0x0018)
   - LDR PC, [PC,#0xFFFFFF04 - (0x1C+8)] ; FIQ Vector (located at A=0x001C)
2. INTC_DFLT should be initialized to the address of an error handling ISR. This is to provide a vector for error handling in the case where INTC_IVEC or INTC_FVEC are read when there is no interrupt active. This usually indicates a spurious interrupt or a software error.
3. Set up any or all of the interrupt channels. The following registers and fields are used:
   - INTC_CFGN → IRQ_FIQ, PRI
   - INT_VEC_N → All Bits
   - INTC_EN → Bit "N"
4. The "I" and/or "F" bits of the ARM922T CPSR register should be cleared to enable IRQ and/or FIQ interrupts.

6.4 ISR Invocation

Below is an example of how the CPU, ARM Exception Table, INTC_VECN's, and INTC_IVEC operate together to invoke a Channel 1 ISR upon occurrence of a Channel 1 interrupt. Note the sequence of steps listed at the bottom of the table. FIQ ISR invocation is almost identical, except that the CPU will read the FIQ vector in the ARM exception table and this will cause INTC_FVEC to be read.

1. IRQ is set up:
   - The IRQ Vector, INTC_VEC_1 are initialized and the Channel 1 ISR is located as shown above.
   - Channel 1 is configured as an IRQ (as opposed to an FIQ), programmed to have the highest priority, and enabled.
   - A channel 1 interrupt enters the INTC, which is passed on to the CPU by activating IRQ.
2. An active IRQ causes the CPU to execute the IRQ vector instruction at 00000018h:
   - LDR PC, [PC, #0xFFFFFF00-(0x18+8)]
3. LDR instruction causes CPU to read memory location FFFFFF00h (INTC_IVEC).
4. INTC provides the vector of the highest priority active interrupt (00000180h).
5. CPU move the data 00000180h) into PC. Program execution then continues at this address.

6.5 ISR Return from Interrupt

Before returning from an IRQ or FIQ interrupt, the ISR must write to INTC_IEND or INTC_FEND. The data written is not important and is discarded. After this write is done, the final instruction of an IRQ or FIQ ISR must restore the PC and CPSR. When using nested interrupts, the interrupts should be disabled before writing to INTC_IEND.

6.6 Interrupt Nesting

The IRQ processor nests interrupts (preemption of an ISR by a higher priority interrupt). The FIQ processor cannot nest interrupts. Active and enabled interrupt channels with a higher programmed priority always result in an interrupt being passed on to the CPU (via either IRQ or FIQ). To utilize this feature, software must clear the "I" bit of the ARM922T CPSR register within the ISR to re-enable interrupts, thereby "enabling" interrupt nesting. More information can be found in Chapter A2, Programmer's Model of the *ARM Architecture Reference Manual*.

To keep track of the nested interrupts, the IRQ Processor contains a priority stack. Since there are 8 priorities, the stack is 8 deep. A read of INTC_IVEC pushes the stack. A write to INTC_IEND pops the stack. During the time between push and pop, interrupts of the same or lower priority are masked. The programmer's responsibility is to ensure there is a write to INTC_IEND every read of INTC_IVEC.

Certain registers within the ARM922T may be overwritten by nested interrupts, namely SPSR and LR(R14). These registers should be saved prior to re-enabling interrupts and restored after interrupts are re-disabled. Refer to the *ARM Architecture Reference Manual* for complete information.

6.7 Interrupt Latching

Interrupts are not latched within the INTC and it is the responsibility of the ISR to perform whatever operations are necessary to clear the interrupt at the source peripheral where it is latched. An enabled interrupt into the INTC will be passed on to the CPU via the IRQ/FIQ signals. This interrupt will remain active until cleared at the source or the corresponding interrupt enable bit is cleared.

6.8 Registers: Base → FFFFF000h

Offset	Register	Description
000h	INTC_EN	Interrupt Enable Register
004h	INTC_ESET	Interrupt Enable Set Register
008h	INTC_ECLR	Interrupt Enable Clear Register
00Ch	INTC_DFLT	Default Vector
010h	INTC_ISTA	Interrupt Status Register
014h	INTC_RSTA	Raw (Unmasked) Interrupt Status Register
018h	INTC_IDBG	IRQ Processor Debug Register
01Ch	INTC_FDBG	FIQ Processor Debug Register
020h	INTC_SWINT	Software interrupt register
024h	INTC_SWINT_SET	Software interrupt set register
028h	INTC_SWINT_CLR	Software interrupt clear register
080h - OFCh	INTC_VECN	Channel “N” Vector Register (N = 0 to 31)
100h – 17Ch	INTC_CFGN	Channel “N” Configuration Register (N = 0 to 31)
F00h	INTC_IVEC	IRQ Vector Register
F04h	INTC_FVEC	FIQ Vector Register
F08h	INTC_IEND	IRQ End-of-Interrupt Register
F0Ch	INTC_FEND	FIQ End-of-Interrupt Register

6.8.1 Offset 000h: INTC_EN – Interrupt Controller Enable Register

Bits	Type	Reset	Description
31:00	RW	0	**Enable (EN):** When set, enables the corresponding channel. Bit 0 enables channel 0, Bit 1 enables channel 1, etc. These bits can be set or cleared by writing directly to it, or by writing to INTC_ESET and INTC_ECLR.

6.8.2 Offset 004h: INTC_ESET – Interrupt Controller Enable Set Register

Bits	Type	Reset	Description
31:00	WO	0	**Set (SET):** Wires of '1' set the corresponding bit in INTC_EN.

6.8.3 Offset 008h: INTC_ECLR – Interrupt Controller Enable Clear Register

Bits	Type	Reset	Description
31:00	WO	0	**Clear (CLR):** Writes of '1' clear the corresponding bit in INTC_EN.

6.8.4 Offset 00Ch: INTC_DFLT – Default Vector Register

Bits	Type	Reset	Description
31:00	RW	0	**Default Vector (VEC):** Contains the value to appear in either INTC_IVEC or INTC_FVEC in the case that no IRQ or FIQ is active when that register is read. Should be loaded with a valid ISR address (to handle this error condition).

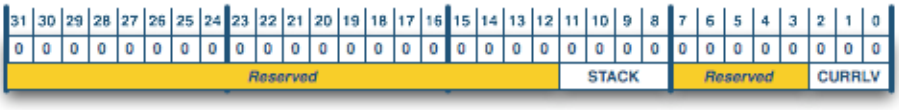
6.8.5 Offset 010h: INTC_ISTA – Interrupt Status Register

Bits	Type	Reset	Description
31:0	RO	0	**Status (STS):** Indicates which enabled (non-masked) interrupts are active. Bit 0 corresponds to channel 0, Bit 1 enables channel 1, etc. When cleared, the interrupt is either not enabled or not active. When set, the interrupt is enabled and active.

6.8.6 Offset 014h: INTC_RSTA – Raw Interrupt Status Register

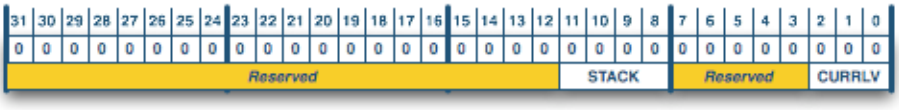
Bits	Type	Reset	Description
31:00	RO	0	**Status (STS):** Indicates which interrupts are active before masking. Bit 0 corresponds to channel 0, Bit 1 enables channel 1, etc. When cleared, the interrupt is active. When set, the interrupt is not active.

6.8.7 Offset 018h: INTC_IDBG – IRQ Processor Debug Register



Bits	Type	Reset	Description
31:12	*RO*	*0*	*Reserved*
11:08	RO	0h	**Stack Pointer (STACK):** Holds the current stack pointer for the IRQ Processor Stack. • 0000: Stack is empty; no entries are valid. • 0001: Stack holds one value. • .... • 1000: Stack is full (holds 8 values).
07:03	*RO*	*0*	*Reserved*
02:00	RO	0	**Current Level (CURRLV):** Indicates the current priority level.

6.8.8 Offset 01Ch: INTC_FDBG – FIQ Processor Debug Register



Bits	Type	Reset	Description
31:12	*RO*	*0*	*Reserved*
11:08	RO	0h	**Stack Pointer (STACK):** Holds the current stack pointer for the FIQ Processor Stack. Since nesting is not permitted for an FIQ, this field only holds only values of 0000 or 0001. • 0000: Stack is empty; no entries are valid. • 0001: Stack holds one value. • 0010 – 1000: Invalid.
07:03	*RO*	*0*	*Reserved*
02:00	RO	0	**Current Level (CURRLV):** Indicates the current priority level.

6.8.9 Offset 020h: INTC_SWINT – Software Interrupt Register

Bits	Type	Reset	Description
31:00	WO	0	**Enable (EN):** Each bit causes a software interrupt for each of the INTC channels. Bit 0 for channel 0, Bit 1 for channel 1, etc. These bits can be set or cleared by writing directly to it, of by writing to the INTC_SWINT_SET and INTC_SWINT_CLR.

6.8.10 Offset 024h: INTC_SWINT_SET – Software Interrupt Set Register

Bits	Type	Reset	Description

31:00	WO	0	**Set (SET):** Writes of '1' set the corresponding bit in INTC_SWINT.

6.8.11 Offset 028h: INTC_SWINT_CLR – Software Interrupt Clear Register

Bits	Type	Reset	Description
31:00	WO	0	**Clear (CLR):** Writes of '1' clear the corresponding bit in INTC_SWINT

6.8.12 INTC_VECN – Channel N Vector Register

Offset	Channel	Offset	Channel	Offset	Channel	Offset	Channel
080h	0	0A0h	8	0C0h	16	0E0h	24
084h	1	0A4h	9	0C4h	17	0E4h	25
088h	2	0A8h	10	0C8h	18	0E8h	26
08Ch	3	0ACh	11	0CCh	19	0ECh	27
090h	4	0B0h	12	0D0h	20	0F0h	28
094h	5	0B4h	13	0D4h	21	0F4h	29
098h	6	0B8h	14	0D8h	22	0F8h	30
09Ch	7	0BCh	15	0DCh	23	0FCh	31

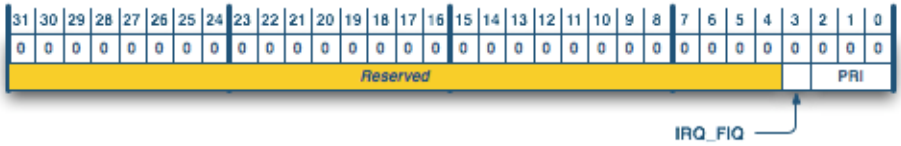
There is one register per interrupt channel, as shown in the table above. The bits in each register are described in the table below.

Bits	Type	Reset	Description
31:00	WO	0	**Vector (VEC):** Contains the value to appear in either INTC_IVEC or INTC_FVEC when channel N is the highest priority active interrupt.

6.8.13 INTC_CFGN – Channel N Configuration Register

Offset	Channel	Offset	Channel	Offset	Channel	Offset	Channel
100h	0	120h	8	140h	16	160h	24
104h	1	124h	9	144h	17	164h	25
108h	2	128h	10	148h	18	168h	26
10Ch	3	12Ch	11	14Ch	19	16Ch	27
110h	4	130h	12	150h	20	170h	28
114h	5	134h	13	154h	21	174h	29
118h	6	138h	14	158h	22	178h	30
11Ch	7	13Ch	15	15Ch	23	17Ch	31

There is one register per interrupt channel, as shown in the table above. The bits in each register are described in the table below.



Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03	RW	0	**IRQ/FIQ (IRQ_FIQ):** When cleared, channel N produces an IRQ. When set, channel N produces an FIQ.
02:00	RW	000	**Priority (PRI):** Specifies the interrupt priority for channel N. • 000: Priority 0 (highest priority) • ... • 111: Priority 7 (lowest priority)

6.8.14 Offset F00h: INTC_IVEC – IRQ Vector Register

Bits	Type	Reset	Description
31:00	RO	Undef	**Interrupt Vector (VEC):** Contains the value of INTC_VECN associated with the highest priority IRQ interrupt. Reading this register pushes the priority of the associated interrupt onto the IRQ Processor Stack (see section 6.6). If no enabled IRQ is active, contains the value of INTC_DFLT.

6.8.15 Offset F04h: INTC_FVEC – FIQ Vector Register

Bits	Type	Reset	Description
31:00	RO	Undef	**Fast IRQ Vector (VEC):** Contains the value of INTC_VECN associated with the highest priority FIQ interrupt. If no enabled FIQ is active, contains the value of INTC_DFLT.

6.8.16 Offset F08h: INTC_IEND – IRQ End-of-Interrupt Register

Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	WO	0	**End (END):** A write to this register pops the IRQ Processor Stack. The written value is discarded and has no other effect.

6.8.17 Offset F0Ch: INTC_FEND – FIQ End-of-Interrupt Register

Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	WO	0	**End (END):** No effect. FIQ cannot be nested.

Chapter 7: External Bus Interface (EBI)

The EBI consists of the following blocks:

- **External Memory Controller (MEMC):** The primary bus supports standard asynchronous memories (such as SRAM, ROM, Flash, etc.), I/O devices, and asynchronous PSRAM.
- **SDRAM Controller:** This provides direct interfacing to standard SDRAM.

MA[23:0], MD[15:0], SA[22:0], and SD[15:0] are all actively driven low whenever idle.

7.1 Asynchronous Memory Controller

This controller interfaces to up to 10 external memory or I/O devices. Features:

- Asynchronous memory support (SRAM, ROM, Flash, PSRAM, Flash etc.)
- Support for I/O devices which utilize an asynchronous SRAM-like interface
- Programmable setup, hold, access and burst timings for each chip select
- Optional ready/wait line for each chip select
- Support for single asynchronous reads and writes, and asynchronous page reads.

7.1.1 Programmable Features

Each chip select has a number of programmable features which can be configured via MEMC_CFGN. Some are listed below.

7.1.1.1 Ready/Wait

The READY pin can be used to extend the read and write access times until signaled by an external slave device. Two configuration bits are provided for each chip select: one enables or ignores the READY pin; the other sets the polarity as READY or nREADY. READY can only be used for the primary and not the secondary or the secondary and not the primary. This is because each bus has its own controller and can act independently of the other (Primary and Secondary accesses can occur at the same time). If HREADY is enabled for Chip Selects on both buses, conflicts can occur since accesses to peripherals which use the HREADY line could occur at the same time. There is no internal arbitration for the HREADY pin. Each Chip Select will respond to the HREADY if programmed to do so.

7.1.1.2 Device Data Width

8-bit and 16-bit devices are supported. Both can be accessed transparently over AHB with byte, half-word, word and burst accesses.

7.1.1.3 Byte Control Style

16-bit devices can be accessed with two write enables and no byte enable, or with a single write enable and two byte enables. The table below shows the muxing which can be done to support these cases.

Table 7-1: Pin Functions vs. Control Style

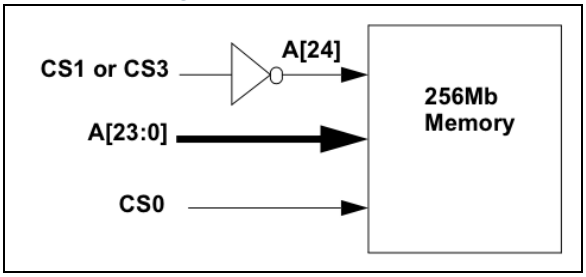
Byte Control Style Settings		Z32AN Series Pin Function		
Width	Style	nWEU	nWEL	A[0]
0 (8-bit)	0 (no BE)	1	nWE	A[0]
	1 (BE)	nBE	nWE	A[0]
1 (16-bit)	0 (no BE)	nWEU	nWEL	0
	1 (BE)	nBEU	nWE	nBEL

7.1.1.4 Extending CS0 memory space

The default memory range for CS0 is16MB or 128Mb is extended to 32MB by using the CS1 address space. MEMC_CFG_1.EXT has been added for CS1 and CS3 only. When ‘0’, CS0 is ‘0’ from address 10000000h – 10FFFFFFh (16MB) as presently specified. When EXT is ‘1’ for CS1, CS0 is ‘0’ for addresses 10000000h – 11FFFFFFh or 32MB (256Mb) and CS1 becomes the 25th address bit, A[24], for either the primary or secondary bus depending on programmable selection. The CS1 pin as A[24] is inverted. When EXT is ‘1’ for CS3, CS0 is ‘0’ for addresses 10000000h - 10FFFFFFh and 13000000h – 13FFFFFFh or 32MB (256Mb) and CS3 becomes the 25th address bit, A[24], for either the primary or secondary bus depending on programmable selection. The CS3 pin as A[24] is inverted.

To use CS1 or CS3 as A[24], an inverter may be added to the output of this signal to address the larger 256Mb memory device.

Figure 7-1: CS1 as A[24]



Operation is the same for CS1 of CS3 except for the address space. CS1 is used here to explain the function. After reset, the boot ROM reads CS0 which sets CS0 =0 and CS1= 1 (default memory configuration). The inverter will drive A24 low, placing the start of the larger memory at 10000000h. The boot ROM checks memory for a valid boot image and if valid, control is turned over to this memory.

When a larger memory device is used (256Mb) then the user code must set MEMC_CFG_1.EXT to ‘1’, to extend the memory range for CS0. This must be done within the first 16MB or 128Mbits of code space.

After MEMC_CFG_1.EXT is set to ‘1’, CS0 is ‘0’ for the entire address space 10000000h – 11FFFFFFh. CS1 is ‘0’ for address spaces 11000000h – 11FFFFFFh. The rest of the time CS1 is ‘1’. Since CS1 behaves as an inverted address bit going into the memory, the external inverter on CS1 will cause correct behavior of A[24] for the address space 10000000h – 11FFFFFFh.

7.1.2 Asynchronous Single Read and Write Transactions

The programmable timing associated with the single reads and writes are shown below. The example provided here has settings of N_{RS}=3, N_{RA}=3, N_{RH}=1, N_{WS}=3, N_{WA}=3, N_{WH}=1, N_{RWI} and READY ignored. hclk is the reference for cycle timing.

Back-to-back accesses will be performed with the chip select remaining constantly active, if the timing of the requests and the programmed parameters allow it.

Figure 7-2: Single Read/Write Timing Diagram

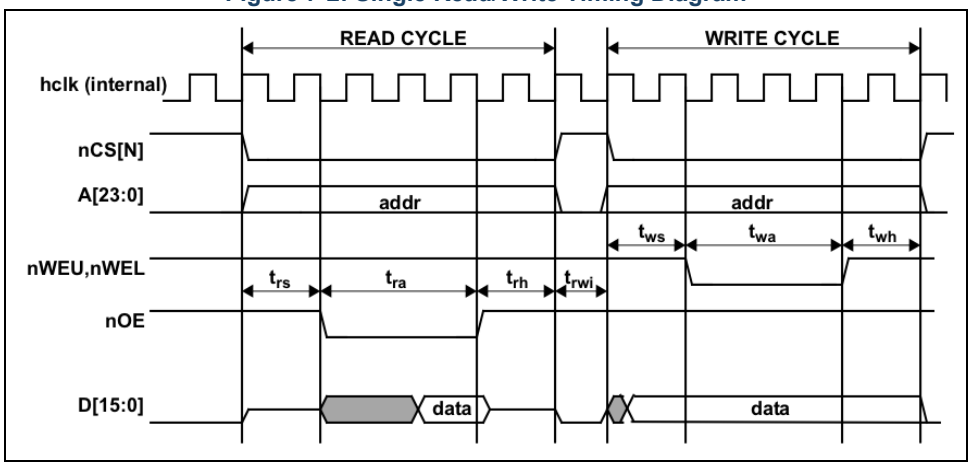


Table 7-2: Single Read/Write Timing (based on MEMC_TIMN)

Symbol	Equation	Description
t_{RS}	$\frac{N_{RS} \times t_{HCLK}}{2}$	**Read Cycle Setup Time:** Specifies the number of half clock cycles of setup time before nOE goes active during a read cycle. Also, this specifies setup time for A and nCS as well as provides time for D to go high-Z.
t_{RA}	$N_{RA} \times t_{HCLK}$	**Read Access Time:** Specifies the number of clock cycles before data is samples and nOE returns to inactive.
t_{RH}	$N_{RH} \times t_{HCLK}$ - or - $(N_{RH} + 1/2) \times t_{HCLK}$	**Read Cycle hold Time:** Specifies the number of clocks that OEn is inactive before the end of the read cycle. This time is increased by a half clock if $N_{RS}[0]$=1.
t_{RWI}	$N_{RWI} \times t_{HCLK}$	**Read/Write Idle Time:** Specifies the minimum number of clocks after a read before a subsequent write cycle, access on a different chip select, or data bus clamping can occur.
t_{WS}	$\frac{N_{WS} \times t_{HCLK}}{2}$	**Write Cycle Setup Time:** Specifies the number of half clock cycles of setup time before nWE goes active during a write cycle. Also, this specifies setup time for A, D and nCS.
t_{WA}	$N_{WA} \times t_{HCLK}$	**Write Access Time:** Specifies the number of clock cycles before data is sampled and nOE returns to inactive.
t_{WH}	$N_{WH} \times t_{HCLK}$ - or - $(N_{WH} + 1/2) \times t_{HCLK}$	**Write Cycle Hold Time:** Specifies the number of clocks that OEn is inactive before the end of the write cycle. This time is increased by a half clock if $N_{WS}[0]$=1.

7.1.3 Asynchronous Page Read Transactions

The programmable timing of asynchronous page reads are shown below. For page reads, the burst is continued until the end of the request is reached, or until the page boundary is reached, whichever occurs first. The example provided here has settings of N_{RS}=3, N_{RA}=3, N_{RPA}=2, N_{RH}=1, N_{RWI}=0 and READY is ignored.

Figure 7-3: Asynchronous Page Read Timing Diagram

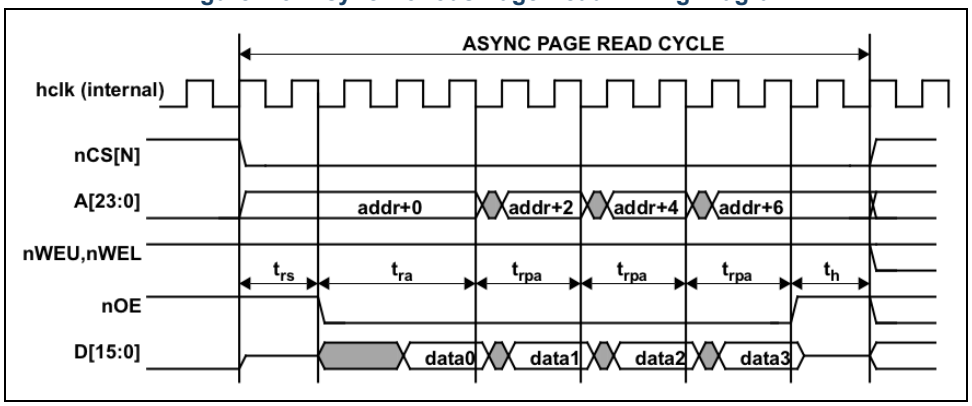


Table 7-3: Asynchronous Page Read Timing (based on MEMC_TIMN)

Symbol	Formula	Description
t_{RS}	$\frac{N_{RS} \times t_{HCLK}}{2}$	**Read Cycle Setup Time:** Same as for single cycle access
t_{RA}	$N_{RA} \times t_{HCLK}$	**Read Access Time:** Same as for single cycle access
t_{RPA}	$N_{RPA} \times t_{HCLK}$	**Read Page Access Time:** Specifies the number of clocks to wait to sample read data again for a sequential page read.
t_{RH}	$N_{RH} \times t_{HCLK}$ **- or -** $(N_{RH} + {}^{1}/_{2}) \times t_{HCLK}$	**Read Cycle Hold Time:** Same as for single cycle access

7.1.4 Clock Divided Transactions

MEMC_CFG.FCLK_DIV divides hclk to allow access to slow peripherals that do not have a READY pin. The example provided here has settings of N_{RS}=1, N_{RA}=3, N_{RH}=1, N_{RWI}=1, N_{WS}=1, N_{WA}=3, N_{WH}=1 and READY ignored.

Figure 7-4: flclk Based Timing Diagram

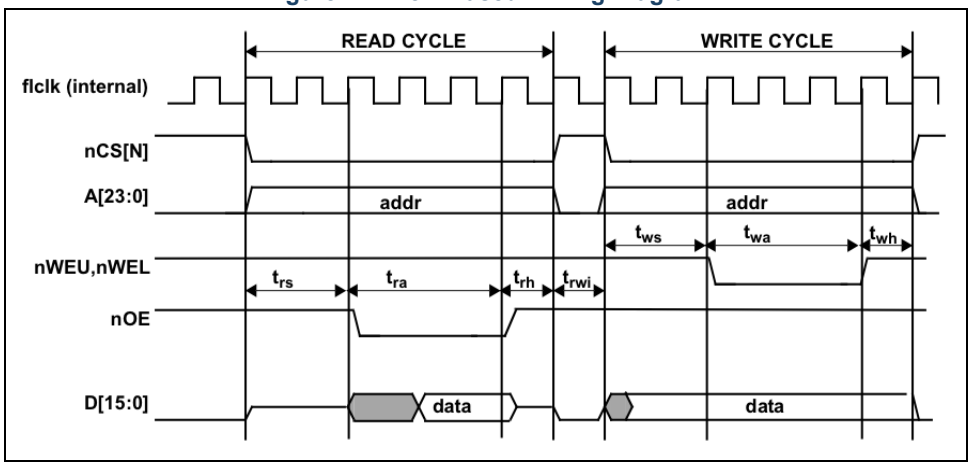


Table 7-4: Extended Timing Parameters (based on MEMC_TIMN)

Symbol	Formula	Description
t_{FLCLK}	$N_{FLKDIV} \times t_{HCLK}$	**Clock Division:** Controls the frequency of flclk by dividing hclk.
t_{RS}	$(N_{RS} \times t_{FLCLK}) + 1$	**Read Setup Time:** Number of flclks before nOE goes active during a read.
t_{RA}	$N_{RA} \times t_{FLCLK}$	**Read Access Time:** Number of flclks after nOE active before first data sampled.
t_{RH}	$N_{RH} \times t_{FLCLK}$	**Cycle Hold Time:** Number of hclks between nOE inactive and end of the cycle.
t_{RWI}	$N_{RWI} \times t_{FLCLK}$	**Read/Write Idle Time:** Number of clocks after a read before a subsequent write cycle, access on a different chip select, or data bus clamping can occur.
t_{WS}	$(N_{WS} \times t_{FLCLK}) + 1$	**Write Setup Time:** Number of clocks before nWE, A, D, and nCS active on a write.
t_{WA}	$N_{WA} \times t_{FLCLK}$	**Write Access Time:** Number of clocks before data is sampled and nOE inactive.
t_{WH}	$N_{WH} \times t_{FLCLK}$	**Write Hold Time:** Number of clocks nOE is inactive before end of write cycle.

7.2 SDRAM Controller

Key features of SDRAM Controller:

- Support for 64 Mb, 128 Mb, 256 Mb and 512 Mb SDRAM devices
- Direct interface for two 16-bit SDRAM devices (if both interfaces are used, the SDRAM used must be the same on each interface)
- Programmable timing parameters
- Support for entering and exiting power-down modes

7.2.1 Operation

Once configured, the SDRAM controller interfaces to a x16 SDRAM configuration and allows any AHB master to access memory transparently. Byte, half-word and word accesses are supported. In addition, burst accesses are supported. AUTO_REFRESH commands are issued automatically based on the user configuration. Power saving modes can be controlled by user commands and configuration.

7.2.2 Address Mapping

SDR_CFG			Memory Address																										
INTER	ROW	COL	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	00	00						B[1:0]		ROW[10:0]												COLUMN[7:0]							
		01					B[1:0]			ROW[10:0]												COLUMN[8:0]							
		10				B[1:0]				ROW[10:0]												COLUMN[9:0]							
		11			B[1:0]					ROW[10:0]												COLUMN[10:0]							
	01	00					B[1:0]			ROW[11:0]												COLUMN[7:0]							
		01				B[1:0]				ROW[11:0]												COLUMN[8:0]							
		10			B[1:0]					ROW[11:0]												COLUMN[9:0]							
		11		B[1:0]						ROW[11:0]												COLUMN[10:0]							
	10	00				B[1:0]				ROW[12:0]												COLUMN[7:0]							
		01			B[1:0]					ROW[12:0]												COLUMN[8:0]							
		10		B[1:0]						ROW[12:0]												COLUMN[9:0]							
		11	B[1:0]							ROW[12:0]												COLUMN[10:0]							
1	00	00								ROW[10:0]										B[1:0]		COLUMN[7:0]							
		01								ROW[10:0]										B[1:0]		COLUMN[8:0]							
		10								ROW[10:0]										B[1:0]		COLUMN[9:0]							
		11								ROW[10:0]										B[1:0]		COLUMN[10:0]							
	01	00								ROW[11:0]										B[1:0]		COLUMN[7:0]							
		01								ROW[11:0]										B[1:0]		COLUMN[8:0]							
		10								ROW[11:0]										B[1:0]		COLUMN[9:0]							
		11								ROW[11:0]										B[1:0]		COLUMN[10:0]							
	10	00								ROW[12:0]										B[1:0]		COLUMN[7:0]							
		01								ROW[12:0]										B[1:0]		COLUMN[8:0]							
		10								ROW[12:0]										B[1:0]		COLUMN[9:0]							
		11								ROW[12:0]										B[1:0]		COLUMN[10:0]							

7.2.3 Supported Configurations

	64 Mb	128 Mb	256 Mb	512 Mb
Configuration	4 Meg x 16	8 Meg x 16	16 Meg x 16	32 Meg x 16
Row Addressing	4K : A[11:0]	4K : A[11:0]	8K : A[12:0]	8K : A[12:0]
Bank Addressing	4 : BA[1:0]	4 : BA[1:0]	4 : BA[1:0]	4 : BA[1:0]
Column Addressing	256 : A[7:0]	512 : A[8:0]	512 : A[8:0]	1K : A[9:0]

7.2.4 SDRAM Performance

Access Type	Page Status	AHB Clocks	Notes
Single Read: Byte, Half-Word or Word	Bank Available	8	
	Page Hit	6	
	Page Miss	11	
4-Burst Read	Bank Available	8-2-2-2	
	Page Hit	6-2-2-2	
	Page Miss	11-2-2-2	
8-Burst Read	Bank Available	8-2-2-2-2-2-2-2	D-Cache or I-Cache line fill
	Page Hit	6-2-2-2-2-2-2-2	
	Page Miss	11-2-2-2-2-2-2-2	
Single Write: Byte, Half-Word or Word	Bank Available	1	
	Page Hit	1	
	Page Miss	1	
4-Burst Write	Bank Available	5-2-2-2	D-Cache half-line dirty
	Page Hit	3-2-2-2	
	Page Miss	8-2-2-2	
8-Burst Write	Bank Available	5-2-2-2-2-2-2-2	D-Cache or I-Cache line fill
	Page Hit	3-2-2-2-2-2-2-2	
	Page Miss	8-2-2-2-2-2-2-2	
Single Read followed immediately by Single Write (same page)	Bank Available	8-2	
	Page Hit	6-2	
	Page Miss	11-2	
Single Write followed immediately by Single Read (same page)	Bank Available	5-5	
	Page Hit	3-5	
	Page Miss	8-5	

7.2.5 Open Bank Policy

The SDRAM Controller leaves all pages open if possible. Banks are closed by page misses and refresh commands.

7.2.6 Power Saving Modes

3 power savings modes are available to the SDRAM controller:

- **Pre-charge Standby:** This state is the same as Active Standby, except that all banks are closed. In this case, nCS = 1 and CKE = 1. To close all banks, This is entered via a PRECHARGE_ALL command to SDR_CMD, or when the refresh timer times out. Apart from periodic refreshes generated by the SDRAM Controller, no commands are issued to the SDRAM. There is no penalty delay incurred when a memory access invokes Active Operation.
- **Pre-charge Power Down:** In this mode, all banks are closed and nCS = 1 and CKE = 0. This state is entered either by a POWER_DOWN command issued to SDR_CMD or by the power-down timer of SDR_APD. In this mode, SDCLK may be configured to be properly disabled and re-enabled automatically.
- **Self Refresh:** In this mode, all banks are closed and nCS = 1 and CKE = 0. This state is entered either by a SELF_REFRESH command issued to SDR_CMD or by the power-down timer of SDR_APD. In this mode, SDCLK may be configured to be properly disabled and re-enabled automatically.

While the SDRAM controller is operating on the external memory (nCS = 0 and CKE = 1), SDRAM is not in a power saving mode. No matter what power down state is active, any access causes an immediate transition to this state. For the Active Standby and Auto Refresh states, there is no performance penalty associated with the access. For the Power-Down and Self-Refresh states, there is a delay to transition to the Active state.

7.2.7 Pin Multiplexing

Multiplexing the bank, row and column bits is described in section 7.2.2. Within the EBI, additional pin multiplexing and de-multiplexing occurs. For more details, see section 7.1.

► ***Note:*** *DQMU, DQML, nRAS, nCAS, and nWE are also multiplexed on Address pins 19-18 and 15-13 for the secondary interface.*

7.2.8 Programmer's Guide

7.2.8.1 Initialization

Initialization is achieved through a sequence of delays and SDRAM commands. This sequence is device dependent, but a general sequence is provided below; this is an example initialization only – refer to the SDRAM datasheet for a more specific initialization sequence.

1. Set the SDRAM clock enable and frequency via PMU registers (Power and SDRAM clocks).
2. Configure the SDRAM Controller by writing to SDR_CFG and SDR_RFSH.
3. Wait for 100 µs (time required for the SDRAM Controller to initialize). During this time, the SDRAM controller will drive CS inactive (COMMAND_INHIBIT).
4. Initiate a PRECHARGE_ALL command to the SDRAM by writing to SDR_CMD.
5. Initiate two AUTO_REFRESH commands to the SDRAM by writing twice to SDR_CMD.
6. Initiate a LOAD_MODE_REGISTER command to the SDRAM by writing to SDR_CMD. By design, the SDRAM Controller waits at least two clocks before issuing a subsequent command.

7.2.8.2 Refresh Control

AUTO_REFRESH cycles are performed automatically at a period based upon SDR_RFSH. The setting of the timer register depends on SDCLK frequency as well as the number of rows and t_{REF} for the SDRAM. The register must be set according to the following equation:

$$\frac{t_{REF}}{(Number\ of\ Rows)} > (SDR\_RFSH \times t_{SDCLK})$$

Additionally, the user can perform a refresh by sending the AUTO_REFRESH command of SDR_CMD.

7.2.8.3 Zeroization Command

The SDRAM Controller is capable of clearing all SDRAM memory locations to 0 by issuing a ZEROIZATION command to SDR_CMD. During this operation, all locations are cleared at an approximate rate of one half-word per SDCLK cycle. Once started, you cannot stop this command before it is completed.

7.3 Example Configurations

Below is a diag1./ram showing an example configuration, and the register settings for that configuration. The register programming for this configuration is:

- SDR_CMD.SEC_SDRM_EN set to '1' to enable use of the secondary bus and nCS[1] for SDRAM
- MEMC_CFG_2.PRI_SEC cleared to '0' (primary bus)
- MEMC_CFG_0.PRI_SEC cleared to '0' (primary bus)

Figure 7-5: External Memory Example

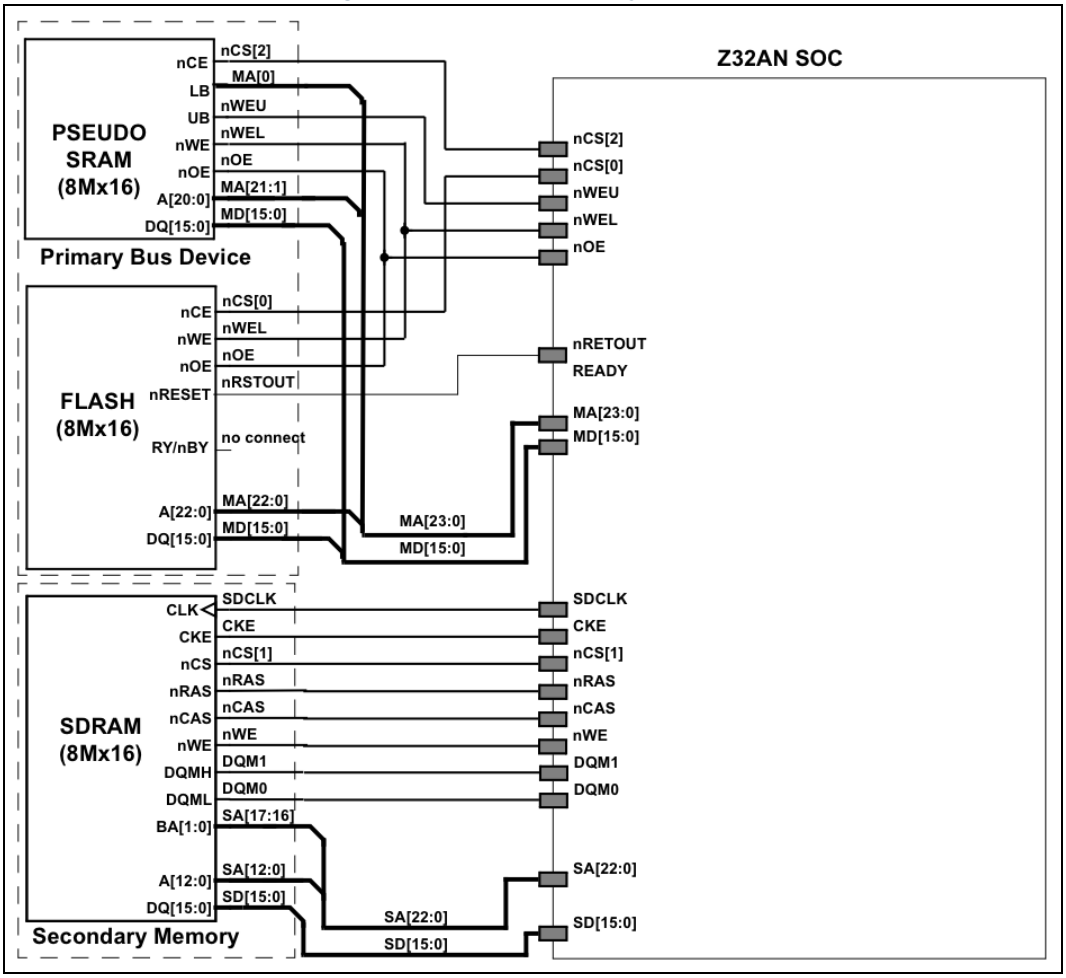


Figure 7-6: Connection to an 8-bit SRAM Device

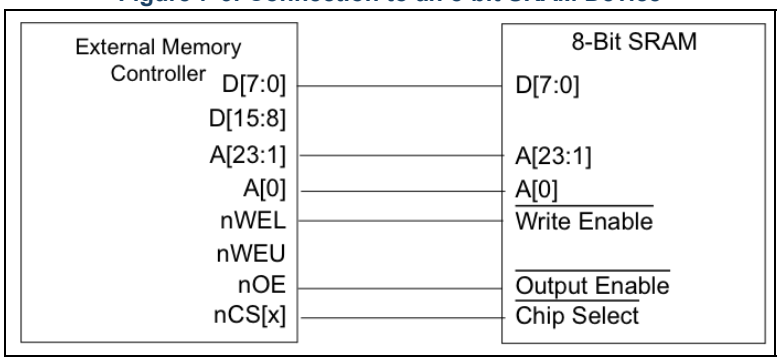


Figure 7-7: Connection to a 16-bit SRAM Device

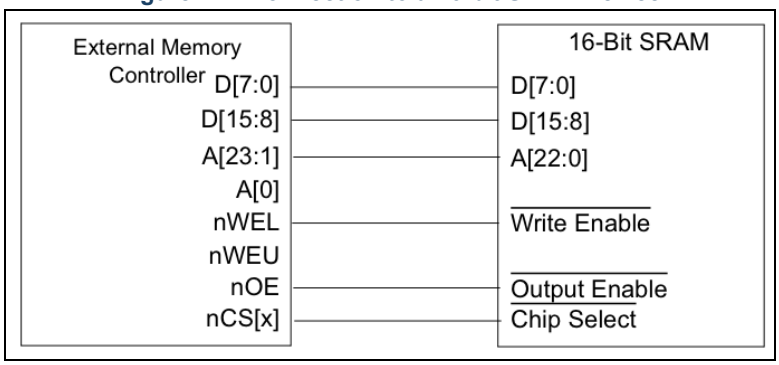


Figure 7-8: Connection to a 16-bit SRAM Device with Byte Enable

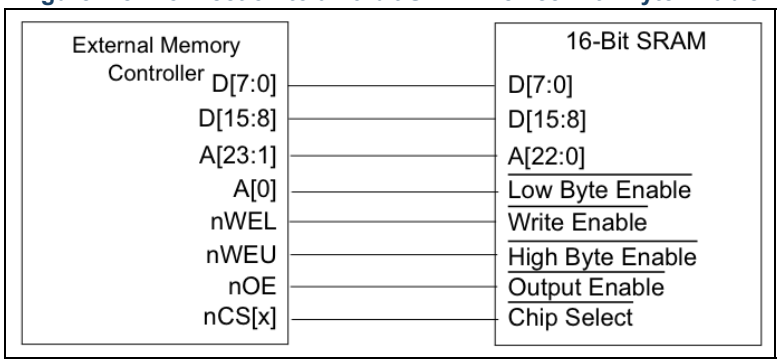


Figure 7-9: Connection to 2 x 8-bit SRAM Devices

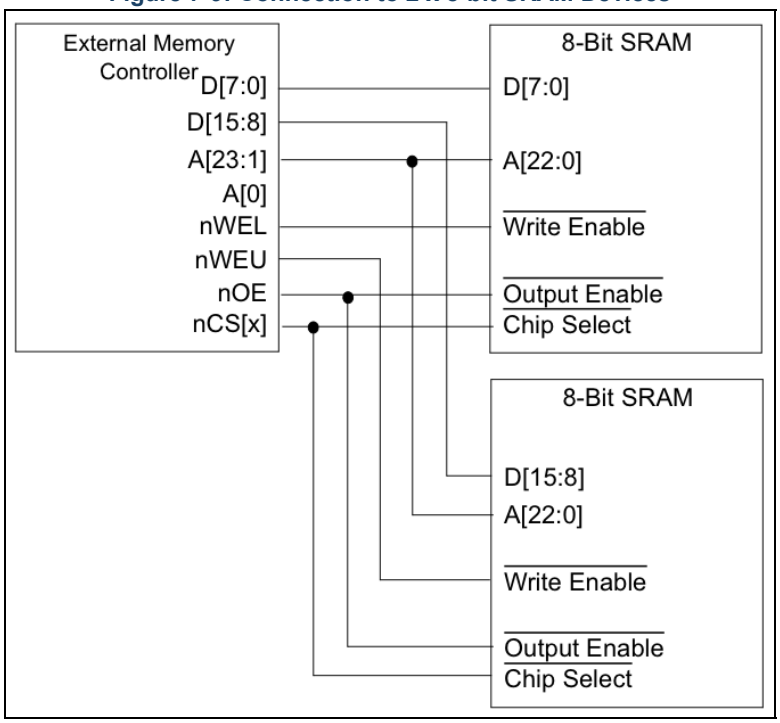


Figure 7-10: Connection to an 8-bit FLASH Device

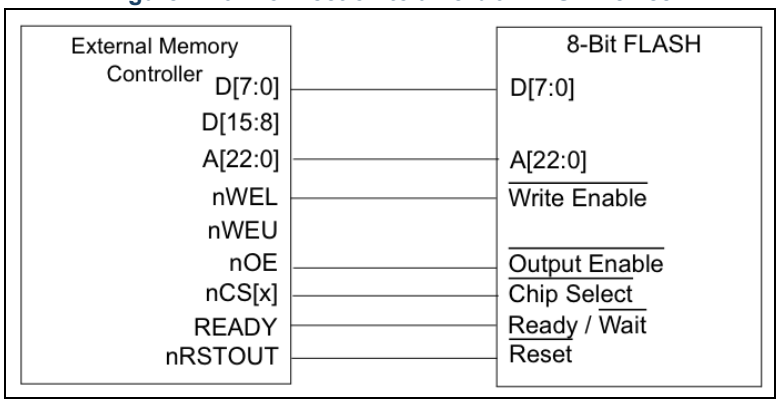


Figure 7-11: Connection to a 16-bit FLASH Device

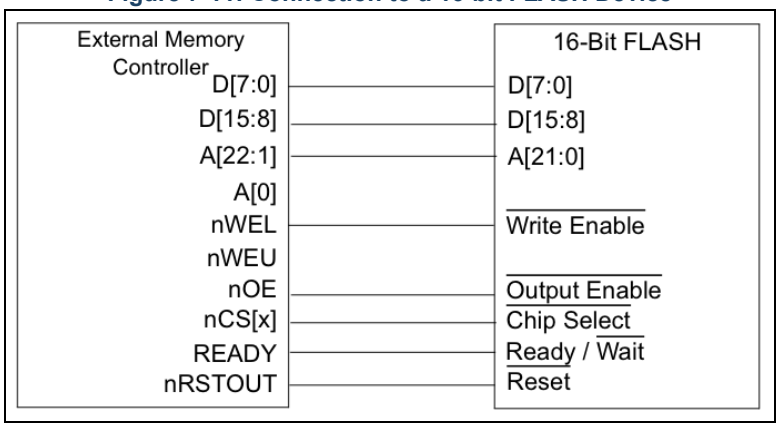


Figure 7-12: Sync Burst Flash Configuration (AM29BL802C)

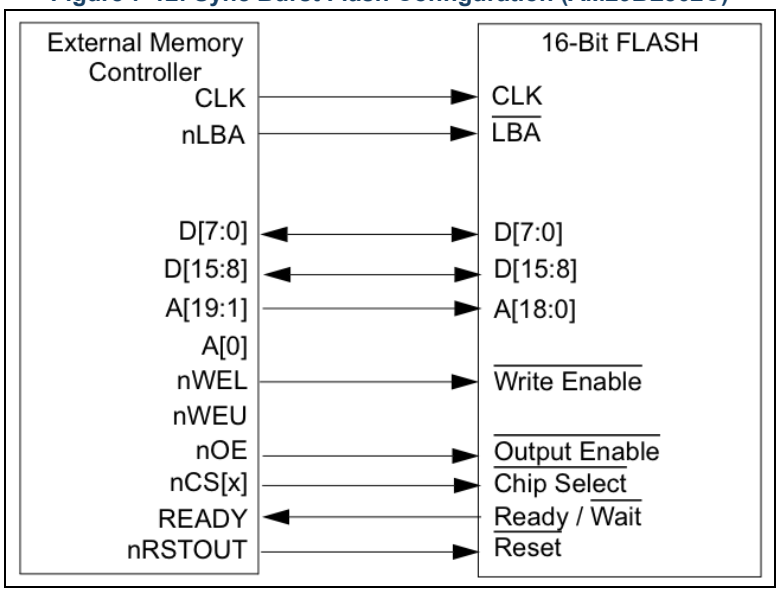
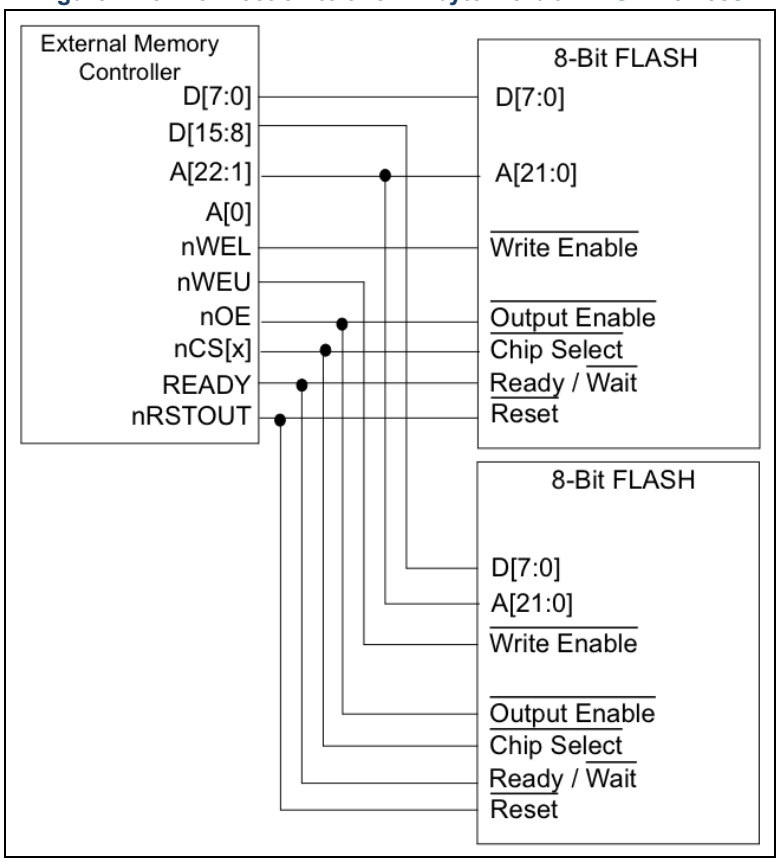


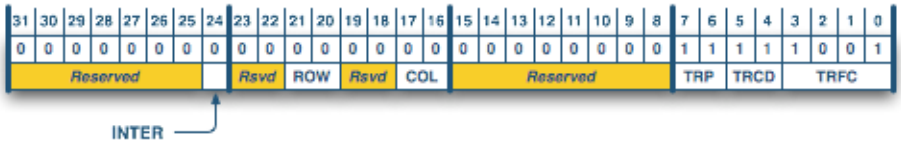
Figure 7-13: Connection to two 4M byte x 8-bit FLASH Devices



7.4 Registers (Base → FFFF8000h)

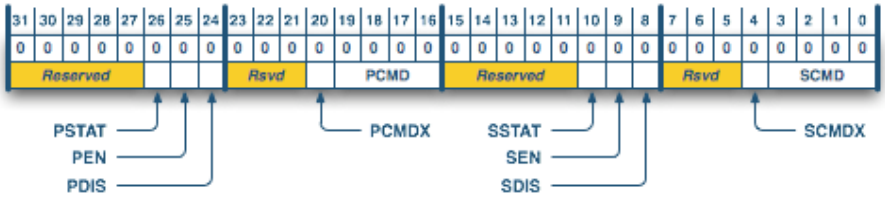
Offset	Register	Description
000h	SDR_CFG	SDRAM Configuration Register
004h	SDR_CMD	SDRAM Command Register (for both interfaces)
008h	SDR_RFSH	SDRAM Refresh Register
00Ch	SDR_APD	SDRAM Automatic Power-Down Register
010h	MEMC_GCFG	Global Configuration Register
014h – 038h	MEMC_CFGn	nCS[“n”] Configuration Register
03Ch – 060h	MEMC_TIMn	nCS[“n”] Timing Register
064h	MEMC_STA	Status Register

7.4.1.1 Offset 000h: SDR_CFG – SDRAM Configuration Register



Bits	Type	Reset	Description
31:25	*RO*	*0*	*Reserved*
24	RW	0	**Interleave Address (INTER):** When cleared, bank address derived from MSBs. When set, bank address derived from address bits between the column (LSBs) and the row (MSBs). See section 7.2.2.
23:22	*RO*	*00*	*Reserved*
21:20	RW	00	**Row Width (ROW):** Row bits. See section 7.2.2. • 00: 11 bits • 01: 12 bits • 10: 13 bits • 11: reserved
19:18	*RO*	*00*	*Reserved*
17:16	RW	00	**Column Width (COL):** Column bits. See section 7.2.2. • 00: 8 bits • 01: 9 bits • 10: 10 bits • 11: 11 bits
15:08	*RO*	*0*	*Reserved*
07:06	RW	11	**Minimum PRECHARGE Delay (TRP):** Minimum delay from PRECHARGE to any other command to the same bank: The SDRAM Controller will guarantee the specified number of SDCLK cycles between PRECHARGE and any subsequent command to the same bank. • 00: 1 clock • 01: 2 clocks • 10: 3 clocks • 11: 4 clocks
05:04	RW	11	**Minimum ACTIVE Delay (TRCD):** Minimum delay from ACTIVE to READ or WRITE command. The SDRAM Controller will guarantee the specified number of SDCLK cycles between ACTIVE and READ/WRITE commands. • 00: 1 clock • 01: 2 clocks • 10: 3 clocks • 11: 4 clocks
03:00	RW	9h	**Minimum AUTO_REFRESH Period (TRFC):** The SDRAM Controller will guarantee the specified number of SDCLK cycles between AUTO REFRESH commands. • 0000: 1 clock • 0001: 2 clocks • ... • 1001: 10 clocks • 1010 - 1111: reserved

7.4.1.2 Offset 004h: SDR_CMD – SDRAM Command Register

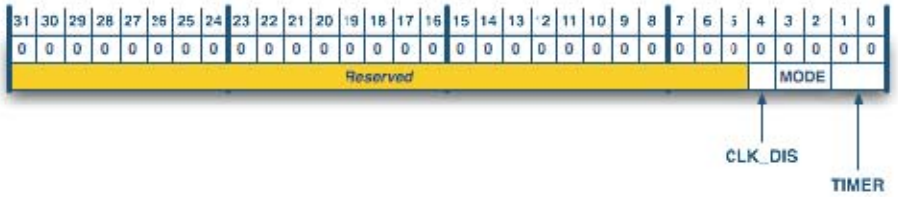


Bits	Type	Reset	Description
31:27	*RO*	*0*	*Reserved*
26	RO	0	**Primary Status (PSTAT):** When set, SDRAM controller enabled
25	WO	0	**Primary Controller Enable (PEN):** When written to '1', enables the primary SDRAM controller
24	WO	0	**Primary Controller Disable (PDIS):** When written to '1', disables the primary SDRAM Controller.
23:21	*RO*	*0*	*Reserved*
20	RW	0	**Primary Command Execute (PCMDX):** When written to '1', invokes the command specified in PCMD. Writes of '0' have no effect. When read as '1', command execution is pending. When read as '0' command is completed.
19:16	RW	0h	**Primary Command (PCMD):** Specifies the command to be executed PCMDX is set. • 0000: Normal Operation • 0001: NOP • 0010: PRECHARGE_ALL • 0011: LOAD_MODE_REGISTER • 0100: SELF_REFRESH • 0101: POWER_DOWN • 0110: AUTO_REFRESH • 0111: SDRAM_Zeroization • 1000 - 1111: Reserved
15:11	*RO*	*0*	*Reserved*
10	RO	0	**Secondary Status (SSTAT):** See PSTAT.
09	WO	0	**Secondary Controller Enable (SEN):** See PEN.
08	WO	0	**Secondary Controller Disable (SDIS):** See PDIS.
07:05	*RO*	*0*	*Reserved*
04	RW	0	**Secondary Command Execute (SCMDX):** See PCMDX.
03:00	RW	0h	**Secondary Command (SCMD):** See PCMD.

7.4.1.3 Offset 008h: SDR_RFSH – SDRAM Refresh Register

Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**Refresh Timer (TRFC):** Sets the period for AUTO_REFRESH commands generated automatically by the SDRAM Controller in SDCLKs. Set to 0 to disable automatic refresh.

7.4.1.4 Offset 00Ch: SDR_APD – SDRAM Automatic Power-Down



Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RW	0	**SDCLK Disable (CLK_DIS):** Specifies whether SDCLK must be disabled (gated) as part of the automatic power down sequence. • 0: Do not disable SDCLK automatically • 1: Disable SDCLK upon automatic power down time-out.
03:02	RW	00	**Mode (MODE):** Selects type of power down mode of the SDRAM Controller. • 00: Disable Automatic Power Down • 01: Automatic Pre-charge Power Down • 10: Automatic Active Power Down • 11: Automatic Self Refresh
01:00	RW	00	**Timer (TIMER):** Specifies the number of SDCLKs after an access of SDRAM before initiating the action specified in MODE. • 00: 64 SDCLK cycles • 01: 128 SDCLK cycles • 10: 256 SDCLK cycles • 11: 512 SDCLK cycles

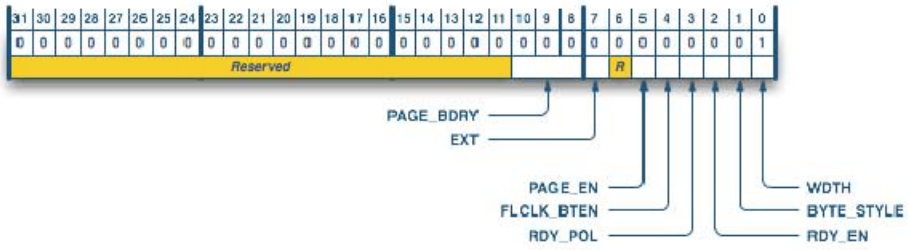
7.4.1.5 Offset 010h: MEMC_GCFG – Memory Controller Global Configuration

Bits	Type	Reset	Description
31:28	*RO*	*0*	*Reserved*
27:16	RW	FFFh	**Ready Time-out (READY_TIMEOUT):** Global READY time-out limit. Time-out results in an AHB bus error. 0-4k hclk cycles. 0-4k hclk cycles.
15:05	*RO*	*0*	*Reserved*
04	RW	0	**Bus Zero Mode (BUS_ZERO):** When set, MD[15:0] and SD[15:0] are driven "0" when SDRAM and Memory Controller state machines are idle. When cleared, MD[15:0] and SD[15:0] are floating when SDRAM and Memory Controller State Machines are idle.
03:02	RW	00	**FLCLK Divider (FLCLK_DIV):** Specifies the divider to derive flclk from hclk: • 00: divide hclk by 2 • 01: divide hclk by 4 • 10: divide hclk by 6 • 11: divide hclk by 8
01:00	*RO*	*0*	*Reserved*

7.4.1.6 MEMC_CFGn – Memory Controller nCS[“n”] Configuration Registers

Offset	Chip Select	Offset	Chip Select
014h	0	028h	5
018h	1	02Ch	6
01Ch	2	030h	7
020h	3	034h	8
024h	4	038h	9

There are 10 memory controller chip select configuration registers. The above table lists the offsets, and the table below describes the bits in each register.



Bits	Type	Reset	Description
31:11	*RO*	*0*	*Reserved*
10:08	RW	000	**Page Boundary (PAGE_BDRY):** Specifies the boundary limit for sync or async burst reads. This is defined in terms of the least significant AHB address bit which must remain constant for the burst to continue. • 000: A[2] • 001: A[3] • 010: A[4] • 011: A[5] • 100: A[6] • 101: A[7]
07	RW	0	**Extend (EXT):** When set, CS1 or CS3 becomes MA[24] and CS0 address space is 32MB. When cleared, CS1 or CS3 used normally. **Note:** Can use CS1 only if there is no SDRAM.
06	*RO*	*0*	*Reserved*
05	RW	0	**Page Mode (PAGE_EN):** When set, enables asynchronous page reads for this device.
04	RW	0	**FLCLK Based Timing Enable (FLCLK_BTEN):** When set, enables the clock divider for this external device. Uses the flclk divider to provide longer clock cycles for slow peripherals. When cleared, flclk Timing disabled.
03	RW	0	**READY Pin Polarity (RDY_POL):** When set, ‘1’ means ready. When cleared, VSS means ready.
02	RW	0	**READY Pin Enable (RDY_EN):** When set, READY indicates cycle completion. When cleared, READY pin ignored
01	RW	0	**Byte Control Style (BYTE_STYLE):** Specifies the style of the control bits used. This is used to provide a seamless solution for interfaces: • 16-bit using 2xWE and no BE, 16-bit using 1xWE and 2xBE, 8-bit using 1xWE and no BE, 8-bit using 1xWE and 1xBE16-bit device: • When cleared, nWEU=nWEU, nWEL=nWEL, A[0] = unused, held low • When set, nWEU=nBEU, nWEL=nEW, A[0] = nBEL • 8-bit device: • When cleared, nWEU= unused, held high, nWEL=nWE, A[0] = A0 • When set, nWEU=nBE, nWEL=nWE, A[0] = A0
00	RW	1	**Data Bus Width (WDTH):** When cleared, external device width is 8 bits. When set, external device width is 16-bits.

7.4.1.7 MEMC_TIMn – Memory Controller nCS[“n”] Timing Registers

Offset	Chip Select	Offset	Chip Select
03Ch	0	050h	5
040h	1	054h	6
044h	2	058h	7
048h	3	05Ch	8
04Ch	4	060h	9

There are 10 memory controller chip select timing registers. The above table lists the offsets, and the table below describes the bits in each register. All timing parameters are based on hclk when FLCLKBTEN = ‘0’, and based on flclk when FLCLKBTEN = ‘1’.

Bits	Type	Reset	Description
31:30	*RO*	*0*	*Reserved*
29:27	RW	111	**NRWI:** Number of hclks of the minimum time after a read before a subsequent write cycle, an access on a different chip select, or data bus clamping can occur. • FLCLKBTEN = ’0’ → 000: 0 hclk cycles, 001: 1 hclk cycles, ... 111: 7 hclk cycles • FLCLKBTEN = ’1’ → 000: 1 flclk cycle, 001: 2 flclk cycles, ... 111: 7flclk cycles
26:24	RW	111	**NWH:** Number of clocks of the write cycle hold time. • FLCLKBTEN=0 & NWS[0] = 0 → 000: 0 hclks, 001: 1 hclk, ... 111: 7 hclks • FLCLKBTEN=0 & NWS[0] = 1 → 000: $^1/_2$ hclks, 001: $1^1/_2$ hclks, 010: $2^1/_2$ hclks, ..., 111: $7^1/_2$ hclks • FLCLKBTEN=1 → 000: 1 flclk, 001: 2 flclks, ..., 111: 8 flclks
23:20	RW	Fh	**Write Access Time (NWA):** Number of clocks: • FLCLKBTEN = ’0’ → 0000: 1 hclk cycle, 0001: 2 hclk cycles, ... 1111: 16 hclk cycles • FLCLKBTEN = ’1’ → 0000: 1 flclk cycle, 0001: 2 flclk cycles, ... 1111: 16 flclk cycles
19:16	RW	Fh	**NWS:** Number of clocks for write cycle setup time (before nWE goes active). • FLCLKBTEN=’0’ => 0h: 0 hclks, 1h: $^1/_2$ hclks, 2h: 1 hclks, 3h: $1^1/_2$ hclks..., Fh: $7^1/_2$ hclk cycles • FLCLKBTEN=1 => 0000: 1 flclk cycle, 0001: 2 flclk cycles, ..., 1111: 16 flclk cycles
15	*RO*	*0*	*Reserved*
14:11			**NRPA:** Number of clocks for read access time of sequential access of a page burst read. • 0000: 1 cycle, 0001: 2 cycles, 0010: 3 cycles, ... 1111: 16 cycles
10:08	RW	111	**NRW:** Number of hclks of the read cycle hold time. • FLCLKBTEN=0 & Page=0 & NRS[0]=0 → 000: 0 hclks, 001: 1 hclk, ..., 111: 7 hclks • FLCLKBTEN=0 & Page=0 & NRS[0]=1 → 000: $^1/_2$ hclk, 001: $1^1/_2$ hclks, 010: $2^1/_2$ hclks, ..., 111: $7^1/_2$ hclks • FLCLKBTEN=0 and Page=1 → 000: 0 hclk cycle, 001: 1 hclk cycles, ..., 111: 7 hclk cycles • FLCLKBTEN=1 → 000: 0 flclk cycle, 001: 1 flclk cycles, ..., 111: 7 flclk cycles
07:04	RW	Fh	**Read Access Number (NRA):** Number of clocks for read access time. • FLCLKBTEN=0 → 0h: 1 hclk, 1h: 2 hclks, ..., Fh: 16 hclks • FLCLKBTEN=1 → 0h: 1 flclk, 1h: 2 flclks, ..., Fh: 16 flclks

03:00	RW	Fh	**NRS:** Number of clocks for read cycle setup time (before nOE goes active). • FLCLKBTEN=0 & Page=0 → 0h: 0 hclks, 1h: $^1/_2$ hclk, 2h: 1 hclk, ... Fh: $7^1/_2$ hclks • FLCLKBTEN=0 & Page=1 → 0h: 1 hclk, 1h: 2 hclks, ..., 16h: 16 hclks • FLCLKBTEN=1 → 0h: 1 flclk, 1h: 2 flclks, ..., Fh: 16 flclks

7.4.1.8 Offset 064h: MEMC_STA – Memory Controller Status Register

Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	RW1C	0	**Write Error Status (WR_ERR):** When cleared, no write error was detected during Ready Mode. When set, indicates a write error was detected during Ready Mode.

Chapter 8: DMA Controller

The DMA Controller is an AHB device that provides eight fully programmable, chaining capable DMA channels that can transfer from peripheral-to-memory, memory-to-memory, or memory-to-peripheral. All transactions consist of an AHB burst read into a DMA FIFO followed by an AHB burst write from the FIFO. Each channel has the following features:

- Full 32-bit source and destination addresses with 24-bit (16 MB) address increment capability
- Up to 16 MB for each DMA buffer
- Programmable burst size
- Programmable priority
- Interrupt upon count-to-zero
- Abort on error

8.1 Channel Arbitration and Bursts

Once a channel has been programmed, it generates a request either immediately (memory-to-memory) or when its associated peripheral requests DMA (memory-to-peripheral or peripheral-to-memory). The arbiter grants on the basis of priority – a higher priority request is always granted. Within a priority level, requests are granted on a round-robin basis. Once a channel's request has been granted, it executes two steps:

- Burst movement of data from the source device into the FIFO.
- Burst movement of data from the FIFO to the destination device.

Any required data alignment is achieved through the FIFO. Once granted, only an error condition interrupts execution. The occurrence of a higher priority request will not. Once both steps are complete, re-arbitration occurs. The fields EN, REQ, and PRI in DMA_CFGN are involved in arbitration.

8.2 DMA Source and Destination Addressing

For memory, DMA_SRCN/DMA_DESTN are addresses of the source and destination. For peripherals, all or part of the address is fixed based upon DMA_CFGN.REQ. The table below shows how the source and destination addresses as well as the address increment controls are constructed based on DMA_CFGN.REQ. A "P" in SINCR or DINCR indicates the field is programmable, while a '0' indicates the field is forced to zero.

Table 8-1: Source and Destination Address Construction

REQ	Transfer	Source Address[31:0]	SINCR	Destination Address[31:0]	DINCR
00h	Mem-to-Mem	00b:DMA_SRC[29:0]	P	00b:DMA_DEST[29:0]	P
01h	Rx UART 0	FFFE0000h	0		
02h	Rx UART 1	FFFE1000h	0		
03h	Rx UART 2	FFFE2000h	0		
04h	Rx SPI 0	FFFEE000h	0		
05h	Rx Smart Card	FFFF0h:DMA_SRC[11:0]	0		
06h	*Reserved*				
07h	Rx Ext Req	00b:DMA_SRC[29:0]	P	00b:DMA_DEST[29:0]	P
08h	Rx ADC	FFFF2008h	0		
09h	Rx MCR	FFFF300Ch	0		
0Ah	Rx SPI 1	FFFEF000h	0		
0Bh – 10h	*Reserved*				
11h	Tx UART 0	00b:DMA_SRC[29:0]	P	FFFE0000h	0
12h	Tx UART 1			FFFE1000h	0
13h	Tx UART 2			FFFE2000h	0
14h	Tx SPI 0			FFFEE000h	0
15h	Tx Smart Card			FFFF0h:DMA_DEST[11:0]	0
16h	*Reserved*				
17h	Tx Ext Req	00b:DMA_SRC[29:0]	P	00b:DMA_DEST[29:0]	P
18h	Tx SHA-1			FFFF9014h	0
19h	Tx LCD			FFFED008h	0
1Ah	Tx SPI 1			FFFEF000h	0
1Bh – 1Fh	*Reserved*				

The registers and fields involved with source data movement are:

- DMA_SRCN → All bits
- DMA_CNTN → All bits
- DMA_CFGN → BURST, SWIDTH, SINCR

The table below depicts how data is moved into the DMA FIFO based on the settings of DMA_SRCN[1:0] and DMA_CFGN.SWIDTH. If the width of the device is larger than the current value of DMA_CNTN, the DMA Controller performs an AHB cycle to a smaller width.

Table 8-2: Inbound Data Alignment

Source Device					Register Field Values				Resulting AHB Burst
Word Size	Active Bytes				SWIDTH		DMA_SRCN		
	3	2	1	0	1	0	1	0	
8				X	0	0	0	0	One AHB byte read for each byte moved. Only data from the indicated byte lane will be moved into the DMA FIFO (LS byte first).
			X				0	1	
		X					1	0	
	X						1	1	
16			X	X	0	1	0	0	One AHB half-word read for each half-word moved. Only data from the indicated byte lanes will be moved into the DMA FIFO (least significant half-word first).
	X	X					1	0	
32	X	X	X	X	1	0	0	0	One AHB word read for each word moved. The entire word is moved into the DMA FIFO.

8.3 Data Movement from the DMA FIFO to the Destination

The following registers and fields affect destination data movement of the DMA transfer:

- DMA_DESTN → All bits
- DMA_CNTN → All bits
- DMA_CFGN → BURST, DWIDTH, DINCR

The table below depicts how data is moved out of the FIFO based on the settings of DMA_FIFO_COUNTER, DMA_DESTN[1:0] and DMA_CFGN.DWIDTH. If the width of the device is larger than the number of bytes left in the FIFO, the DMA Controller performs an AHB cycle to a smaller width.

Table 8-3: Outbound Data Alignment

Destination Device					Register Field Values				Resulting AHB Burst
Data Size	Active Bytes				DWIDTH		DMA_DESTN		
	3	2	1	0	1	0	1	0	
8				X	0	0	0	0	One AHB byte write for each byte moved. Bytes will be moved out of the DMA FIFO (LS byte first) onto the indicated byte lane.
			X				0	1	
		X					1	0	
	X						1	1	
16			X	X	0	1	0	0	One AHB half-word write for each half-word moved. Half-words will be moved out of the DMA FIFO (least significant half-word first) onto the indicated byte lanes.
	X	X					1	0	
32	X	X	X	X	1	0	0	0	One AHB word write for each word moved. The entire word is moved out of the DMA FIFO.

8.4 Memory Buffer Alignment

The DMA controller adjusts the transfer size to provide correct buffer alignment.

8.5 Count-to-Zero Condition

When a channel AHB burst completes, the DMA controller checks to see if DMA_CNTN has been decremented to 0. If so, there are two possible responses:

- If DMA_CRLDN.EN is set, DMA_SRCN, DMA_DESTN, and DMA_CNTN are loaded from the reload registers and the channel remains active using the newly loaded address/count values and the previously programmed configuration values.
- If DMA_STAN.RLOAD is cleared, the channel is disabled and DMA_STAN.EN is cleared to '0'.

8.6 Chaining Buffers

The reload registers can be used for chaining buffers together. This allows the DMA to continue to service a request without immediate processing from the CPU. When a count-to-zero condition occurs with DMA_CRLDN.EN set to '1', the channel remains active, the DMA_SRCN, DMA_DESTN, and DMA_CNTN is loaded with values from the reload registers, and the DMA operation continues with the new DMA buffer. To prevent improper operation, program the address before setting DMA_CRLDN.EN. The following fields affect buffer chaining.

- DMA_SRLDN → All bits
- DMA_DRLDN → All bits
- DMA_CRLDN → EN, COUNT

8.7 DMA Interrupts

The following registers and fields control DMA interrupts.

- DMA_CTRL → IENx
- DMA_ISTAT → PENDx
- DMA_CFGN → CTZ_IEN, STA_IEN
- DMA_STAN → EN, IPEND, CTZ, RLOAD, BUS_ERR, TO

8.8 Channel Time-outs

Each channel can be configured to generate an interrupt when its associated request line is inactive. An example of this feature is to determine an idle UART receive channel. The time-out mechanism consists of a single global 24-bit pre-scalar and per-channel programmable 8-bit timers. The global pre-scalar has 3 taps: a divide by 256, a divide by 64k, or a divide by 16M.

Each channel's 8-bit timer increments using a tap chosen by DMA_CFGN.PS. DMA_CFGN.TO selects how high the timer must count before generating an interrupt. The timer is reset whenever any of the following conditions occurs:

- The DMA request line programmed for the channel is activated.
- The channel is disabled for any reason (DMA_STAN.EN is zero).

Any timer can be disabled by clearing DMA_CFGN.PS to '00'. When all 8 channels are configured, the global pre-scalar is disabled. Normally, the timer starts counting as soon as the channel is enabled and DMA_CFGN.PS is non-zero. But if DMA_CFGN.WAIT is set, the timer starts counting only after the first DMA request is received from the peripheral. The time-out period can be calculated using the following equation: $T_{TIMEOUT} = T_{HCLK} \times N_{PS} \times N_{TO}$

With hclk frequency of 90 MHz and PS=10b and TO=100b, the time-out period is:

$$T_{TIMEOUT} = \frac{65,536 \times 64}{90\ MHz} = 46.6ms$$

The following registers and fields control channel time-outs.

- DMA_CTRL → IENx
- DMA_CFGN → PS, TO, WAIT
- DMA_STAN → TO

8.9 Register Accesses Restrictions

Any register can be written while a channel is disabled. When DMA_STAN.EN is set to '1', the channel is enabled. As an active channel can be in the middle of read/write burst, DMA_SRCN, DMA_DESTN, or DMA_CNTN must not be written. A DMA channel can be disabled by clearing DMA_CFGN.EN. DMA_STAN.EN must be polled to verify that the channel is disabled. When clearing DMA_CFGN.EN, perform a read-modify-write to ensure that other bits of DMA_CFGN are not modified. A channel is automatically disabled if there is an AHB bus error, or a count-to-zero condition occurs with DMA_CRLDN.EN cleared. If these occur, EN and EN of DMA_CFGN are cleared automatically.

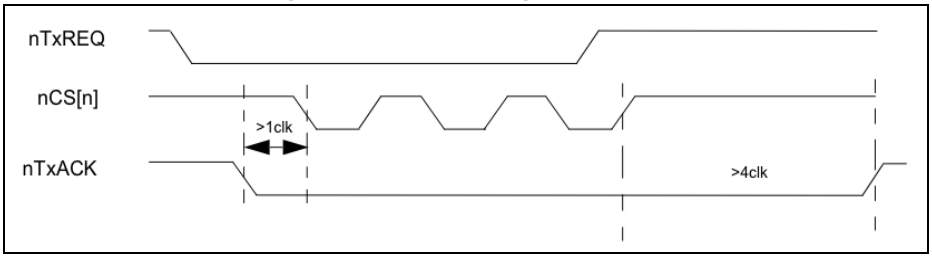
8.10 Memory-to-Memory DMA

Memory-to-memory transfers are performed as if the request is always active. Therefore, assign a lower priority to channels executing memory-to-memory transfers to prevent starvation of other DMA channels.

8.11 External DMA

An external DMA transfer is initiated by asserting nTxREQ. Figure 8-1 displays the nTxACK waveform to an external device. nTxACK is asserted at least 1 hclk before nCS, and de-asserts at least 4 hclks after the de-assertion of nCS. The polarity is selected in DMA_CTRL.

Figure 8-1: Acknowledge Waveform



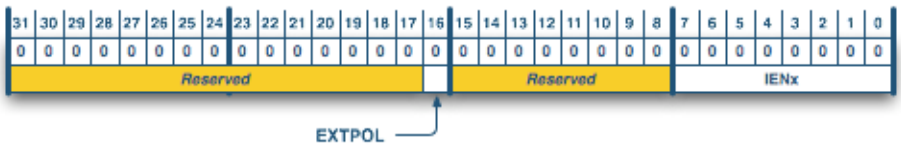
Note: The above figure assumes active low signals

8.12 Registers (Base → FFFF4000h)

8.12.1 Global Registers

Offset	Register	Description
000h	DMA_CTRL	DMA Control Register
004h	DMA_ISTAT	DMA Interrupt Status Register

8.12.1.1 Offset 000h: DMA_CTRL – DMA Control Register



Bits	Type	Reset	Description
31:17	*RO*	*0*	*Reserved*
16	RW	0	**External DMA Polarity (EXTPOL):** Sets the polarity for ALL external DMA inputs and outputs (TxREQ, TxACK, RxREQ, RxACK). When set, the polarity is active high.
15:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Channel "N" Interrupt Enable (IENx):** When set, the interrupt for that channel is enabled. Bit 7 = channel 7, bit 0 = channel 0.

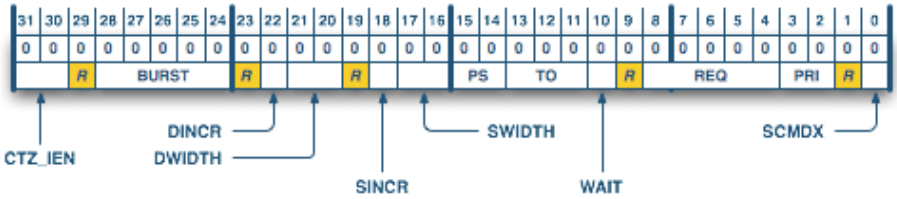
8.12.1.2 Offset 004h: DMA_ISTAT – DMA Interrupt Status Register

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RO	00h	**Channel Interrupt Pending (PENDx):** When set, an interrupt is pending for the channel. More information in DMA_STAN, and all active bits of DMA_STAN must be cleared for this bit to clear. These bits are set only DMA_CTRL.IENx is set.

8.12.2 Per-Channel Registers

Offsets								Register
Ch0	Ch1	Ch2	Ch3	Ch4	Ch4	Ch6	Ch7	
100h	120h	140h	160h	180h	1A0h	1C0h	1E0h	DMA_CFGN
104h	124h	144h	164h	184h	1A4h	1C4h	1E4h	DMA_STAN
108h	128h	148h	168h	188h	1A8h	1C8h	1E8h	DMA_SRCN
10Ch	12Ch	14Ch	16Ch	18Ch	1ACh	1CCh	1ECh	DMA_DESTN
110h	130h	150h	170h	190h	1B0h	1D0h	1F0h	DMA_CNTN
114h	134h	154h	174h	194h	1B4h	1D4h	1F4h	DMA_SRLDN
118h	138h	158h	178h	198h	1B8h	1D8h	1F8h	DMA_DRLDN
11Ch	13Ch	15Ch	17Ch	19Ch	1BCh	1DCh	1FCh	DMA_CRLDN

8.12.2.1 DMA_CFGn – DMA Channel "n" Config Register

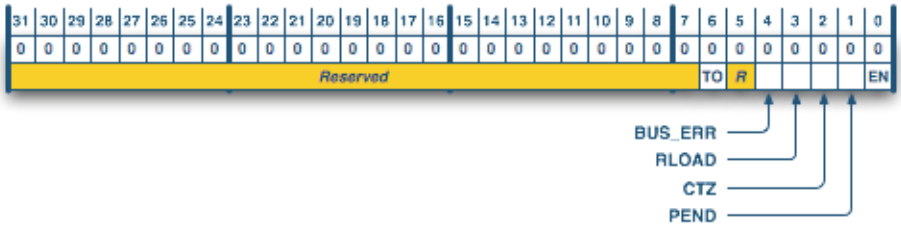


Bits	Type	Reset	Description
31	RW	0	**Count-to-Zero Interrupt Enable (CTZ_IEN):** When set, the IPEND goes active whenever a Count-to-Zero event occurs.
30	RW	0	**Status Interrupt Enable (STA_IEN):** When set, IPEND will go active whenever DMA_STAN.EN transitions from 1 to 0.
29	*RO*	*0*	*Reserved*
28:24	RW	00000	**Burst Size (BURST):** The number of bytes to be transferred into and out of the DMA FIFO in handling a single burst. • 00000: 1 byte • 00001: 2 bytes • ... • 11111: 32 bytes
23	*RO*	*0*	*Reserved*
22	RW	0	**Destination Increment Enable (DINCR):** When set, enables incrementing of DMA_DESTN on every AHB transaction. Forced to 0 for DMA transmit to peripherals.
21:20	RW	00	**Destination Width (DWIDTH):** Indicates the width of the each AHB transaction to the destination peripheral or memory. • 00: byte • 01: half-word • 10: word • 11: reserved
19	*RO*	*0*	*Reserved*
18	RW	0	**Source Increment Enable (SINCR):** When set, enables incrementing of DMA_SRCN upon every AHB transaction. Forced to 0 for DMA receive from peripherals.

Bits	Type	Reset	Description
17:16	RW	00	**Source Width (SWIDTH):** Indicates the width of the source device. In most cases, this will be the data width of each AHB transactions. • 00: byte • 01: half-word • 10: word • 11: reserved
15:14	RW	00	**Pre-Scale Select (PS):** Selects the Pre-Scale divider to use for the channel timer. • 00: Disable timer for this channel. • 01: Pre-Scale is hclk divided by 256 • 10: Pre-Scale is hclk divided by 64k • 11: Pre-Scale is hclk divided by 16M
13:11	RW	000	**Time-Out Select (TO):** Selects the number of pre-scale clocks seen by the channel timer before a time-out conditions is generated for this channel. Since the pre-scalar runs independent of the individual channel timers, the actual number of Pre-Scale clock edges seen has a margin of error equal to a single Pre-Scale clock. • 000: 3-4 Pre-Scale clocks • 001: 7-8 Pre-Scale clocks • 010: 15-16 Pre-Scale clocks • 011: 31-32 Pre-Scale clocks • 100: 63-64 Pre-Scale clocks • 101: 127-128 Pre-Scale clocks • 110: 255-256 Pre-Scale clocks • 111: 511-512 Pre-Scale clocks
10	RW	0	**Request Wait Enable (WAIT):** When cleared, the channel timer is enabled as soon as the channel is enabled and PS is non-zero. Setting this bit delays the channel timer enabled until at least one request has been received from the peripheral.
09	*RO*	*0*	*Reserved*
08:04	RW	00000	**Request Select (REQ):** Used to select which DMA request line is used for the channel.
03:02	RW	00	**Priority (PRI):** Channel DMA priority. 00 = Highest Priority. 11 = Lowest priority
01	*RO*	*0*	*Reserved*
00	RW	0	**Enable (EN):** When set, the channel is enabled. After clearing to '0', DMA_STAN.EN determines when the channel is disabled. This bit is automatically cleared whenever DMA_STAN.EN transitions from 1 to 0.

REQ	Definition	REQ	Definition
00h	memory-to-memory	10h	Reserved
01h	Rx UART0	11h	Rx UART0
02h	Rx UART1	12h	Rx UART1
03h	Rx UART2	13h	Rx UART2
04h	Rx SPI 0	14h	Rx SPI 0
05h	Rx Smart Card	15h	Rx Smart Card
06h	N/A	16h	N/A
07h	Rx External Peripheral	17h	Rx External Peripheral
08h	Rx ADC	18h	Tx SHA-1
09h	Rx MCR	19h	LCD Controller
0Ah	Rx SPI 1	1Ah	Tx SPI 1
0Bh – 0Fh	Reserved	1Bh – 1Fh	Reserved

8.12.2.2 DMA_STAn – DMA Channel "n" Status Register



Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
06	RW1C	0	**Time-Out (TO):** When set, a time-out has occurred for this channel.
05	*RO*	*0*	*Reserved*
04	RW1C	0	**Bus Error (BUS_ERR):** When set, an AHB abort was received and the channel disabled.
03	RW1C	0	**Reload Status (RLOAD):** When set, indicates a reload has occurred on this channel.
02	RW1C	0	**Count-to-Zero Status (CTZ):** When set, a Count-to-Zero condition has occurred.
01	RO	0	**Pending (PEND):** When set, indicates that a DMA request is pending for this channel.
00	RO	0	**Enable Status (EN):** When set, the channel is enabled. When cleared, the configuration, address, and count registers for the channel may be altered. This bit follows DMA_CFGN.EN. This bit automatically clears under the following conditions: • Bus error (cleared immediately) • Count-to-zero with RLOAD EN=0 (cleared at the end of the AHB R/W burst). • EN CTRL cleared by programmer (cleared at the end of the AHB R/W burst). When this bit transitions from 1 to 0, DMA_CFGN.EN also clears (if not cleared already).

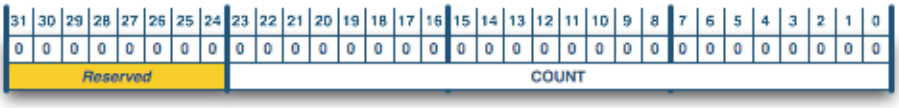
8.12.2.3 DMA_SRCn – DMA Channel "n" Source Register

Bits	Type	Reset	Description
31	*RO*	*0*	*Reserved*
30:00	RW	0	**Address (ADDR):** For peripheral transfers, address bits are fixed (see Table 8-1). For memory transfers, if DMA_CFGN.SRC_INC is '1', the counter is incremented by 1, 2, or 4, depending on the width of each AHB cycle. When a count-to-zero occurs and DMA_CRLDN.EN = '1', this is reloaded with the contents of DMA_SRLDN.

8.12.2.4 DMA_DESTn – DMA Channel "n" Destination Register

Bits	Type	Reset	Description
31	*RO*	*0*	*Reserved*
30:00	RW	0	**Address (ADDR):** For peripheral transfers, address bits are fixed (see Table 8-1). For memory transfers, if DMA_CFGN.DEST_INC is '1', the counter is incremented by 1, 2, or 4, depending on the width of each AHB cycle. When a count-to-zero occurs and DMA_CRLD.EN = '1', this is reloaded with the contents of DMA_DRLD.

8.12.2.5 DMA_CNTn – DMA Channel N Count Register



Bits	Type	Reset	Description
31:24	*RO*	*0*	*Reserved*
23:00	RW	0	**DMA Count (COUNT):** Number of bytes to transfer. It decrements by 1, 2, or 4 depending on the width of each AHB cycle. When the counter reaches 0, a count-to-zero condition occurs. • 0000h: 0 bytes • 0001h: 1 byte • 0002h: 2 bytes • ... • FFFFh: 64K bytes

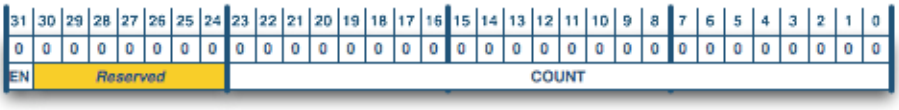
8.12.2.6 DMA_SRLDn – DMA Channel N Source Reload Register

Bits	Type	Reset	Description
31	*RO*	*0*	*Reserved*
30:00	RW	0	**Address (ADDR):** Reload value for DMA_SRCN.

8.12.2.7 DMA_DRLDn – DMA Channel N Destination Reload Register

Bits	Type	Reset	Description
31	*RO*	*0*	*Reserved*
30:00	RW	0	**Address (ADDR):** Reload value for DMA_SRCN.

8.12.2.8 DMA_CRLDn – DMA Channel N Count Reload Register



Bits	Type	Reset	Description
31	RO	0	**Reload Enable (EN):** When set, enable DMA_SRCN, DMA_DESTN and DMA_CNTN to be reloaded with their corresponding reload registers upon count-to-zero. This bit must be set after the address reload registers have been programmed.
30:24	*RO*	*0*	*Reserved*
23:00	RW	0	**Counter Reload Value (COUNT):** Reload value for DMA_CNTN.

Chapter 9: Magnetic Card Reader (MCR)

The Magnetics Card Reader Module is depicted below. Features:

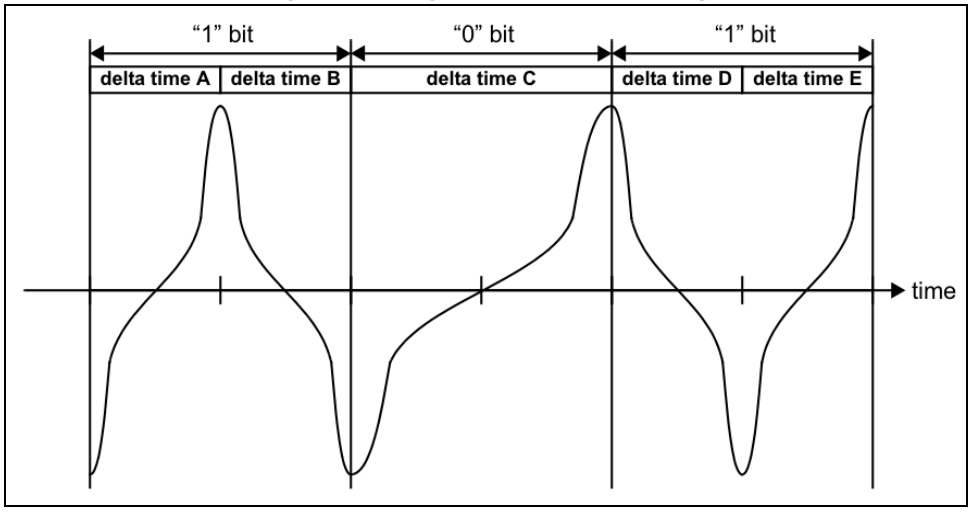
- Support for 3 simultaneous tracks
- Direct mode operation supports direct connection to magnetic heads
- Bypass mode operation supports digital inputs from magnetic card readers
- Automatic peak detection and delta-time generation
- Accurate peak detection between 20mV and 400mV peak-to-peak
- Up to 266kHz sampling resolution per channel to support bit rates of 150bps to 12,000bps
- A single 8-deep, 32-bit FIFO
- Raw ADC sample access
- DMA support
- Interrupt support

9.1 Magnetic Card Reading Overview

Figure 9-1 shows an example of magnetic card bit encoding, derived from a single magnetic card track. The signal contains a series of alternating peaks occurring once or twice within a single bit time. Logic '0' is encoded as one transition per bit; logic '1' is encoded as two transitions. Although the bit rate will vary and distortions will occur for various reasons, the bit rate is relatively constant from bit-to-bit and the delta time between alternating peaks can be used to decode the series of '1's and '0's.

The Z32AN Series MCR can be configured to generate a stream of delta-time integers which can be processed by the user to decode the magnetic card stripes. These delta time values appear in a FIFO along with the associated track number which can be processed by the user's bit decoder application. A '1' can be identified by the fact that it consists of two delta-time integers roughly equal to the single delta-time integer of a '0'.

Figure 9-1: Magnetic Card Bit encoding



9.2 Direct Mode Operation of MCR

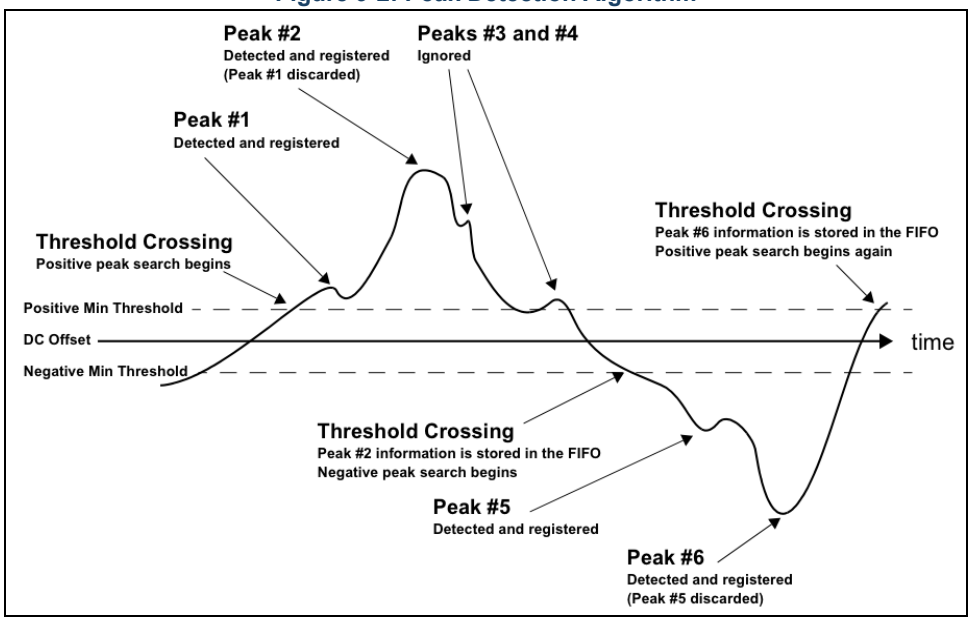
9.2.1 Peak Detection Algorithm

The peak detection algorithm is rather simple and consists of this:

1. The user programs the positive and negative minimum threshold registers.
2. Once the minimum positive threshold is crossed, the Peak Detector begins searching for a positive peak. This first value over the minimum is registered.
3. Every incoming sample is compared to the registered value.
4. If the incoming sample is greater, it is registered and the old value is discarded. Otherwise, the incoming sample is ignored.
5. If the incoming sample falls below the negative threshold, the Peak Detector stores the positive peak information in the FIFO and begins the search for the negative peak.
6. The negative peak search is analogous to the positive peak search.
7. The Peak Detector alternates between the negative and positive peak searches until a time-out occurs.

Figure 9-2 depicts the peak detection algorithm.

Figure 9-2: Peak Detection Algorithm



9.2.2 Stored Peak Information

Peak information stored in the FIFO is a 32-bit value with the following information:

- Delta-time from previous peak
- Track number
- Polarity of current peak
- Whether a time-out caused this entry into the FIFO
- Amplitude of current peak

The format of this can be found in MCR_FIFO. To save memory, using only 16-bits, discard the amplitude information held in the upper bits (for the case of manual read from the FIFO) or program the DMA to use only 16-bits (for the case of DMA operation).

9.2.3 Peak Detection Timer and Time-out

Each track has an internal 12-bit timer which is used to calculate the delta-time between peaks that increments with every sample of the ADC. In addition to calculating the delta-time, this timer is used to detect when peaks are no longer being received by the MCR (indicating the end of a card swipe).

All three timers will only count up to the maximum value specified in the MCR_TMR. Should any timer count up to that value, a time-out condition will occur for that track. Under this condition, the last valid delta-time entry will be stored in the FIFO and the timer will be disabled (preventing further time-out interrupts). Once another threshold crossing occurs, the timer will be automatically re-enabled and the search for time-out conditions begins again.

9.2.4 Card Time-out and Track Timers

As described above, each track has an internal 12-bit timer. An interrupt can be generated on the time-out of each track, or a single time-out can be generated on the time-out of the “last” track. This simplifies MCR interrupt handling. A time-out is considered to be the “last” if the other two timers are either: 1) not enabled; 2) already timed-out; or 3) have not encountered an initial peak (never started).

The card time-out is enabled by setting MCR_CTRL.CART_TO_IEN.

9.2.5 Dynamic Minimum Thresholds

There are two modes of operation for threshold values, selected by MCR_CTRL.THRESH. The first mode is static threshold mode. In this mode, the positive minimum threshold and the negative minimum threshold programmed by the user are always the thresholds used by the peak detector.

The second mode is dynamic threshold mode. In this mode, the previous peak is scaled by 1/4, 1/8, or 1/16 to generate a temporary internal threshold for the next peak (the values in MCRn_THRS do not change). If the scaled value is ever computed to be less than the value of MCRn_THRS, then that minimum value is used instead of the scaled version (i.e. the thresholds can never be less than the programmed values). In the event of a time-out, both thresholds revert to the programmed value.

9.2.6 MCR Interrupts

Below are the registers and fields which are used to control and handle DMA interrupts.

- MCR_CTRL.CARD_TO_IEN → MCR_INT.CARD_TO
- MCR_CTRL.AUX_ADC_IEN → MCR_INT.AUX_ADC
- MCR_CTRL.UFLO_IEN → MCR_INT.UFLO
- MCR_CTRL.OFLO_IEN → MCR_INT.OFLO
- MCR_CTRL.LVL_IEN → MCR_INT.LVL
- MCR_CTRL.TOn_IEN → MCR_INT.TOn
- MCR_INT.STA

Status bits are set whenever the corresponding event occurs whether or not the corresponding enable bit is set. These bits remain set (latched) until cleared by a write to MCR_INT. Whenever a status bit and corresponding enable bit are active, an interrupt is generated.

9.2.7 Acquiring Raw ADC Samples

Any of the three tracks can be used to generate raw ADC data. There are two means of accessing raw ADC samples and these are outlined below.

9.2.7.1 Using the Auxiliary ADC Register

This method has the following advantages and restrictions:

- Only one track can be sampled at a time.
- The track can be sampled while peak detection mode is active.
- There is no FIFO to hold data and DMA is not supported for this register, so response time is more restricted.

Following are the registers and fields that must be configured to enable use of the MCR_AUX_ADC. Once configured, every new sample from the configured track is moved into MCR_AUX_ADC (provided that the previous sample has been read).

- MCR_ADC → All fields
- MCR_CTRL → AUX_ADC_IEN, IENn
- MCR_AUX_ADC → NEW, SAMPLE, OFLO

9.2.7.2 Using ADC Mode

This method has the following advantages and restrictions:

- Any number of tracks can be sampled at a time.
- Peak-Detection must be disabled while a track is in ADC mode.
- There is FIFO and DMA support.

Below are the registers and fields that are used to configure a track for ADC mode. Once configured, ADC samples are streamed into the FIFO along with data from any other enabled channels.

- MCR_ADC → All fields
- MCR_CTRL → MODEn, IENn
- MCR_FIFO → TRACK, TIME

9.2.8 Programming Guide

The MCR provides a stream of either delta time or ADC samples. The optimum parameters are strongly dependent upon system design. Some general guidelines are provided below.

9.2.8.1 Sample Rate Programming

The ADC sample rate is configured by programming MCR_ADC. The sample rate for each track is dependent upon the number of active channels. The equation to determine the ADC divider is:

$$MCR\_ADC.DIV = \frac{hclk\ Frequency}{(Track\ Sample\ Rate \times 13 \times Number\ of\ MCR\ tracks)}$$

For example, if hclk is 90MHz, the number of active MCR channels is 3, and the desired track sample rate is 250kHz, then:

$$MCR\_ADC.DIV = \frac{90\ MHz}{(250\ kHz \times 13 \times 3)} = 8$$

Thus, programming 7 into MCR_ADC.DIV produces a sample rate of 250kHz for each track.

9.2.8.2 DC Offset Programming

The DC offset needs to be programmed for each track by writing to the MCRn_DCO. To determine value to use, the programmer should acquire a number of raw ADC samples and choose the average value. The DC offset should be close to the middle range of 800h. The variation between samples should be less than 00Fh

when no card swipe is active. MCR_AUX_ADC allows the programmer to sample the ADC while the track is still active. See section 9.2.7 for more details.

9.2.8.3 Programmable Gain Amplifier

The reference voltage can be adjusted through the programmable gain amplifier for the voltage reference of the ADC. Increasing the peak to peak input range is achieved by changing MCR_ADC.REFCAL to adjust the ratio of input gain from 1 to 10. See section 9.2.7 for more details.

9.2.8.4 Threshold Programming

The threshold settings should be such that the noise of the MCR does not exceed the threshold limits.

9.2.8.5 Max Delta Time Programming

The maximum delta time can be safely set at the maximum value (FFFh), though the lowest bit rate (150bps) with the highest sample rate (266.7kHz) should imply that 7F2h should correspond to a “bit too wide” condition.

9.2.8.6 Enabling Channels and Handling FIFO Data

Once the above parameters have been programmed the MCR channels can be enabled via MCR_CTRL.IENn. Once this is done, delta time information is moved into the FIFO as a card is swiped and the thresholds are crossed. A time-out interrupt will arrive when the swipe is completed. A channel may be disabled at any time by clearing its MCR_CTRL.IENn. Enabling tracks affects the sample rate. See section 9.2.8.1 for more details.

The user can rely either upon FIFO and time-out interrupts or DMA to move data out of MCR_FIFO into a memory buffer. If multiple tracks are enabled, the buffer will contain data from one track interleaved with data from the other tracks, so it will be necessary to sort the data according to the track number stored with the delta time information. When a time-out occurs, the peak detector for that channel continues to search for another swipe and begins to move data into the FIFO as soon as more peaks begin to be detected.

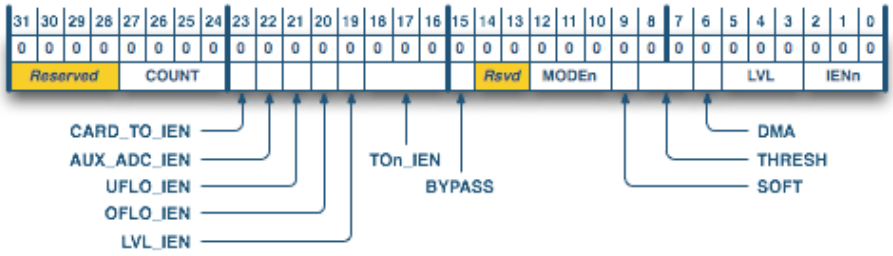
9.2.9 Card Time-out and Track Timers

Each track has an internal 12-bit timer. An interrupt can be generated on the time-out of each track, or a single time-out can be generated on the time-out of the “last” track. This simplifies MCR interrupt handling. A time-out is considered to be the “last” if the other two timers are either: 1) not enabled; 2) already timed-out; or 3) have not encountered an initial peak (never started). The card time-out is enabled by setting MCR_CTRL.CARD_TO_IEN.

9.3 Registers (Base → FFFF3000h)

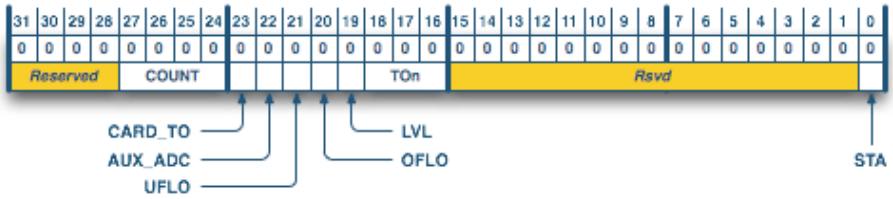
Offset	Register	Description
000h	MCR_CTRL	MCR Global Control Register
004h	MCR_INT	MCR Interrupt Control and Status
008h	MCR_TMR	MCR Timing Register
00Ch	MCR_FIFO	MCR FIFO Register
010h	MCR_ADC	MCR ADC Register
014h – 01Ch	MCRn_DCO	MCR”N” DC Offset Register (N = 0, 1, or 2)
020h – 028h	MCRn_THRS	MCR”N” DC Threshold Register (N = 0, 1, or 2)
02Ch	MCR_AUX_ADC	MCR Auxiliary ADC Register

9.3.1 Offset 000h: MCR_CTRL – MCR Control Register



Bits	Type	Reset	Description
31:24	*RO*	*0*	*Reserved*
27:24	RO	0h	**FIFO Count (COUNT):** Number of valid entries are contained in the MCR FIFO. • 0000: None, 0001: 1 entry, 0010: 2 entries, ..., 1000: 8 entries • 1001 - 1111: Invalid
23	RW	0	**Card Time-out (CARD_TO_IEN):** When set, enables card time-outs to cause an interrupt.
22	RW	0	**Aux ADC (AUX_ADC_IEN):** When set, enables interrupts when a new sample is available in MCR_AUX_ADC.
21	RW	0	**FIFO Underflow (UFLO_IEN):** When set, enables interrupts on a FIFO underflow.
20	RW	0	**FIFO Overflow (OFLO_IEN):** When set, enables interrupts on a FIFO overflow.
19	RW	0	**FIFO Level (LVL_IEN):** When set, enables interrupts on the number of FIFO entries greater than or equal to the FIFO level.
18:16	RW	0	**Track "N" Timeout (TOn_IEN):** When set, enables time-out interrupts from the specified MCR track.
15	WO	0	**Bypass ADC (BYPASS):** When set, enables digital inputs to bypassing the ADC.
14:13	*RO*	*0*	*Reserved*
12:10	RW	0	**Track "N" Mode (MODEn):** When set, the track is in ADC mode. When cleared, the track is in peak-detection mode.
09	RW	0	**Soft Reset (SOFT):** When written to '1', resets the MCR state machines and FIFOs. Does not affect analog or ADC registers. Writes of '0' have no effect.
08:07	RW	00	**Threshold Mode (THRESH):** Selects the threshold mode to use to all tracks: • 00: Static thresholds • 01: Enable 1/4 scaling thresholds • 10: Enable 1/8 scaling thresholds • 11: Enable 1/16 scaling thresholds
06	RW	0	**DMA Enable (DMA):** When set, enables DMA requests when FIFO level is reached or exceeded.
05:03	RW	000	**FIFO Level (LVL):** Sets the number of MCR FIFO entries required for a DMA request or FIFO interrupt. • 000: 1 FIFO entry, 001: 2 FIFIO entries, ..., 111: 8 FIFO entries
02:00	RW	0	**Track "N" Enable (IENn):** Enables each track. When cleared, the track is disabled. When set, the track is enabled.

9.3.2 Offset 004h: MCR_INT – MCR Interrupt Register

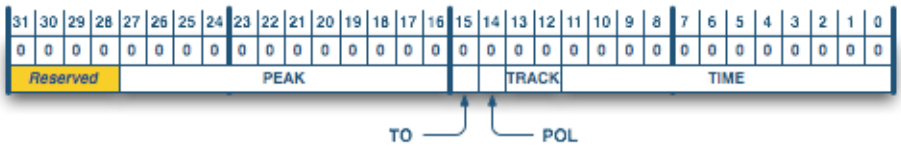


Bits	Type	Reset	Description
31:24	*RO*	*0*	*Reserved*
23	RW1C	0	**Card Time-out (CARD_TO):** When set, indicates occurrence of a card time-out. This occurs when one of the tracks times out and both other tracks are either already timed-out or were never started.
22	RW1C	0	**AUX ADC (AUX_ADC):** When set, indicates availability of a new AUX ADC sample.
21	RW1C	0	**FIFO Underflow (UFLO):** When set, indicates occurrence of a FIFO underflow
20	RW1C	0	**FIFO Overflow (OFLO):** When set, indicates occurrence of a FIFO overflow
19	RW1C	0	**FIFO Level (LVL):** When set, indicates that the number of entries in the FIFO is greater than or equal to the level specified in MCR_CTRL
18:16	RW1C	000	**Track "N" Timeout (TOn):** Indicates that the specified MCR track has encountered a peak search timeout. Bit 18 = MCR2, bit 17 = MCR1, and bit 16 = MCR0.
15:01	*RO*	*0*	*Reserved*
00	RO	0	**Interrupt Status (STA):** When set, indicates MCR is driving an interrupt

9.3.3 Offset 008h: MCR_TMR – MCR Timing Register

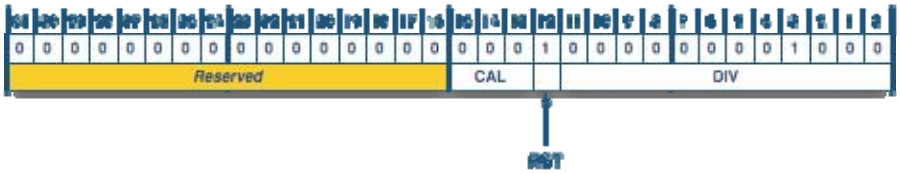
Bits	Type	Reset	Description
31:12	*RO*	*0*	*Reserved*
11:00	RW	FFFh	**Maximum Delta-Time (MAX_DELTA_TIME):** Sets the number of samples before a time-out

9.3.4 Offset 00Ch: MCR_FIFO – MCR FIFO Register



Bits	Type	Reset	Description
31:28	*RO*	*0*	*Reserved*
27:16	RO	000h	**Peak Amplitude (PEAK):** In peak-detection mode, this provides the amplitude of the peak. In Raw ADC mode, these bits are always 0.
15	RO	0	**Timeout (TO):** In peak detection mode, indicates timeout ('0' = no timeout, '1' = timeout). In ADC mode, this bit is always '0'.
14	RO	0	**Peak Polarity (POL):** In peak-detection mode, this provides the polarity of the peak ('0' = negative peak, '1' = positive peak). In Raw ADC mode, this bit is set to zero.
13:12	RO	00	**Track Number (TRACK):** Provides the track number associated with the FIFO information. • 00: Track 0 • 01: Track 1 • 10: Track 2 • 11: N/A
11:00	RO	000h	**Delta-Time/ADC Sample (TIME):** Contains the delta-time value or raw ADC sample.

9.3.5 Offset 010h: MCR_ADC – MCR ADC Register



Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:13	RW	000	**ADC Reference Calibration (CAL):** Selects the Peak to Peak differential input voltage range for full-scale output. • 000 = 0.50 • 001 = 0.40 • 010 = 0.30 • 011 = 0.25 • 100 = 0.20 • 101 = 0.15 • 110 = 0.10 • 111 = 0.05
12	RW	1	**MCR Reset (RST):** When set, resets the MCR ADC.
11:00	RW	004h	**ADC Clock Divider (DIV):** Determines the divider for the ADC clock: See section 9.2.8.1 for details on how to create this value. Acceptable values: • 000h: Illegal • 001h: hclk divided by 2 • 002h: Illegal • 003h: Illegal • 004h: hclk divided by 5 • ... • FFFh: hclk divided by 256

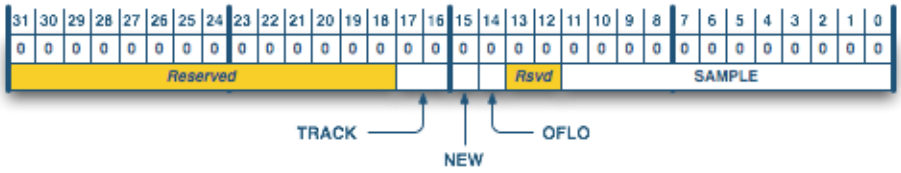
9.3.6 MCRn_DCO – MCR DC Offset Registers (MCR0: 014h, MCR1: 018h, MCR2: 01Ch)

Bits	Type	Reset	DescriptionDS0200.docx
31:12	*RO*	*0*	*Reserved*
11:00	RW	000h	**DC Offset (OFFSET):** Used to set the center point of the ADC. This should be set to the average raw ADC sample value for track N with no card swipe.

9.3.7 MCRn_THRS – MCR Threshold Registers (MCR0: 020h, MCR1: 024h, MCR2: 028h)

Bits	Type	Reset	Description
31:28	*RO*	*0*	*Reserved*
27:16	RW	000h	**Negative Minimum Threshold (NMT):**
15:12	*RO*	*0*	*Reserved*
11:00	RW	000h	**Positive Minimum Threshold (PMT):**

9.3.8 Offset 02Ch: MCR_AUX_ADC – MCR Auxiliary ADC Register



Bits	Type	Reset	Description
31:18	*RO*	*0*	*Reserved*
17:16	RW	00	**Track Number (TRACK):** Configure which track is used to extract the samples. • 00: Track 0 • 01: Track 1 • 10: Track 2 • 11: N/A
15	RO	0	**New Sample (NEW):** When set, indicates a sample has been loaded into ADC_SAMPLE. While this bit is set, ADC_SAMPLE will not be loaded with a new value. Cleared by a read of this register.
14	RO	0	**Overflow (OFLO):** When set, indicates one or more samples have been discarded due to the fact that NEW was set and ADC_SAMPLE could not be reloaded. Cleared by a read of this register.
14:12	*RO*	*0*	*Reserved*
11:00	RO	000h	**ADC Sample (SAMPLE):** Latest ADC sample from track specified in "Track Number".

Chapter 10: Smart Card Controller

The Smart Card Controller is an APB device that allows a seamless connection to external Smart Card Interface devices. Reference to the *ON Semiconductor NCN6001 Smart Card Interface IC* is made throughout this chapter which serves as an example interface for the Smart Card Controllers.

The protocol layer is not implemented in hardware and must be managed by the software. The controller includes the following features:

- Supports two interfaces: One for maincard and one for SIM card.
- Interrupt.
- DMA support.
- Master only SPI Interface
- Programmable Baud Rate Generator.
- Timing Checker.

10.1 SPI Interface

The SPI port is able to serially send and receive 8-bit data to the external Smart Card Interface IC (MSB sent and received first). It is a master only interface. The SC_nSS0 and SC_nSS1 determine which external interface is being accessed.

The clock is active high and idles low. The frequency is programmable through SC_SPI_CLK and allows discrete division ratios from the hclk frequency: all even numbers between 2 and 32 (inclusive). SPI clock frequency is calculated as per the below equation:

$$SC\_SCK = \frac{hclk}{[2 \times (SC\_SPI\_CLK.DIV + 1)]}$$

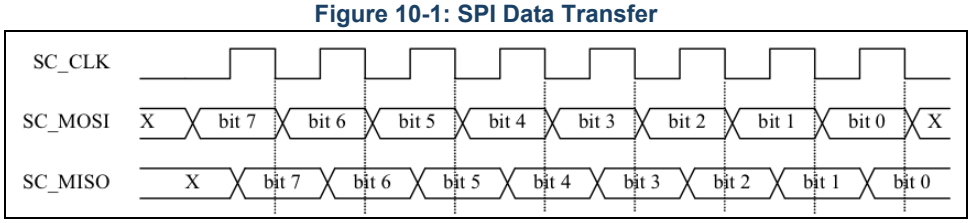
The SPI data interface comprises of two unidirectional ports: SC_MOSI and SC_MISO. SC_MOSI is an output serial data line; SC_MISO is a concurrent input serial data line. Data is sent on SC_MOSI and received from SC_MISO and are clocked by the same clock SC_SCK. Each time a word is sent, another byte is received.

When the CPU sends a byte through the SPI by writing the data in the SPIDATA register, it can get the concurrently received byte by reading SPIDATA. The received byte is valid once bit 8 (MSB) of the SPIDATA is set to 1.

When an automatic RST or VCCON command is sent through the SPI (after a change of one of the 2 Card Reset or Card VCC lines), a byte is received, which is thrown away. This byte cannot be read by the processor as SPIDATA is not updated with this data.

Figure 10-1 shows the waveforms of the SPI port. The SC_MISO input is sampled on the falling edge of SC_SCK, for example, OnSemi Smart Card interface chip (OnSemi NCN6001), must be programmed to operate in Special Mode. Refer to the datasheet for the OnSemi NCN6001 device or the specific device used in the application to determine SPI interface requirements.

Figure 10-1: SPI Data Transfer



10.1.1 Smart Card Controller Interrupt Management

An Interrupt indicates when an over current, overheating, card removal, or card insertion is detected on one of the Smart Card interfaces through SC_nALARM. Status is obtained by polling the Smart Card device.

10.1.2 Reset and Power-up Management

When reset or power-up commands are issued using the COMMAND register, the controller transfers the corresponding command words to the selected Smart Card through the SPI bus. The command request is queued and sent through the SPI bus as soon as the current SPI transfer is complete. The voltage information and clock division ratios must be programmed before the reset or power-up command is issued. The SPI command format is (most significant bit first) is given below:

Where,

- S[2:0]: target Smart Card interface address (INT1=000, INT2=001 INT3=010, INT4=011, EXT ASYNC=100).
- nRST: is the active-Low reset signal to be applied to the Smart Card.
- CK[1:0]: is the content of the CLKDIV register for the Smart Card.
- VCC[1:0]: is the content of the VCC register for the active Smart Card when the card must be powered on or 00 when the active card must be powered off.

10.1.3 Chip to Smart Card Interface Mapping

Smart Card 0 is mapped to the Maincard. Smart Card 1 communicates with a SIM via the interface device. For the non-active Smart Card, the clock and I/O lines are held inactive. Which SIM to use is selected by SIM_SEL. At reset, Smart Card 1 is mapped to SIM0 (deactivated).

10.1.4 DMA Interface

Each DMA channel can be independently assigned to either of the Smart Card RX or TX channels or deactivated. For example, the TX channel can be used with Smart Card 0 while the RX channel is used with Smart Card 1. At reset, the RX and TX DMA channels are not mapped and must be configured prior to use.

> ►***Note:*** *Ensure that Smart Card 0 and Smart Card 1 are not mapped onto the same DMA channel as it can lead to unpredictable operation.*

10.1.5 Synchronous Smart Card Handling

The SPI interface allows management of synchronous Smart Cards by sending corresponding commands to the interface devices to manually activate the clock and data signals.

10.1.6 Interrupt Generation

Following are the interrupt sources:

- **SPI Interrupt:** Occurs whenever an SPI collision occurs. For example, the CPU writes new data into the SPIDATA register while the previous transfer is not completed. Bit 8 of the SPIDATA register remains at 0 when the transfer is in process.
- **ALARM Interrupt:** Occurs whenever one of the Smart Card interfaces generates an interrupt indicating over current, overheating, card removal or card insertion is on one Smart Card socket. The interrupt lines from each Smart Card interface are combined to form the SC_nALARM.
- **ALARM TRIG Interrupt:** Generated whenever a High to Low transition is detected on the SC_nALARM input.

These interrupt sources are combined to generate the Smart Card Alarm interrupt and can be individually masked in SC_IMASK. The Smart Card interfaces also provide interrupts. These interrupts cannot be masked in SC_IMASK. However, their status is available in SC_ISTAT.

10.2 Blocks

The Smart Card function contains two blocks: a UART, and a Controller and Timing Checker:

10.2.1 UART

The UART is connected to APB and supports two DMA requests, one for receive and another for transmit. For receive, a DMA request indicates when a word is available in the receive data register. For transmit, a DMA request indicates when a word can be accepted by the UART for transmission.

The receive channel supports a word register allowing the UART to function without DMA. The CPU can be informed it can read a word from the UART in the by an interrupt. Words to transmit can be written through single DMA transfers or through CPU direct accesses.

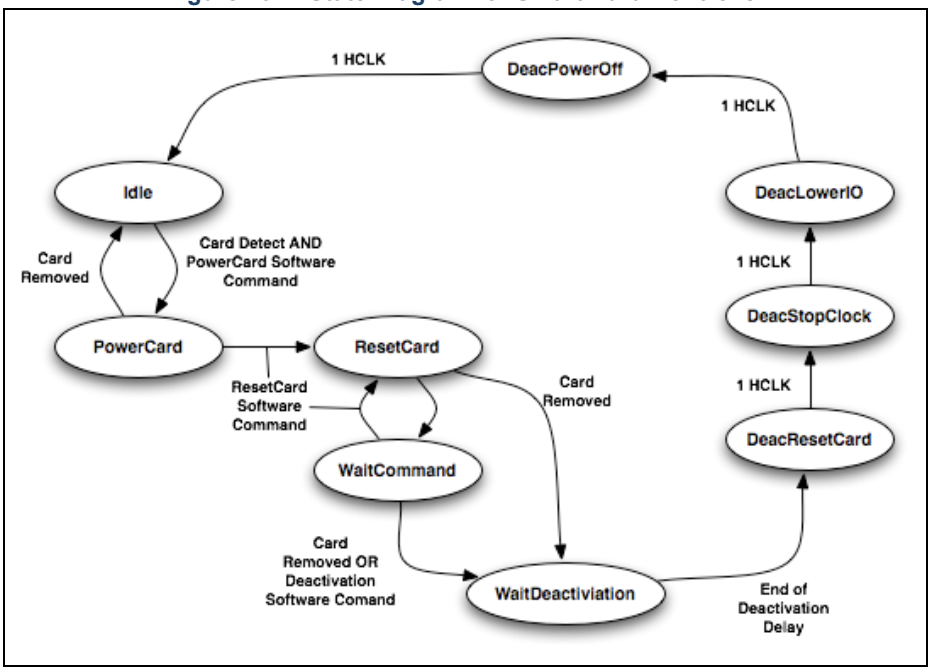
10.2.2 Programmable Baud Rate Generator (BRG)

The BRG produces the Elementary Time Unit (ETU) timing reference which is the serial bit duration used for data transfer on SC_IO. The ETU is obtained by dividing SC_CLK, with a programmable factor defined in BRR.

10.2.3 Controller

There is one controller state machine in Smart Card 0 and four controller state machines in Smart Card 1 to manage card power on, card cold and warm reset, and card deactivation stages. The Smart Card 1 multiplexer must be configured before each command to select the appropriate controller state machine via SIM_SEL. The state diagram for the controller is shown below.

Figure 10-2: State Diagram for Smart Card Controller



The states are described below:

- **Idle**: After card detection (DET_CMD), software sets COMMAND.PWR_CARD to '1' to power up the card before changing to *PowerCard*. All outputs are in their inactive state.
- **PowerCard**: The controller applies power. Software sets COMMAND.DEAC_CARD to '1'. It is up to software to ensure power is stable before sending the CardReset command by reading SCSTATUS. If the card is removed, the controller automatically returns to the Idle state.
- **ResetCard**: On entering this state, the card clock is started, the card IO is '0', the card reset is asserted by the external interface device, and the reset duration counter is started. When the counter reaches the count programmed in RST_LEN, the controller changes to *WaitCommand*. If the card is removed anytime during the reset state the controller changes to *WaitDeactivation*.
- **WaitCommand**: On entering this state, card reset is set inactive and data transfers may occur. When data transfer is complete, one of the following three commands can be issued.
  - **StopCardClock**: The Smart Card clock is stopped.
  - **DeactivateCard**: The controller changes to *WaitDeactivation*.
  - **ResetCard**: The controller performs a warm reset of the card by changing to *ResetCard*.
- If the card is removed during this state the controller automatically changes to the WaitDeactivation state.
- **WaitDeactivation**: On entering this state, the deactivation delay counter is started. When the counter reaches the count programmed in DEAC_DLY, the controller moves to *DeacResetCard*. It is up to the software to program the proper delay according to the protocol or the current card session stage.
- **DeacResetCard**: The external interface card reset is pulled '0' and the controller moves to *DeacStopClock*.

- **DeacStopClock**: The card clock is stopped and the controller moves to *DeacLowerIO*.
- **DeacLowerIO**: The IO line is pulled '0' and the controller moves to *DeacPowerOff*.
- **DeacPowerOff**: The card is powered off and the controller moves to *Idle*.

10.2.4 Timing Checker

This performs the mandatory checking on the Answer to Reset (ATR) delay, ATR length, Block Wait Timer (BWT), Character Wait Time (CWT), and Work Wait Time (WWT) parameters.

10.2.4.1 ATR

The ATR is checked against 3 programmable parameters:

- **AtrMinDelay**: Minimum delay between card reset end and ATR start in card clock units.
- **AtrMaxDelay**: Maximum delay between card reset end and ATR start in card clock units.
- **AtrMaxLength**: Maximum ATR length in ETU units. It is up to the software to notify the Smart Card block of the ATR end with an AtrEnd command.

An ATR delay error is raised as a BWT error if a start bit is received before AtrMinDelay after the end of the card reset or if not start bit is received before AtrMaxDelay after the end of the card reset. An ATR length error is raised as a CWT error if the ATR end is not signaled within AtrMaxLength after the start of the ATR. The error is cleared during a card reset or when software reads INT_STAT.

10.2.4.2 Block Wait Time Error (INT_STAT.BWT)

This is raised when the time between the last transmitted start bit and the next received start bit exceeds BWT.BWT_TO. COMMAND.ATR_TO_EN must be set to '1' in order to be checked. The error is cleared during a card reset or when the software reads INT_STAT.

10.2.4.3 Character Wait Time Error (INT_STAT.CWT_ERR)

This is raised when the time between two consecutive received start bits exceeds CWT.CWT_TO. COMMAND.ATR_TO_EN must be set to '1' in order to be checked. The error is cleared during a card reset or when the software reads INT_STAT.

10.2.4.4 Work Wait Time Error (INT_STAT.WWT_ERR)

This is raised when the time between the last received or transmitted start bit and the next received start bit exceeds WWT.WWT_TO. COMMAND.ATR_TO_EN must be set to '1' in order to be checked. The error is cleared during a card reset or when the software reads INT_STAT.

10.2.4.5 Block Guard Time Error (INT_STAT.BGT_ERR)

This is raised when the time between the last transmitted start bit and the next received start bit is less than BGT.BGT_TO. COMMAND.ATR_TO_EN must be set to '1' in order to be checked. The error is cleared during a card reset or when the software reads INT_STAT.

10.2.5 Interrupt Generation

Interrupt generation is governed by IER and INT_EN. When an interrupt is generated, the corresponding bit is set in LSR or INT_STAT. The status bits are cleared when the status register is read except for the TX Shift and TX Hold Empty. TO_EN.MST_INT_EN acts directly on the interrupt line. For ease of use, the status bits contained in the UART LSR are replicated in the controller's INT_STAT. Reading this register also clears LSR status bits.

10.3 Registers

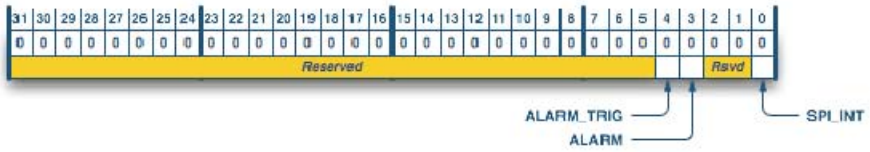
10.3.1 Global Registers (Base → FFFF0000h)

Offset	Register	Description
000h	SC_ISTAT	Interrupt Status
004h	SC_IMASK	Interrupt Mask
008h	SC_VCC_CFG	Smart Cards VCC configuration
00Ch	SC_IF_CLKDIV	Interface Smart Cards Clock divisor configuration
010h	SC_DET_CMD	Smart Card detection command
014h	SC_SPI_DATA	Smart Card SPI Data Register
018h	SC_SPI_CLK	SPI clock division factor
01Ch	SC_STATUS	VCC and Reset status from Smart Card interface
024h	SC_SPIDMASEL	DMA channels to Smart Card and SPI block mapping

10.3.1.1 Offset 000h: SC_ISTAT – Interrupt Status Register

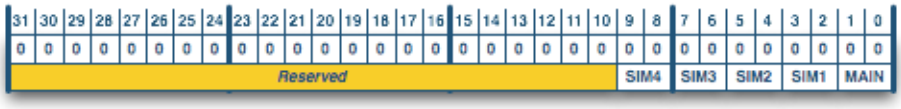
Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RO	0	**Alarm Trigger (ALARM_TRIG):** Set when there is a transition from '1' to '0' on SC_nALARM.
03	RO	0	**Alarm (ALARM):** When set, SC_nALARM is active.
02	RO	0	**Smart Card 1 Interrupt (SC1_INT):** When set, Smart Card 1 interrupt is active. This bit is cleared when the status register of the Smart Card 1 is read.
01	RO	0	**Smart Card 0 Interrupt (SC0_INT):** When set, Smart Card 0 interrupt is active. This bit is cleared when the status register of the Smart Card 1 is read.
00	RO	0	**SPI Interrupt (SPI_INT):** When set, a collision occurred on SPI. This bit is cleared when the register is read.

10.3.1.2 Offset 004h: SC_IMASK – Interrupt Mask



Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RW	0	**Alarm Trigger Mask (ALARM_TRIG):** When set, mask the interrupt when there is a transition from High to Low on the SC_nALARM input.
03	RW	0	**Alarm Mask (ALARM):** Masks SC_nALARM when this bit is set.
02:01	*RO*	*0*	*Reserved*
00	RW	0	**SPI Interrupt Mask (SPI_INT):** When set, mask the interrupt from SPI.

10.3.1.3 Offset 008h: SC_VCC_CFG – Smart Cards VCC Configuration

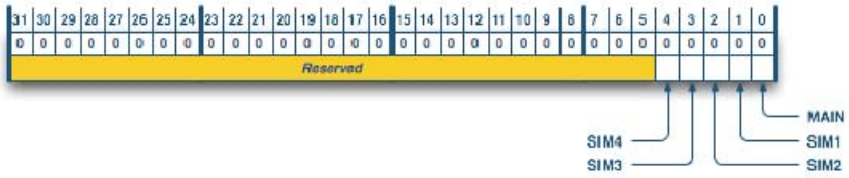


Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09:08	RW	00	**SC4 VCC configuration (SIM4):** Configures VCC voltage • 00 = 0 V • 01 = 1.8 V • 10 = 3 V • 11 = 5 V
07:06	RW	00	**SIM3 VCC configuration (SIM3):** Configures VCC voltage. Same encodings as SIM4.
05:04	RW	00	**SIM2 VCC configuration (SIM2):** Configures VCC voltage. Same encodings as SIM4.
03:02	RW	00	**SIM1 VCC configuration (SIM1):** Configures VCC voltage. Same encodings as SIM4.
01:00	RW	00	**Main VCC configuration (MAIN):** Configures VCC voltage. Same encodings as SIM4.

10.3.1.4 Offset 00Ch: SC_IF_CLKDIV – Interface Smart Cards Clock Divisor Configuration

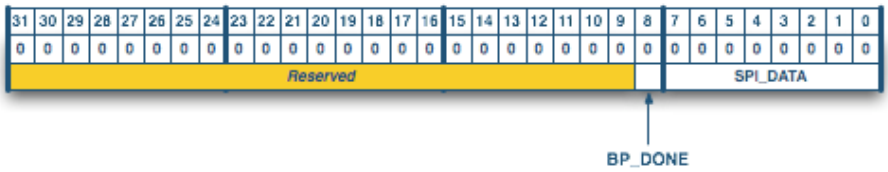
Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09:08	RW	00	**SIM4 CLK Divide Configuration (SIM4):** These 2 bits are sent to the SIM4 card to configure clock division. • 00: No Clock • 01: CLK • 10: CLK/2 • 11: CLK/4
07:06	RW	00	**SIM3 Clock Divide Configuration (SIM3):** Same encoding as SIM4.
05:04	RW	00	**SIM2 Clock Divide Configuration (SIM2):** Same encoding as SIM4.
03:02	RW	00	**SIM1 Clock Divide Configuration (SIM1):** Same encoding as SIM4.
01:00	RW	00	**Main Clock Divide Configuration (MAIN):** Same encoding as SIM4.

10.3.1.5 Offset 010h: SC_DET_CMD – Detect Command



Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RW	0	**SIM4 Card Detect (SIM4):** When set, Smart Card Detected
03	RW	0	**SIM3 Card Detect (SIM3):** When set, Smart Card Detected
02	RW	0	**SIM2 Card Detect (SIM2):** When set, Smart Card Detected
01	RW	0	**SIM1 Card Detect (SIM1):** When set, Smart Card Detected
00	RW	0	**Main Card Detect (MAIN):** When set, Smart Card Detected

10.3.1.6 Offset 014h: SC_SPI_DATA – Smart Card SPI Data

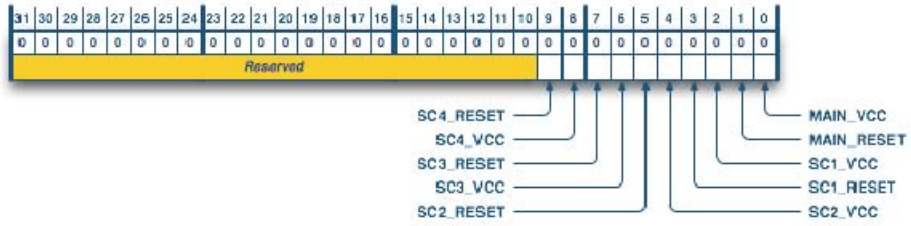


Bits	Type	Reset	Description
31:09	*RO*	*0*	*Reserved*
08	RW	0	**Bypass / Bypass Done (BP_DONE):** When written to '1', the data is sent as it is on SPI. When written to '0', bits 4:0 are replaced with programmed values for Card Reset, Card Clock Divisor and Card VCC (as defined by the OnSemi NCN6001 example interface). Card VCC register value is sent if the internal VCC is selected. When read When read as '1', the previous transfer is complete.
07:00	RW	00h	**SPI data (SPI_DATA):** SPI Data

10.3.1.7 Offset 018h: SC_SPI_CLK – SPI Clock Division Factor Register

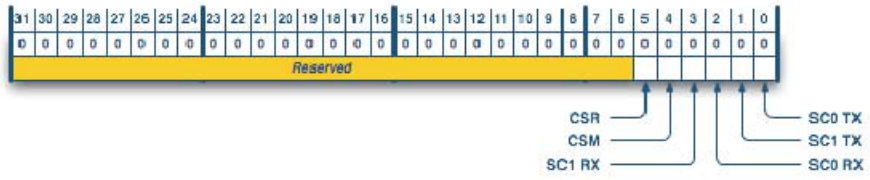
Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	0h	**SPI Clock Divider (DIV):** Clock used is hclk/DIV*2

10.3.1.8 Offset 01Ch: SC_STATUS – Smart Card Interface VCC and Reset Status



Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09	RO	0	**SIM4 Card Reset Status (SIM4_RESET):** When set, reset is valid
08	RO	0	**SIM4 Card VCC Status (SIM4_VCC):** When set, VCC is valid
07	RO	0	**SIM3 Card Reset Status (SIM3_ RESET):** When set, Reset is valid
06	RO	0	**SIM3 Card VCC Status (SIM3_VCC):** When set, VCC is valid
05	RO	0	**SIM2 Card Reset Status (SIM2_RESET):** When set, Reset is valid
04	RO	0	**SIM2 Card VCC Status (SIM2_VCC):** When set, VCC is valid
03	RO	0	**SIM1 Card Reset Status (SIM1_RESET):** When set, Reset is valid
02	RO	0	**SIM1 Card VCC Status (SIM1_VCC):** When set, VCC is valid
01	RO	0	**Main Card Reset Status (MAIN_RESET):** When set, Reset is valid
00	RO	0	**Main Card VCC Status (MAIN_VCC):** When set, VCC is valid

10.3.1.9 Offset 024h: SC_SPIDMASEL – Smart Card DMA Channels and SPI Block Mapping



► ***Warning***: *Ensure that two Smart Card blocks do not map onto the same DMA channel as it can lead to unpredictable operation.*

Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05	RW	0	**CS Reset (CSR):** When set, before each SPI transfer all the Chip Selects will be High for 2 hclk cycles.
04	RW	0	**CS Memory (CSM):** When set, the chip select which are not addressed will keep the same state as prior to SPI. That is, the 2 chip select can be asserted at the same time.
03	RW	0	**SC1 RX (SC1_RX):** When set, selects SC1 RX for DMA
02	RW	0	**SC0 RX (SC0_RX):** When set, selects SC0 RX for DMA
01	RW	0	**SC1 TX (SC1_TX):** When set, selects SC1 TX for DMA
00	RW	0	**SC0 TX (SC0_TX):** When set, selects SC0 TX for DMA

10.3.2 Smart Card UART Mode Registers (Base: SC0 → FFFF0100h, SC1 → FFFF0200h)

Offset	Register	Description
000h	SC_RBR	Receiver Buffer Register
000h	SC_THR	Transmitter Holding Register
004h	SC_IER	Interrupt Enable Register
008h	SC_IIR	Interrupt Identification Register
00Ch	SC_LCR	Line Control Register
010h	SC_BRR	Baud Rate Register
014h	SC_LSR	Line Status Register
018h	SC_GR	Global Register
01Ch	SC_GTIME	Guard time Register
020h	SC_NUM_REP	Number of repetition in Tx Register
024h	SC_REV_DLY	Reverse delay Register
028h	SC_CLK_DIV	Clock Divide Register
02Ch	SC_NUMTX	Number of Data to Send
030h	SC_NUMPE	Number of Repetition in RX for parity error
034h	SC_NUMRX	Number of Data to Receive
038h	SC_AUTO_PARITY	Automatic Parity

10.3.2.1 Offset 00h: SC_RBR – Receive Buffer

► ***Note:*** *RBR and THR share the same address space.*

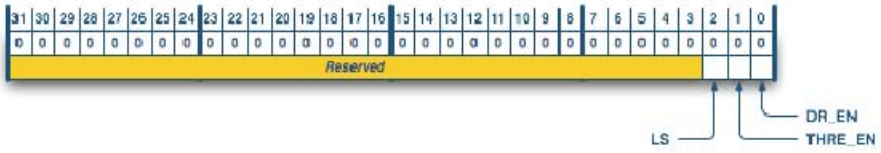
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Data (DATA):** Receive data.

10.3.2.2 Offset 00h: SC_THR – Transmit Holding

► ***Note:*** *RBR and THR share the same address space.*

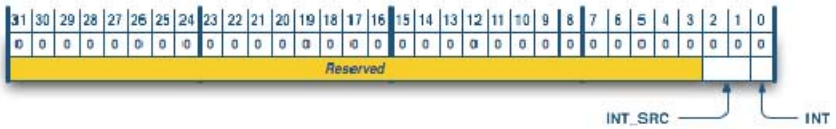
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Data (DATA):** Transmit data.

10.3.2.3 Offset 04h: SC_IER – Interrupt Enable



Bits	Type	Reset	Description
31:03	*RO*	*0*	*Reserved*
02	RW	0	**Rx Line Status Enable (LS):** When set, line status interrupt is enabled.
01	RW	0	**Tx Hold Register Empty Enable (THRE_EN):** When set, THRE interrupt enabled
00	RW	0	**Data Ready Enable (DR_EN):** When set, data ready interrupt is enabled

10.3.2.4 Offset 08h: SC_IIR – Interrupt Identification



Bits	Type	Reset	Description
31:03	*RO*	*0*	*Reserved*
02:01	RO	00	**Interrupt source (INT_SRC):** Interrupts are prioritized into three levels, as shown below: • 00: Not used • 01: THRE is source of interrupt • 10: DR is source of interrupt • 11: Line Status is source of interrupt, OE, PE, or FE.
00	RO	1	**Interrupt (INT):** When cleared, indicates an interrupt is pending.

10.3.2.5 Offset 0Ch: SC_LCR – Line Control

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Data Convention for RX Data (CNV):** When set, inverse convention. When cleared, normal convention.
06	RW	0	**Repetitions Interrupt (REP_INT):** When set, enables interrupt when the number of repetitions is reached.
05	RW	0	**Sticky Parity (STKY_PAR):** When set, sticky parity. When cleared, non-sticky parity.
04	RW	0	**Parity Select (PAR_SEL):** When set, even parity. When cleared, odd parity.
03	RW	0	**Parity Enable (PAR_EN):** When set, parity bit is generated or checked between the last data word bit and Stop bit of the serial data.
02	RW	0	**Stop Bits (STOP_BITS):** When set, enables the guard time (stop bit). The minimum is 1 stop bit before sending a new character. The receiver checks the first Stop bit only, regardless of the number of Stop bits selected. When cleared, one stop bit only.
01:00	RW	0	**Character Length (CHAR_LEN):** Specifies number of bits transmitted and received in each serial character. • 00: 5 characters • 01: 6 characters • 10: 7 characters • 11: 8 characters

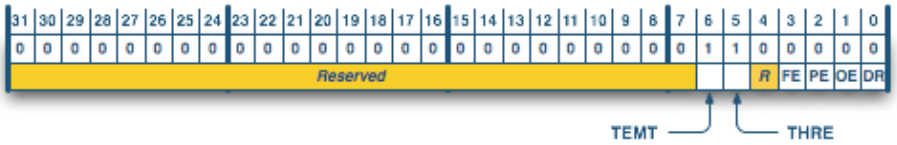
10.3.2.6 Offset 10h: SC_BRR – Baud Rate Register

This holds the factor for dividing the Smart Card clock to produce an ETU timing reference. The formula is:

$$1\,ETU = 10 \times BRR_{TUNING} + BRR_{FINE\_TUNE}$$

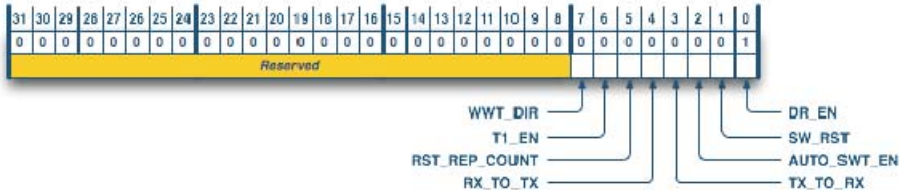
Bits	Type	Reset	Description
31:14	*RO*	*0*	*Reserved*
13:04	RW	000h	**Tuning (TUNING):** Tuning value for ETU timing reference.
3:0	RW	0h	**Fine Tune (FINE_TUNE):** Fine tune adder for ETU reference value 0h to 9h.

10.3.2.7 Offset 14h: SC_LSR – Line Status Register



Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
06	RO	1	**Transmitter Empty (TEMT):** When set, THR and TSR are both empty.
05	RO	1	**Transmitter Holding Register Empty (THRE):** When set, THR is empty.
04	*RO*	*0*	*Reserved*
03	RO	0	**Framing Error (FE):** When set, received character did not have a valid stop bit.
02	RO	0	**Parity Error (PE):** When set, received data character does not have the correct even or odd parity, as selected by the even parity select bit.
01	RO	0	**Overrun Error (OE):** When set, CPU did not read data from RBR before the next character was transferred into RBR, destroying the previous character.
00	RO	0	**Receive Data Ready (DR):** When set, data is ready in the receive buffer.

10.3.2.8 Offset 18h: SC_GR – Global



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**WWT Direction (WWT_DIR):** When set, this timer can be used for generating an interrupt when the card is talking too early (based upon the value WWT_TO). When cleared, WWT timeout is used when the CARD is talking too late (based upon WWT_TO).
06	RW	0	**T1 Enable (T1_EN):** When set, during reception no parity error is made on the I/O line and no parity error is signaled, and during transmission parity bit is sent but no error on the I/O line will be taken into account.
05	RW	0	**Reset Reception Counter (RST_REP_COUNT):** When set, automatically reset NUM_RX in reading mode.
04	RW	0	**Automatic Switch of RX to TX (RX_TO_TX):** When set, and AUTO_SWT_EN is set to '1', auto switches from RX to TX in the case of CWT error or an ATR length error.
03	RW	0	**Automatic Switch TX to RX (TX_TO_RX):** When set, auto switches from TX to RX just after the parity and 2-ETU of stop bit. When cleared, switches after total guard time.
02	RW	0	**Automatic Switch from TX to RX (AUTO_SWT_EN):** When set, the switch is made when the number of data to send is reached and after the second stop bit of the last send character. When cleared to '1', see TX_TO_RX description.
01	RW	0	**Software Reset (SW_RST):** When set, generates a software rest. Automatically clears after the reset is complete.
00	RW	1	**Communication Direction (DR_EN):** When set, receive. When cleared, transmit.

10.3.2.9 Offset 1Ch: SC_GTIME – Guard Time

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Guard Time (GUARD_TIME):** Provides the Guard Time value in ETUs. • 0x00: 11 ETU (1 stop bit) • 0x01: 12 ETU • 0x02: 13 ETU • ... • 0xFF: 266 ETU (256 stop bits)

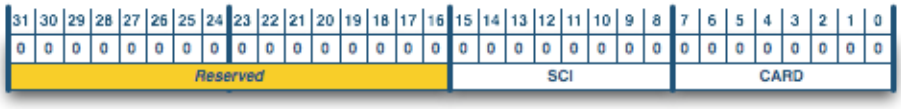
10.3.2.10 Offset 20h: SC_NUM_REP – Number of Repetition

Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	0h	**Number of repeated Parity Errors allowed in TX (RX?) Register (NUM_REP):** Holds the maximum number of repetitions in case of receive parity errors. • 0x0: 0 repetitions • 0x1: 1 repetitions • ... • 0xF: 15 repetitions

10.3.2.11 Offset 24h: SC_REV_DLY – Reverse Delay

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Number of Delay (DELAY_TIME):** • 0x00: 0 delay • 0x01: 1 delay • ... • 0xFF: 255 delay

10.3.2.12 Offset 28h: SC_CLK_DIV – Clock Divider

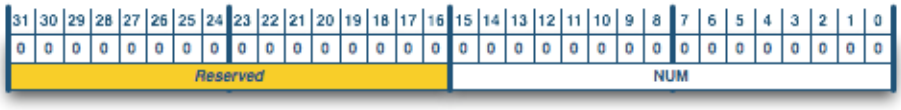


Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:08	RW	00h	**SCI Clock Divider (SCI):** Configures the programmable global clock divider situated before the Analog interface Chip.
07:00	RW	00h	**Card Clock Divider (CARD):** Configures the programmable global clock divider.

10.3.2.13 Offset 2Ch: SC_NUMTX – Number of Data to Send

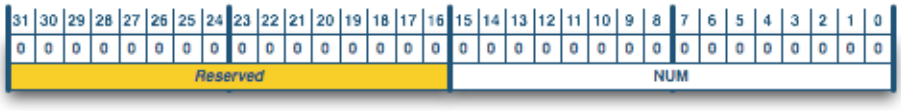
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**Number (NUM):** Defines the number of data transmissions. The number of transmission is the value in this register plus one. Automatic switch from transmission to reception will occur if IER register bit 13 is set.

10.3.2.14 Offset 30h: SC_NUMPE – Number of Repetition in Rx for Parity Error Before Interrupt



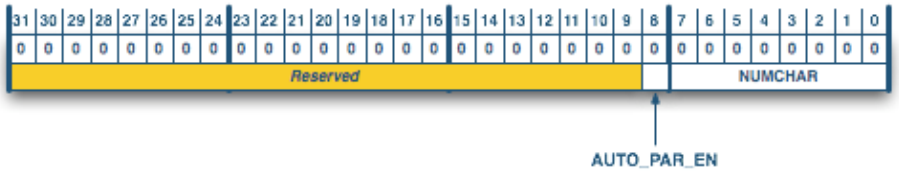
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**Number (NUM):** Defines the number of characters with parity error before generating an interrupt. The number of parity errors is the value in this register plus one.

10.3.2.15 Offset 34h: SC_NUMRX – Number of Receptions Before Interrupt



Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**Number (NUM):** Defines the number of characters to receive before generating an interrupt. RX interrupt must be enabled to use this register. The number of characters is the value in this register plus one.

10.3.2.16 Offset 38h: SC_AUTO_PARITY – Automatic Parity

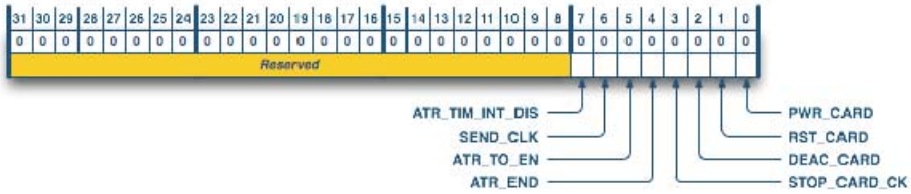


Bits	Type	Reset	Description
31:09	*RO*	*0*	*Reserved*
08	WO	0	**Automatic Parity Enable (AUTO_PAR_EN):** When set, enables Auto Parity.
07:00	WO	00h	**Number of Characters (NUMCHAR):** If AUTO_PAR_EN is set, these bits are the value to detect in Rx data for the FIRST_ATR byte. Parity is equal to the bit written to LCR.STKY_PAR, and if not equal parity is changed. When AUTO_PAR_EN is cleared to '0', parity is LCR.STKY_PAR. For configuring parity with LCR.STKY_PAR, you have to disable AUTO_PAR_EN and be idle state.

10.3.3 Smart Card Controller Registers (Base: SC0 → FFFF0100h, SC1 → FFFF0200h)

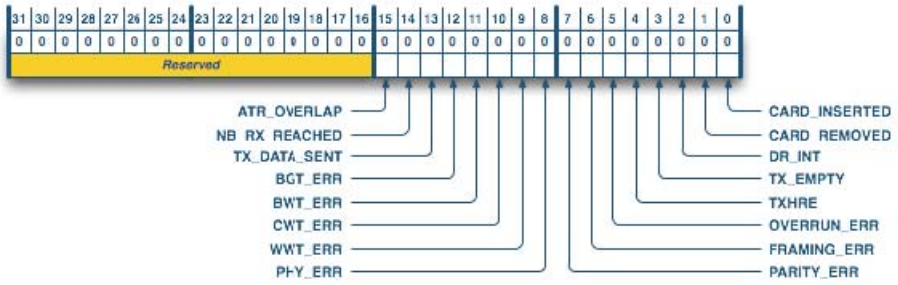
Offset	Register	Description
040h	SC_COMMAND	Command Register
044h	SC_INT_STAT	Interrupt Status Register
048h	SC_INT_EN	Interrupt Enable Register
04Ch	SC_TO_EN	Timeout Enable Register
060h	SC_RST_LEN	Reset length Register
064h	SC_ATR_MIN_DLY	Min ATR delay Register
068h	SC_ATR_MAX_DLY	Max ATR delay Register
06Ch	SC_ATR_MAX_LEN	Max ATR length Register
070h	SC_WWT_TO	WWT timeout Register
074h	SC_CWT_TO	CWT timeout Register
078h	SC_BWT_TO	BWT timeout Register
07Ch	SC_BGT_TO	BGT timeout Register
080h	SC_DEAC_DLY	Card deactivation delay Register
284h	SC_SIM_SEL	Selection of the SIM (Smart Card 1 Only)

10.3.3.1 Offset x40h: SC_COMMAND – Smart Card Command



Bits	Type	Reset	Description
31:09	*RO*	*0*	*Reserved*
07	RW	0	**ATR Timing Interrupt Disable (ATR_TIM_INT_DIS):** When set, disables all ATR timing and does not generate interrupt from ATR.
06	RW	0	**Send Clock (SEND_CLK):** When set, sends the clock without going through the states of the controller state machine. The sequence of activation of the Card Signals (VCC, Clock, IO and reset) must be made with the interface chip (ONSEMI) in parallel with the IP SIM. Not valid with Main Card.
05	RW	0	**ATR Timeout Enable (ATR_TO_EN):** When set, allows the use of the ATR timing without being in ATR. This bit is reset by writing 1 to ATR_END.
04	RW	0	**ATR End (ATR_END):** When set, signals the completion of the ATR reception. The bit is automatically reset when the command has been executed.
03	RW	0	**Stop Card Clock (STOP_CARD_CK):** When set, card clock is stopped. Can be issued anytime during the WaitCommand state of the controller.
02	RW	0	**De-activate Card (DEAC_CARD):** When set, starts de-activiation sequence. Can be issued anytime during the ***WaitCommand*** state of the controller. DEAC_DLY must be loaded with the correct value. This bit is reset when the command has been executed. **Note:** In the case where the card is removed during the ***ResetCard*** Controller state, a deactivation sequence is automatically started. Therefore, DEAC_DLY must be loaded with the correct value prior to a ResetCard command.
01	RW	0	**Reset Card Command (RST_CARD):** When set, issues a cold or warm reset of the card. This bit is reset when the command has been executed. When issued, a timer is started to count RST_LEN. In the case of a cold reset, software must ensure the power is stable before issuing the command. A warm reset is performed when the controller is in the ***WaitCommand*** state.
00	RW	0	**Power Card Command (PWR_CARD):** When set, the card is powered-on. This bit is reset when the command has been executed.

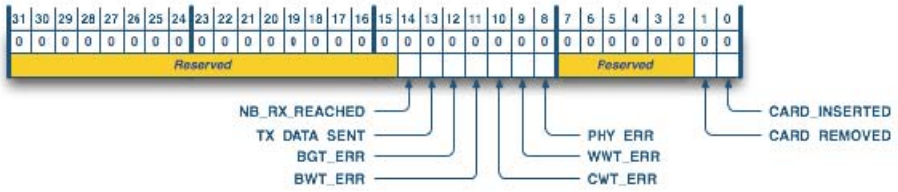
10.3.3.2 Offset x44h: SC_INT_STAT – Smart Card Interrupt Status



Status bits from SC_LSR are replicated in this register to allow software to identify the interrupt source. Reading this register clears LSR (except DR_INT which can only be reset by reading data).

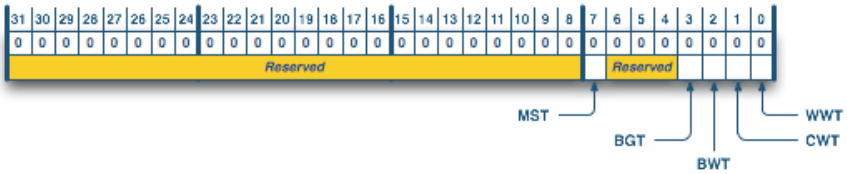
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15	RO	0	**ATR Overlap (ATR_OVERLAP):** When set, ATR length is reached and a data is overlapping. It is cleared upon reading the register.
14	RO	0	**Number of Reception Reached (NB_RX_REACHED):** When set, the number of received data has been reached. It is cleared upon reading the register.
13	RO	0	**TX Data Sent (TX_DATA_SENT):** When set, a byte has been sent by the UART. It is cleared upon reading the register, a soft reset, or when automatic switch is enabled and reception mode is activated.
12	RO	0	**BGT Error (BGT_ERR):** When set, a BGT error occurred. TO_EN.EN_BGT must be set to '1'. It is cleared on reading the register.
11	RO	0	**BWT Error (BWT_ERR):** When set, a BWT error occurred. TO_EN.EN_BWT must be set to '1'. Also occurs when an ATR delay error occurs. It is cleared on reading the register.
10	RO	0	**CWT Error (CWT_ERR):** When set, a CWT error occurred. TO_EN.EN_CWT must be set to '1'. Also occurs when an ATR length error occurs. It is cleared on reading the register.
09	RO	0	**WWT Error (WWT_ERR):** When set, a WWT error occurred. TO_EN.EN_WWT must be set to '1'. It is cleared on reading the register.
08	RO	0	**PHY Error (PHY_ERR):** When set, the number of character repetitions is reached (in transmission) due to continuous parity errors. It is cleared on reading the register
07	RO	0	**Parity Error (PARITY_ERR):** When set, a parity error occurred.
06	RO	0	**Framing Error (FRAMING_ERR):** When set, a framing error occurred.
05	RO	0	**Overrun Error (OVERRUN_ERR):** When set, an overrun error occurred.
04	RO	0	**TX Hold Register Empty (TXHRE):** When set, the TX Hold register is empty.
03	RO	0	**TX Empty (TX_EMPTY):** When set, TX is empty.
02	RO	0	**RX Data Ready (DR_INT):** When set, receive data is ready.
01	RO	0	**Card Removed (CARD_REMOVED):** When set, the Card Detect signal goes low. It is cleared on reading the register. No internal de-bouncing of the Card Detect signal is performed.
00	RO	0	**Card Inserted (CARD_INSERTED):** When set, the Card Detect signal goes high. It is cleared on reading the register. No internal de-bouncing of the Card Detect signal is performed.

10.3.3.3 Offset x48h: SC_INT_EN – Smart Card Interrupt Enable



Bits	Type	Reset	Description
31:15	*RO*	*0*	*Reserved*
14	RW	0	**Number of reception reach (NB_RX_REACHED):** When set, enables INT_STAT.NB_RX_REACHED to generate an interrupt.
13	RW	0	**TX Data Sent interrupt (TX_DATA_SENT):** When set, enables INT_STAT.TX_DATA_SENT to generate an interrupt.
12	RW	0	**BGT Error Interrupt (BGT_ERR):** When set, enables INT_STAT.BGT_ERR to generate an interrupt.
11	RW	0	**BWT Error Interrupt (BWT_ERR):** When set, a BWT error occurred. BWT timeout checking must be enabled in the Command register or when an ATR delay error occurs. It is cleared on reading the register.
10	RW	0	**CWT Error Interrupt (CWT_ERR):** When set, a CWT error occurred. CWT timeout checking must be enabled in the Command register or when an ATR length error occurs. It is cleared on reading the register.
09	RW	0	**WWT Error Interrupt (WWT_ERR):** When set, a WWT error occurs. WWT timeout checking must be enabled in the Command register. It is cleared upon reading the register
08	RW	0	**PHY Error Interrupt (PHY_ERR):** When set, indicates the number of character repetitions is reached (in transmission) due to continuous parity errors. It is cleared upon reading the register
07:02	*RO*	*0*	*Reserved*
01	RW	0	**Card Removed (CARD_REMOVED):** Set whenever the Card Detect signal goes Low. It is cleared upon reading the register. No de-bouncing is performed. It must be done externally.
00	RW	0	**Card Inserted (CARD_INSERTED):** Set whenever the Card Detect signal goes High. It is cleared upon reading the register. No de-bouncing is performed. It must be done externally.

10.3.3.4 Offset x4Ch: SC_TO_EN – Smart Card Timeout Enable

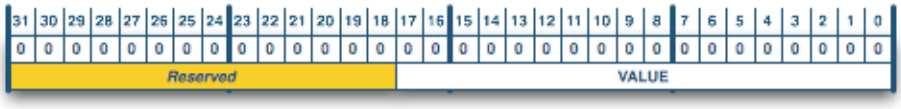


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Master Interrupt Enable (MST):** When set, use Interrupt Enable register
06:04	*RO*	*0*	*Reserved*
03	RW	0	**Enable BGT Timeout Checking (BGT):** When set, BGT timeout checking enabled
02	RW	0	**Enable BWT Timeout Checking (BWT):** When set, BWT timeout checking enabled
01	RW	0	**Enable CWT Timeout Checking (CWT):** When set, CWT timeout checking enabled
00	RW	0	**Enable WWT Timeout Checking (WWT):** When set, WWT timeout checking enabled

10.3.3.5 Offset x60h: SC_RST_LEN – Smart Card Reset Length

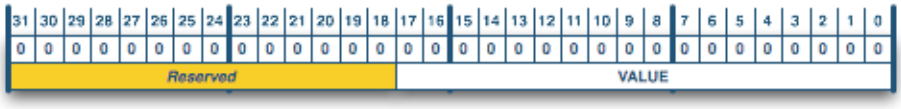
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**Length (LENGTH):** Defines the number of card clocks for reset from 0 to 65,535.

10.3.3.6 Offset x64h: SC_ATR_MIN_DLY – Smart Card ATR Minimum Delay



Bits	Type	Reset	Description
31:18	*RO*	*0*	*Reserved*
17:00	RW	00000h	**Value (VALUE)**: Defines number of card clocks for ATR minimum delay from 0 to 262,143 clocks.

10.3.3.7 Offset x68h: SC_ATR_MAX_DLY – Smart Card ATR Maximum Delay

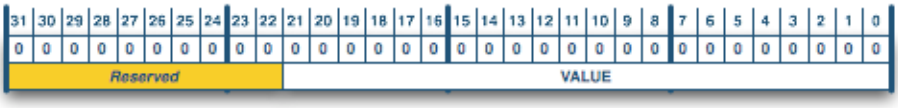


Bits	Type	Reset	Description
31:18	*RO*	*0*	*Reserved*
17:00	RW	00000h	**Value (VALUE):** Defines number of card clocks for ATR maximum delay from 0 to 262,143 clocks.

10.3.3.8 Offset x6Ch: SC_ATR_MAX_LEN – Smart Card ATR Maximum Length

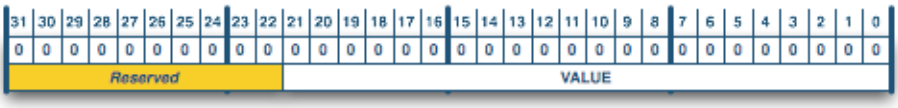
Bits	Type	Reset	Description
31:18	*RO*	*0*	*Reserved*
17:00	RW	00000h	**Value (VALUE):** Defines number of ETUs for ATR maximum length from 0 to 262,143.

10.3.3.9 Offset x70h: SC_WWT_TO – Smart Card WWT Timeout



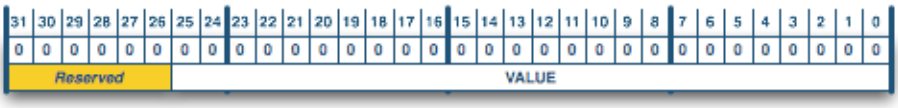
Bits	Type	Reset	Description
31:22	*RO*	*0*	*Reserved*
21:00	RW	000000h	**Value (VALUE):** Defines number of ETUs for WWT timeout from 0 to 4,194,303.

10.3.3.10 Offset x74h: SC_CWT_TO – Smart Card CWT Timeout



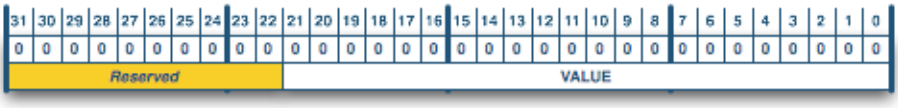
Bits	Type	Reset	Description
31:22	*RO*	*0*	*Reserved*
21:00	RW	000000h	**Value (VALUE):** Defines the number of ETUs for CWT timeout from 0 to 4,194,303.

10.3.3.11 Offset x78h: SC_BWT_TO – Smart Card BWT Timeout



Bits	Type	Reset	Description
31:26	*RO*	*0*	*Reserved*
25:00	RW	0000000h	**Value (VALUE):** Defines the number of ETUs for BWT TO from 0 to 33,554,432.

10.3.3.12 Offset x7Ch: SC_BGT_TO – Smart Card BGT Timeout



Bits	Type	Reset	Description
31:22	*RO*	*0*	*Reserved*

21:00	RW	0000h	**Value (VALUE):** Defines the number of ETUs for BGT timeout from 0 to 65,535.

10.3.3.13 Offset x80h: SC_DEAC_DLY – Smart Card Deactivate Delay

Bits	Type	Reset	Description
31:22	*RO*	*0*	*Reserved*
21:00	RW	000000h	**DEAC_DLY:** Defines number of ETUs for deactivation delay from 0 to 33,554,432.

10.3.3.14 Offset 284h: SC_SIM_SEL – Smart Card SIM Selection

Each SIM has its own controller state machine.

Bits	Type	Reset	Description
31:02	*RO*	*0*	*Reserved*
01:00	RW	00	**Sim Selection (SIM_SEL):** Selects the appropriate SIM card to receive command. • 00: SIM1 • 01: SIM2 • 10: SIM3 • 11: SIM4

Chapter 11: Real-Time Clock (RTC)

The real-time clock (RTC) keeps time by maintaining a count of seconds, minutes, hours, day-of-the-month, month, and year. The current time is kept in 24-hour format. All count and alarm registers are in binary format. The calendar operation maintains the correct day of the month and automatically compensates for leap year.

The RTC has a dedicated power supply and maintains operation through the external battery when main power is not available. It contains an alarm which can be used to generate an interrupt or to generate a wake condition.

11.1 Real-Time Clock Time/Counter Registers

The time is accessible via a number of RTC registers. The seconds, minutes, hours, day-of-the-month, the month, and year can be read or written through the RTC_SEC, RTC_MIN, RTC_HRS, RTC_DOM, RTC_MON, and RTC_YR, respectively.

An additional register, RTC_TIME can be used to read all the time registers in a single 32-bit value. However, only 6 LSB of the RTC_YR are available in this register.

> ►***Note:*** *All read and write operations to these registers must be followed by a read-verify operation. In the case of reads, a second read must be executed and verified as equal to the first read. For writes, the written register must be read back and verified as equal to the expected value. In both cases, this operation must be repeated until verified as correct. This is due to the asynchronous nature of the 32 kHz clock relative to the hclk. In rare cases, complete incorrect values may be read or written. If the RTC APB interface is locked, the RTC cannot be written to.*

11.2 RTC Alarm

An alarm can be programmed when the current count matches the alarm set-point registers. Alarm registers are available for seconds, minutes, hours, day-of-the-month, month, and year. Each alarm register can be independently enabled in RTC_ACTRL. For example, if the minute and hour alarms are both enabled (while the seconds, day-of-the-month, the month, and year registers are disabled), the alarm only occurs at the specified minute and hour. The alarm triggers an interrupt if RTC_CTRL.ALARM_EN is set to '1'. When this occurs, RTC_CTRL.ALARM_STATUS is set to '1'. It can be cleared by writing a '1' to RTC_CTRL.ALARM_STATUS. Alarm value and control registers can be written at any time. The comparison of the alarm value to the time value is done once every second.

11.3 RTC Wake

Whenever an RTC alarm is generated, a wake indication is also sent to the PMU. For the wake to result in activation of disabled clocks, the PMU must also be programmed. See section Chapter 3: for more details.

11.4 RTC Oscillator Source

The RTC contains an internal oscillator which must be connected to an external 32.768 kHz crystal. The crystal must be connected to the RTCXI and RTCXO pins.

11.5 RTC Battery Backup

The RTC derives its power from system power, when available. When system power is not available, the RTC derives its power from the battery power supply pin (V_{BAT}). Irrespective of the power source, the RTC continues driving the oscillator and keeping accurate time while valid power remains on at least one of the power sources.

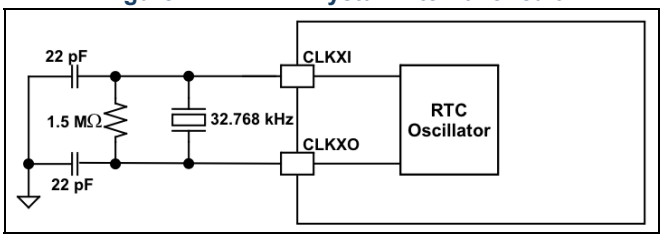
11.6 RTC Reset

The alarm, control and time registers of the RTC are not reset in the event of a system reset.

11.7 Oscillator External Circuit

A 32.768 kHz crystal is required for operation of the RTC oscillator.

Figure 11-1: RTC Crystal External circuit



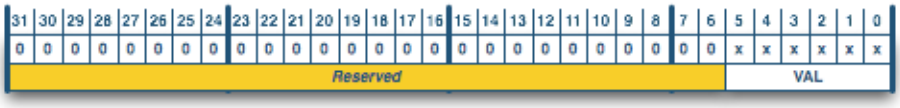
11.8 RTC Registers (Base → FFFFB000h)

These registers are unknown on initial power up of the battery voltage.

11.8.1 Current Time Registers

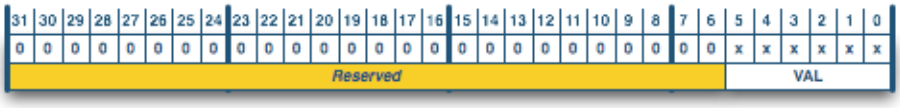
Offset	Register	Description
000h	RTC_SEC	Current Seconds
004h	RTC_MIN	Current Minutes
008h	RTC_HRS	Current Hours
00Ch	RTC_DOM	Current Day-of-the-month
010h	RTC_MON	Current Month
014h	RTC_YR	Current Year
018h	RTC_TIME	Current Time

11.8.1.1 Offset 000h: RTC_SEC – Current Seconds



Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05:00	RW	Undef	**Value (VAL):** Stores the current seconds count. Maximum value is 59.

11.8.1.2 Offset 004h: RTC_MIN – Current Minutes



Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05:00	RW	Undef	**Value (VAL):** Stores the current minutes count. Maximum value is 59.

11.8.1.3 Offset 008h: RTC_HRS – Current Hours

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Stores the current hours count. Maximum value is 23.

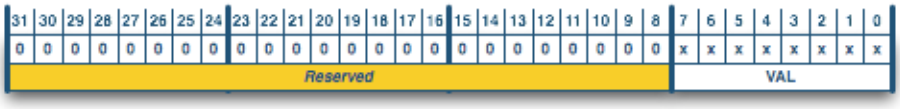
11.8.1.4 Offset 00Ch: RTC_DOM – Current Day-of-the-Month

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Stores the current day-of-the-month count, including leap year calculations. The counter does not correctly exclude leap days for the years 1900, 2100, i.e. every 200th year. 1 = 1st day, 2 = 2nd day, etc.

11.8.1.5 Offset 010h: RTC_MON – Current Month

Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	Undef	**Value (VAL):** Stores the current month count. 0 = N/A. 1 = January, 12 = December

11.8.1.6 Offset 014h: RTC_YR – Current Year



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	Undef	**Value (VAL):** Stores the current year count. The current year is 1900+YEAR. Valid for 1900–2155. Example: 0x65 = 2001

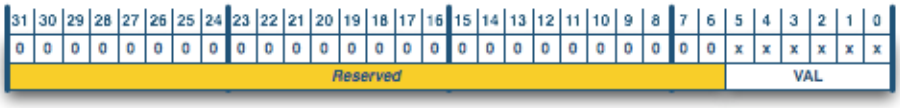
11.8.1.7 Offset 018h: RTC_TIM – Current Time

Bits	Type	Reset	Description
31:26	RO	Undef	**Year (YR):** 6 least significant bits of the 8 bit year value. The current year is 1900+YEAR. Valid for 1900–2155.
25:22	RO	Undef	**Month (MON):** Stores the current month count. 1 = January, 12 = December. 0 = N/A
21:17	RO	Undef	**Day of Month (DOM):** Stores the current day-of-the-month count, including leap year calculations. The counter does not correctly exclude leap days for the years 1900, 2100, i.e. every 200th year. 1 = 1st day, 2 = 2nd day, etc.
16:12	RO	Undef	**Hours (HRS):** Stores the current hours count. Maximum value = 23
11:06	RO	Undef	**Minutes (MIN):** Stores the current minutes count. Maximum value = 59
05:00	RO	Undef	**Seconds (SEC):** Stores the current seconds count. Maximum value = 59

11.8.2 Alarm Registers

Offset	Register	Description
01Ch	RTC_ASEC	Alarm Seconds
020h	RTC_AMIN	Alarm Minutes
024h	RTC_AHRS	Alarm Hours
028h	RTC_ADOM	Alarm Day-of-the-month
02Ch	RTC_AMON	Alarm Month
030h	RTC_AYR	Alarm Year

11.8.2.1 Offset 01Ch: RTC_ASEC – RTC Alarm Seconds



Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05:00	RW	Undef	**Value (VAL):** Maximum value is 59.

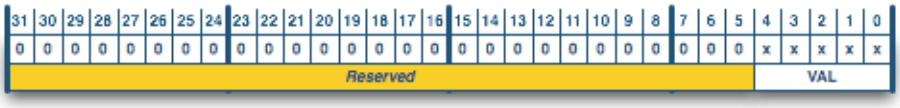
11.8.2.2 Offset 028h: RTC_AMIN – RTC Alarm Minutes

Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05:00	RW	Undef	**Value (VAL):** Maximum value is 59.

11.8.2.3 Offset 024h: RTC_AHRS – RTC Alarm Hours

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Maximum value is 23.

11.8.2.4 Offset 028h: RTC_ADOM – RTC Alarm Day-of-the-Month



Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04:00	RW	Undef	**Value (VAL):** Value must be between 1 and 31, inclusive.

11.8.2.5 Offset 02Ch: RTC_AMON – RTC Alarm Month

Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	Undef	**Value (VAL):** Value must be between 1 and 12, inclusive.

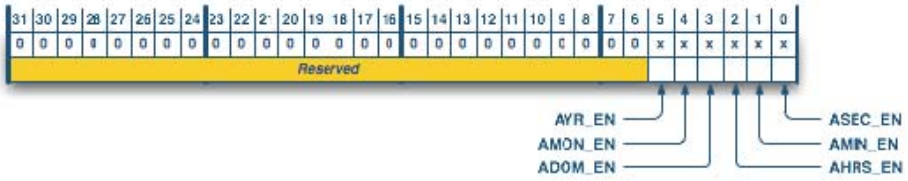
11.8.2.6 Offset 030h: RTC_AYR – RTC Alarm Year

Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
07:00	RW	Undef	**Value (VAL):** Value must be between 0 and 255, inclusive.

11.8.3 Control Registers

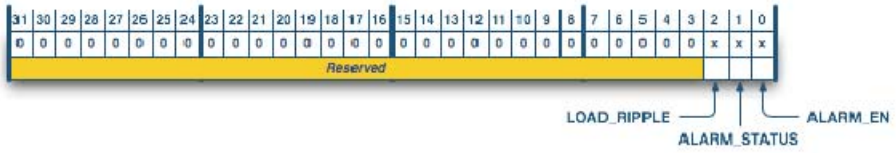
Offset	Register	Description
034h	RTC_ACTRL	Alarm Control
038h	RTC_CTRL	RTC Control

11.8.3.1 Offset 034h: RTC_ACTRL – RTC Alarm Control



Bits	Type	Reset	Description
31:06	*RO*	*0*	*Reserved*
05	RW	Undef	**Year Alarm Enable (AYR_EN):** When set, include year in alarm compare.
04	RW	Undef	**Month Alarm Enable (AMON_EN):** When set, include month in alarm compare.
03	RW	Undef	**Day-of-the-Month Alarm Enable (ADOM_EN):** When set, Include day in alarm compare.
02	RW	Undef	**Hour Alarm Enable (AHRS_EN):** When set, include hour in alarm compare.
01	RW	Undef	**Minute Alarm Enable (AMIN_EN):** When set, include min in alarm compare.
00	RW	Undef	**Seconds Alarm Enable (ASEC_EN):** When set, include sec in alarm compare.

11.8.3.2 Offset 038h: RTC_CTRL – RTC Control

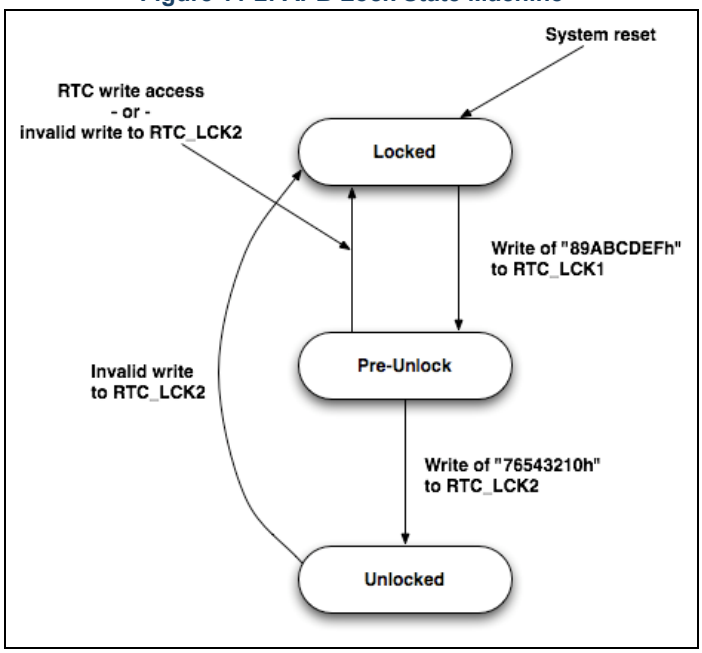


Bits	Type	Reset	Description
31:03	*RO*	*0*	*Reserved*
02	WO	0	**Load Ripple (LOAD_RIPPLE):** This bit is used only for manufacturing test. Writing one to this bit forces the RTC ripple counter to a value such that a 1 Hz clock positive edge is generated upon the next rising edge of the 32 kHz clock. Always returns '0' on reads.
01	RW1C	Undef	**RTC interrupt status (ALARM_STATUS):** The interrupt is activated on an RTC alarm compare. When set, the interrupt is active.
00	RW	Undef	**Interrupt on Alarm Condition (ALARM_EN):** When set, enabled.

11.9 RTC Locking

Access to the RTC is locked upon a system reset of the main system, or an invalid write to the RTC_LCK2. While locked, writes are disabled to the RTC except to RTC_LCK1 and RTC_LCK2. APB may be identified as “unlocked” by reading RTC_APB_STA.APB_UNLOCKED. The lock is provided to prevent spurious writes during power up and down of the main system.

Figure 11-2: APB Lock State Machine



Address	Register	Description
FFFFC00Ch	RTC_APB_STA	RTC APB Status Register
FFFFC694h	RTC_LCK1	RTC Lock 1 Register
FFFFC968h	RTC_LCK2	RTC Lock 2 Register

11.9.1 Address FFFFC00Ch: RTC_APB_STA – RTC APB Status Register

Bits	Type	Reset	Description
31	RO	1	**APB Unlocked (APB_UNLOCKED):** Indicates if the APB bus is locked or unlocked. Security accesses and secure memory are not possible if APB is locked.
30:00	*RO*	*0*	*Reserved*

11.9.2 Address FFFFC694h: RTC_LCK1 – RTC Lock 1 Register

Bits	Type	Reset	Description
31:00	WO	0	**Value (VAL):** First of two registers which must be written to unlock accesses to the RTC registers. A write of 89ABCDEFh must be done to start the sequence, and a write of 76543210h to RTC_LCK2 completes the sequence. If any other write occurs to the RTC between these two writes, the RTC remains locked.

11.9.3 Address FFFFC698h: RTC_LCK2 – RTC Lock 2 Register

Bits	Type	Reset	Description
31:00	WO	0	**Value (VAL):** Second of two registers which must be written to unlock accesses to the RTC registers. See description for RTC_LCK1. Once unlocked, APB remains unlocked until: • A hard reset of the main system • This register is written with a value other than 0x76543210. • A read access of this write only register

Chapter 12: Random Number Generator (RNG)

The Random Number Generator (RNG) is an APB device with the following features:

- True 32-bit random number generation
- Interrupt Generation on completion of random number
- Generation rate of 1 number in less than 256 hclk cycles
- NIST 800-22 Compliant

12.1 Programming Guide

1. Ensure that RNG_CTRL.REQ and RNG_CTRL.GRANT are cleared to '0'.
2. Set RNG_CTRL.REQ to '1'. If an interrupt is desired also set RNG_CTRL.IRQ_EN to '1'. RNG_CTRL.IRQ_CLR must be cleared to '0'.
3. Wait for RNG_CTRL.GRANT to be set '1', either by polling or receiving an IRQ (if RNG_CTRL.IRQ_EN was set).
4. Clear RNG_CTRL.REQ and RNG_CTRL.IRQ_CLR to '0'. This action clears RNG_CTRL.GRANT bit as well as the contents of RNG_DATA.
5. If active, the interrupt can be cleared by writing RNG_CTRL.IRQ_CLR to '1'.

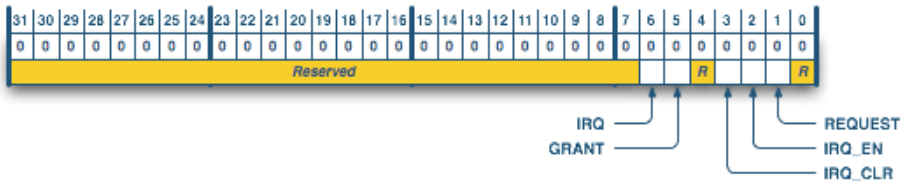
12.2 RNG Registers (Base → FFFFA000h)

Offset	Register	Description
000h	RNG_DATA	RNG Data Register
004h	RNG_CTRL	RNG Control Register

12.2.1 Offset 000h: RNG_DATA – Random Number Generator Data Register

Bits	Type	Reset	Description
31:00	RO	0	**Random Number (NUM):** Valid when RNG_CTRL.GRANT is '1'. Clearing RNG_CTRL.REQ clears this register.

12.2.2 Offset 004h: RNG_CTRL – Random Number Generator Control Register



Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
06	RO	0	**IRQ Status (IRQ):** Set when GRANT is set, and IRQ_EN was '1'.
05	RO	0	**Grant (GRANT):** When set, indicates a new random number is available in RNG_DATA. This goes active some time after REQUEST is set. Once set, it remains set until REQUEST is cleared.
04	*RO*	*0*	*Reserved*
03	WO	0	**IRQ Clear (IRQ_CLR):** Writing 1 to this bit causes the IRQ to be cleared. Any write with this bit set does not affect other bits of this register. Writes of '0' have no effect.
02	RW	0	**IRQ Enable (IRQ_EN):** When set, enables IRQ to go active when a GRANT transitions to 1. Any write to RNG_CTRL with the IRQ CLR bit set do not affect this bit.
01	RW	0	**Request (REQUEST):** When set, initiates generation of a new random number. After this bit is set, GRANT indicates a new random number is available. Once GRANT is set to '1', this bit must be cleared. Clearing REQUEST will have following two effects: • GRANT is cleared. • The contents of RNG_DATA are cleared. Any write to RNG_CTRL with IRQ_CLR set do not affect this bit.
00	*RO*	*0*	*Reserved*

Chapter 13: SHA-1

The SHA-1 module provides a message digest for given text of almost any length per the SHA-1 protocol.

13.1 Programming Guide

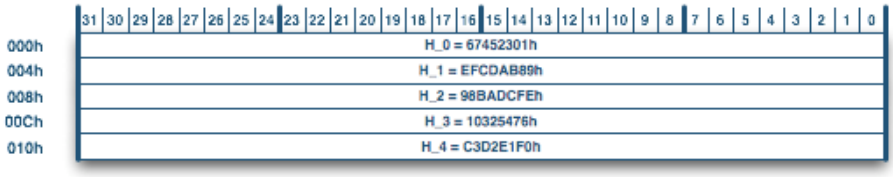
To generate a hashed signature value, execute the following actions:

1. If the existing hash value is acceptable, set SHA1_CONTROL.INIT to '1'. Alternatively, write SHA1_WH with a hash value, and set SHA1_CONTROL.INIT and SHA1_CONTROL.INIT_SEL to '1'.
2. Write a value to SHA1_DATA_IN.DATA. Sixteen writes to W_IN must be performed to provide data to the SHA-1. DMA can be used to provide these values.
3. Wait for 65 clocks and read SHA1_STATUS.VALID. Once this bit is set, SHA1_H contains the output. SHA1_STATUS.VALID remains set until all 5 registers are read.

13.2 SHA-1 Registers (Base → FFFF9000h)

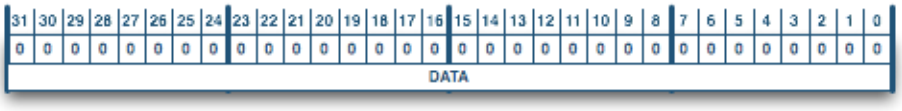
Address	Register	Description
000h – 010h	SHA1_H	Hashed value 0-4 from SHA-1
014h	SHA1_DATA_IN	Data IN register
018h	SHA1_CONTROL	SHA-1 Control
01Ch	SHA1_STATUS	SHA-1 Status Control
020h – 030h	SHA1_WH	Initial hash value 0-4

13.2.1 Offset 000h: SHA1_H – Hashed Value



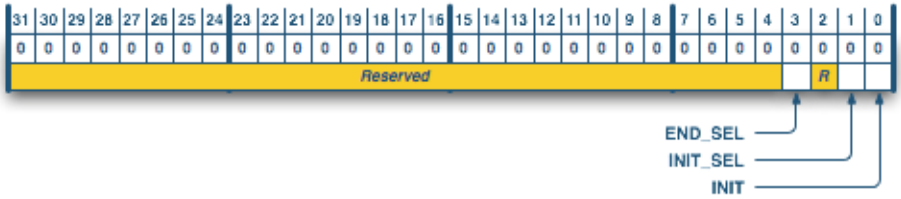
Bits	Type	Reset	Description
159:127	RO	C3D2E1F0h	**H-4 (H_4):** This value is valid when STATUS.VALID is '1'.
127:96	RO	10325476h	**H-3 (H_3):** This value is valid when STATUS.VALID is '1'.
95:64	RO	98BADCFEh	**H-2 (H_2):** This value is valid when STATUS.VALID is '1'.
63:32	RO	EFCDAB89h	**H-1 (H_1):** This value is valid when STATUS.VALID is '1'.
31:00	RO	67452301h	**H-0 (H_0):** This value is valid when STATUS.VALID is '1'.

13.2.2 Offset 014h: SHA1_DATA_IN – Data In



Bits	Type	Reset	Description
31:00	WO	00000000h	**Data (DATA):** 32-bit value for SHA-1.

13.2.3 Offset 018h: SHA1_CONTROL – SHA-1 Control



Bits	Type	Reset	DescriptionDS0200.docx
31:02	*RO*	*0*	*Reserved*
03	WO	0	**Endian Select (END_SEL):** Set to match the Endian type of data used. When written to '0', little Endian data. When written to '1', big Endian data
02	*RO*	*0*	*Reserved*
01	WO	0	**Initialization Select (INIT_SEL):** When set, WH_0 through WH_4 are copied to H_0 to H_4 when INIT is set to '1'.
00	WO	0	**Initialize (INIT):** Initializes to start the SHA-1. When written to '0', no effect. When written to '1', initializes SHA-1 registers

13.2.4 Offset 01Ch: SHA1_STATUS – SHA-1 Status

Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	RO	0	**Valid (VALID):** When set, value is valid to read from H_0 through H_4. After H_0 through H_4 are read, the bit clears to '0'.

13.2.5 Offset 020h: SHA1_WH – SHA-1 Initialization Value

Bits	Type	Reset	Description
159:128	RW	00000000h	**Initial H_4 Value (WH_4):** Copied to H_4 when CONTROL.INIT_SEL is set.
127:96	RW	00000000h	**Initial H_3 Value (WH_3):** Copied to H_3 when CONTROL.INIT_SEL is set.
95:64	RW	00000000h	**Initial H_2 Value (WH_2):** Copied to H_2 when CONTROL.INIT_SEL is set.
63:32	RW	00000000h	**Initial H_1 Value (WH_1):** Copied to H_1 when CONTROL.INIT_SEL is set.
31:00	RW	00000000h	**Initial H_0 Value (WH_0):** Copied to H_0 when CONTROL.INIT_SEL is set.

Chapter 14: Analog-to-Digital Converter (ADC)

The ADC is an APB device with the following features:

- 10-bit resolution, 45 kHz, successive-approximation ADC
- Multiplexing to support 4 channel inputs
- 4 sample FIFO
- Support for DMA
- SINGLE SHOT or CONTINUOUS sample modes
- Programmable ADC clock
- 12 cycle conversion time (ADC clock cycles)
- Power down to <1 µA
- Differential Nonlinearity (DNL) ± 1 LSB
- Integral Nonlinearity (INL) ± 2 LSB

14.1 Voltage Reference

The ADC requires an external V_{REF}. The accuracy of conversions depends on the quality of V_{REF} supplied. V_{REF} can be tied to the analog V_{DD} for the ADC but be sure to filter the noise to achieve optimum performance.

14.2 Clock / Sample Rate

The ADC clock is derived from hclk. ADC_CFG is used to configure the divider from 2 up to 256. This value must be selected to produce an ADC clock slower than 540 kHz. The conversion time is 12 ADC clocks which provides a maximum sample rate of 45 kHz.

14.3 Modes of Operation

14.3.1 Continuous Rotating

In this mode, the ADC samples every 12 ADC clocks and is capable of rotating between the 6 input channels. To use this mode, decide the number of samples to be in the rotation. This can range from 1–8 and is programmed in ADC_CMD. The channels sampled are programmed in ADC_CFG. Start the sequence by setting ADC_CMD.COMP_SMPL to ‘1’. Sampling is stopped by clearing this bit.

- **Example 1 (channel 3):** ADC_CMD.ROT_LIMIT=001, and ADC_CFG.ACH_SEL = 001. Channel 3 is sampled continuously at the rate of ADC Clock / 12.
- **Example 2 (channel 1,2,1,3):** ADC_CMD.ROT_LIMIT=100, and ADC_CFG fields ACH_SEL=001, BCH_SEL=010, CCH_SEL=001, and DCH_SEL=011. This samples 1-2-1-3 -- 1-2-1-3 -- etc. (A,B,C,D -- A,B,C,D -- etc.) continuously at the rate of ADC clock / 12.

14.3.2 Single Shot

This mode performs one sample of a single channel. To use this mode, select the channel in ADC_CFG and then set ADC_CMD.SNGL_SMPL to ‘1’. Once sampling is complete, it is placed in the FIFO.

14.4 DMA Operation

DMA requests can be generated based on the number of entries in the FIFO. ADC_INT.FIFO_LEVEL and ADC_INT.REQ_EN are used to control DMA requests.

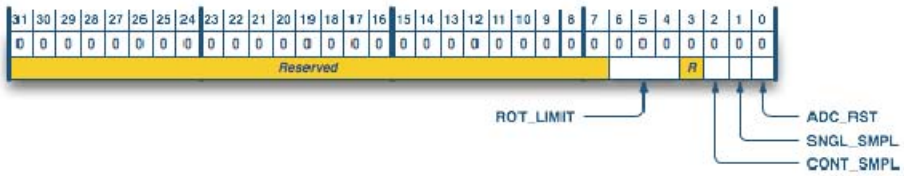
14.5 Registers (Base → FFFF2000h)

Offset	Register	Description
000h	ADC_CFG	ADC Configuration Register
004h	ADC_CMD	ADC Command Register
008h	ADC_FIFO	ADC FIFO Register
00Ch	ADC_INT	ADC Interrupt Register
010h	ADC_STA	ADC Status Register

14.5.1 Offset 000h: ADC_CFG – ADC Configuration Register

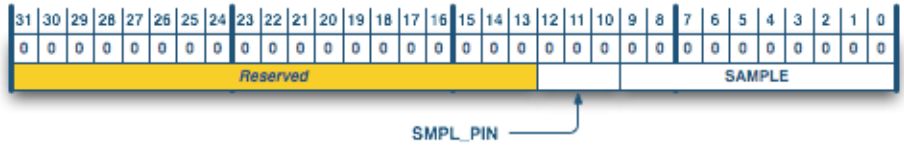
Bits	Type	Reset	Description
31:29	RW	000	**H-Channel Select (HCH_SEL):** • 000: N/A • 001: ADC_IN[1] pin • 010: ADC_IN[2] pin • 011: ADC_IN[3] pin • 100: ADC_IN[4] pin • 101 - 111: N/A
28:26	RW	000	**G-Channel Select (GCH_SEL):** Same encoding as HCH_SEL
25:23	RW	000	**F-Channel Select (FCH_SEL):** Same encoding as HCH_SEL
22:20	RW	000	**E-Channel Select (ECH_SEL):** Same encoding as HCH_SEL
19:17	RW	000	**D-Channel Select (DCH_SEL):** Same encoding as HCH_SEL
16:14	RW	000	**C-Channel Select (CCH_SEL):** Same encoding as HCH_SEL
13:11	RW	000	**B-Channel Select (BCH_SEL):** Same encoding as HCH_SEL
10:08	RW	000	**A-Channel Select (ACH_SEL):** Same encoding as HCH_SEL
07:00	RW	00h	**ADC Clock Divider (ADC_CLK_DIV):** Value used to derive the ADC clock from the ADC hclk. This clock is driven into the analog section of the ADC and must be configured to ensure that the divided clock is less than 600 kHz. The sample rate is 1/12 of the divided clock frequency. • 0: No clock to ADC (analog portion) • 1: ADC Divided Clock = hclk / 2 • 2: ADC Divided Clock = hclk / 3 • … • 255: ADC Divided Clock = hclk / 256

14.5.2 Offset 004h: ADC_CMD – ADC Command Register



Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
06:04	RW	000	**Rotation Limit (ROT_LIMIT):** Indicates how many A-H Channel Select are involved in the rotation of continuous sampling. These bits are only effective in the continuous sample mode. • 0: A Channel only • 1: A, B Channels • 2: A, B, C Channels • 3: A, B, C, D Channels • ..... • 7: A, B, C, D, E, F, G, H Channels
03	*RO*	*0*	*Reserved*
02	RW	0	**Continuous Sample (CONT_SMPL):** When set, enters continuous sample mode. This bit has lower priority than the single sample control.
01	RW	0	**Single Sample (SNGL_SMPL):** When set, initiates a single conversion. The analog sample input pin is selected from ADC_IN[4:1] as programmed in the A-CH SEL bits of ADC_CFG. This bit always reads 0. This bit has no effect if Continuous Sample is enabled.
00	RW	0	**ADC Reset (ADC_RST):** When set, resets the FIFO and status bits of ADC, and holds the ADC in reset until this bit is cleared to '0'.

14.5.3 Offset 008h: ADC_FIFO – ADC FIFO

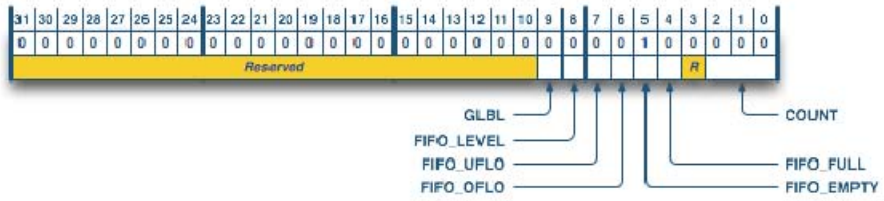


Bits	Type	Reset	Description
31:13	*RO*	*0*	*Reserved*
12:10	RO	000	**Sample Pin (SMPL_PIN):** These bits indicate the ADC_IN[4:1] pin associated with the ADC Sample. In case of an underflow, the content is undefined. • 000: N/A • 001: Sample holds data from ADC_IN[1] • 010: Sample holds data from ADC_IN[2] • 011: Sample holds data from ADC_IN[3] • 100: Sample holds data from ADC_IN[4] • 101 - 111: N/A
09:00	RO	000h	**ADC Sample (SAMPLE):** 10-bit ADC sample that has been moved into the FIFO. If the FIFO is empty, the underflow bit is set and the content is undefined.

14.5.4 Offset 00Ch: ADC_INT – ADC Interrupt Register

Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09	RW	0	**Global Enable (GLBL_IE):** When set, enables ADC interrupts.
08	RW	0	**FIFO Level Enable (FIFO_LEVEL):** When set, enables interrupts on FIFO Level condition.
07	RW	0	**FIFO Underflow Enable (FIFO_UNDERFLOW):** When set, enables interrupts on FIFO underflow condition.
06	RW	0	**FIFO Overflow Enable (FIFO_OVERFLOW):** When set, enables interrupts on FIFO overflow condition.
05:04	*RO*	*0*	*Reserved*
03	RW	0	**DMA Request Enable (DMA_REQUEST_EN):** When set, enables DMA requests to the DMA controller when the number of FIFO entries is greater than the FIFO interrupt level. This bit is not directly related to ADC interrupts.
02:00	RW	000	**FIFO Interrupt Level (FIFO_INT_LEVEL):** Sets the number of samples in the FIFO before ADC_STA.FIFO_LVL is set. This bit is also used with DMA_REQUEST_EN to generate DMA requests. Encoding: • 000: One or more entries • 001: Two or more entries • 010: Three or more entries • 011: Four entries • 100–111: Invalid

14.5.5 Offset 010h: ADC_STA – ADC Status Register



Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09	RO	0	**Global Interrupt Status (GLBL_INT):** When set, an ADC interrupt active
08	RW1C	0	**FIFO Level Status (FIFO_LVL):** When set, FIFO count is at or above level
07	RW1C	0	**FIFO Underflow Status (FIFO_UFLO):** When set, FIFO underflow has occurred
06	RW1C	0	**FIFO Overflow Status (FIFO_OFLO):** When set, FIFO overflow has occurred.
05	RO	1	**FIFO Empty Status (FIFO_EMPTY):** When set, FIFO is empty.
04	RO	0	**FIFO Full Status (FIFO_FULL):** When set, FIFO is full
03	*RO*	*0*	*Reserved*
02:00	RO	000	**FIFO Count: (COUNT):** Indicates the number of sample entries stored in the FIFO. The FIFO can hold up to 4 samples. • 000: FIFO empty • 001: one sample • 010: two samples • 011: three samples • 100: four samples • 101 – 111: N/A

Chapter 15: LCD Interface

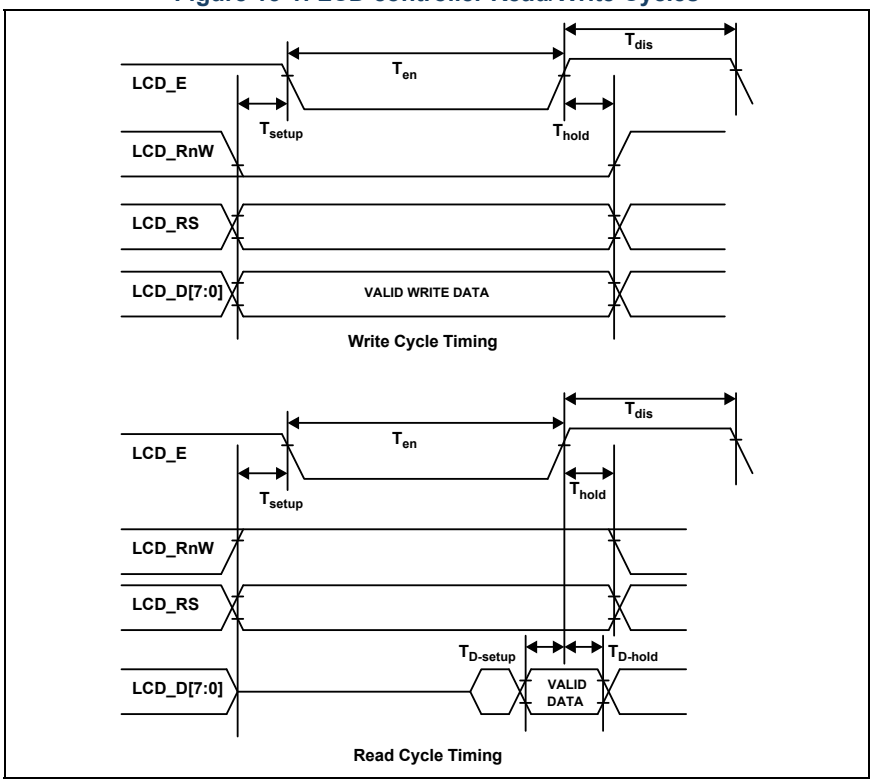
The LCD interface is an APB device providing an interface to an external LCD display. The attached display controller may be character or pixel based, and it may support only black and white, or full color. Features:

- Directly compatible with popular LCD display, text, and graphics modes.
- Supports interfaces of 4-bit or 8-bit data and 3 controls.
- Read and writes to external LCD module supported register operations.
- Programmable read and write cycle times.

15.1 Interface Timing

The following figure is a timing diagrams for read and write cycles generated by the LCD interface In the following figure, LCD_E is shown programmed as active Low.

Figure 15-1: LCD controller Read/Write Cycles



15.1.1 Read Cycle

Read cycles are generated by the LCD interface whenever there is a pending read request through the LCD_RD register.

15.1.2 Write Cycle

Write cycles are generated automatically by the LCD interface whenever there is a pending write in LCD_WR and there is no pending read. The values for T_{SETUP}, T_{HOLD}, T_{EN}, and T_{DIS} are programmable through LCD_CFG. Values for $T_{D\text{-}SETUP}$ and $T_{D\text{-}HOLD}$ are not programmable and are nominally 10ns and 0ns, respectively (for exact parameters, see section Chapter 21:. The polarity of LCD_E is also programmable.

15.2 Read and Write Commands

LCD_RD and LCD_WR generate single reads and writes to the LCD device. The command bits self-clear once the command is executed. Software must poll these bits and find them cleared before writing another command. Writes to LCD_RD and LCD_WR registers while the command bits are set must be avoided.

15.3 4-bit and 8-bit Operation

If configured for 8-bit operation, a read or write command results in a single 8-bit operation. If configured for 4-bit operation, two 4-bit operations (High nibble first) are executed one after the other for each read or write command. In this mode, only LCD_D[7:4] bits are used and LCD_D[3:0] bits are not used.

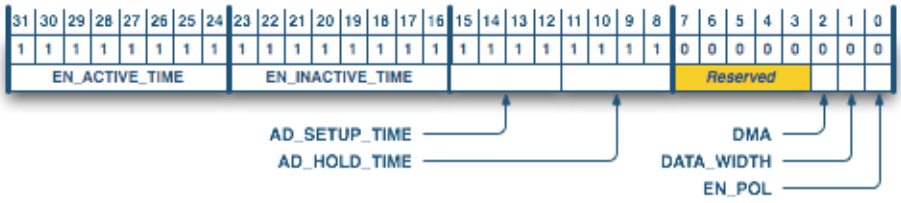
15.4 DMA Operation

By setting LCD_CFG.DMA to '1', a DMA request is generated if there is no pending write command. DMA must be configured to move a single 16-bit data into LCD_WR.

15.5 LCD Interface Registers (Base → FFFED000h)

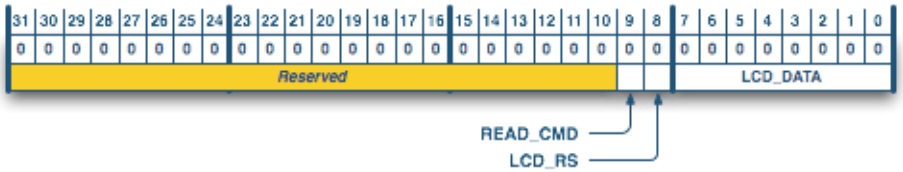
Offset	Register	Description
000h	LCD_CFG	LCD Configuration Register
004h	LCD_RD	LCD Read Register
008h	LCD_WR	LCD Write Register

15.5.1 Offset 000h: LCD_CFG – LCD Configuration Register



Bits	Type	Reset	Description
31:24	RW	FFh	**Enable Active Time (EN_ACTIVE_TIME):** Specifies the number of hclks LCD_E is driven active. • 00h: 2 clocks, 01h: 4 clocks, ... FFh: 512 clocks
23:16	RW	FFh	**Enable Inactive Time (EN_INACTIVE_TIME):** Specifies the number of hclks LCD_E is driven inactive before it is enabled to be driven active again. • 00h: 32 clocks, 01h: 64 clocks, ... FFh: 8192 clocks
15:12	RW	Fh	**Address/Data Setup Time (AD_SETUP_TIME):** Address and data setup time before rise of LCD_E • 0h: 1 clock, 1h: 2 clocks, ..., Fh: 16 clocks
11:08	RW	Fh	**Address/Data Hold Time (AD_HOLD_TIME):** Address and data hold time after fall of LCD_E. • 0h: 1 clock, 1h: 2 clocks, ... Fh: 16 clocks
07:03	*RO*	*0*	*Reserved*
02	RW	0	**DMA Enable (DMA):** When set, enables DMA requests for LCD writes.
01	RW	0	**Data Bus Width (DATA_WIDTH):** When set, 8-bits. When cleared, 4-bits.
00	RW	0	**LCD_ Polarity (EN_POL)**: When set, polarity of the LCD_E signal is '1'.

15.5.2 Offset 004h: LCD_RD – LCD Read Register



Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09	RW	0	**Read Command (READ_CMD):** When written to '1', initiates a read cycle. This is cleared automatically once the read cycle is completed. While read as '1', a read is not yet complete.
08	RW	0	**State (LCD_RS):** When set, drive a '1' on LCD_RS pin when the read cycle is executed.
07:00	RO	00h	**LCD Data (LCD_DATA):** Data resulting from the read cycle is left in this register.

15.5.3 Offset 008h: LCD_WR – LCD Write Register

Bits	Type	Reset	Description
31:10	*RO*	*0*	*Reserved*
09	RW	0	**Write Command (WR_CMD):** When written to '1', activates a write command. Cleared automatically on write completion. If both read and write commands are active, the read takes precedence. When read as '1', a write command is pending.
08	RW	0	**LCD_RS State (LCD_RS):** When cleared, drive logic 0 on LCD_RS pin during write. When set, drive logic 1 on LCD_RS pin during write.
07:00	RW	00h	**LCD Data (LCD_DATA):** Data to appear on the LCD_D[7:0] pins during the write cycle.

Chapter 16: Timers

16.1 Watchdog Timer (WDT)

The WDT is an APB device that protects against conditions that may place the microcontroller into an unsuitable operating state. When enabled, the WDT periodically sends interrupts and waits for interrupt service. If software does not respond by servicing the interrupt within a pre-defined time and WDT reset is enabled, the WDT sends a system reset request to the Power Management Unit (PMU). The WDT features sixteen programmable time delay periods, 2^{16} through 2^{31} clock cycles. The architecture for the Watchdog Timer is shown below.

16.1.1 Enabling

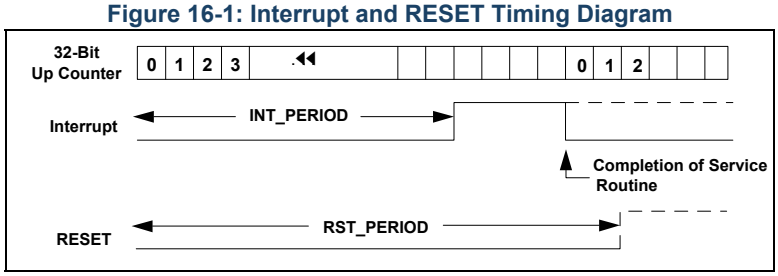
Software sets WDT_CTL.WDT_EN to '1' to enable the WDT. The WDT can be disabled by system reset or clearing WDT_CTL.WDT_EN to '0'. Interrupt and reset from the WDT is controlled through WDT_CTL.INT_EN and WDT_CTL.RST_EN.

16.1.2 Time Delay Period Selection

There are two independent time delay periods can be specified in the WDT, an "interrupt period", which specifies the time delay before the WDT sends an interrupt to the CPU, and a "reset period", which specifies the time delay before the WDT sends a system reset request to the PMU if the interrupt is not serviced.

After the interrupt is generated, software services it with a sequence of writes to WDT_RR. If the interrupt is not serviced within this timeframe and the WDT reset is enabled, the WDT sends a system reset request (RESET) to the PMU. Figure 15-1 shows the timing diagrams for interrupt and reset.

Figure 16-1: Interrupt and RESET Timing Diagram



There are sixteen choices of time delay periods for the WDT—2^{16}, 2^{17}, 2^{18}, through 2^{31} hclk cycles. The time delay for a specific clock source can be calculated by the following equation:

$$Time\ Delay = \frac{Divider\ Value}{Clock\ Source}$$

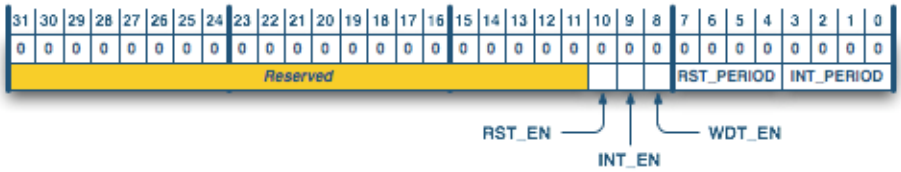
Table 16-1: Watchdog Timer Approximate time Delays

Clock Source	Divider Value	Time Delay
90 MHz hclk	2^{18}	3ms
90 MHz hclk	2^{22}	47ms
90 MHz hclk	2^{25}	0.4s
90 MHz hclk	2^{27}	1.5s

16.1.3 Registers (Base → FFFEC000h)

Offset	Register	Description
000h	WDT_CTL	Watchdog Timer Control Register
004h	WDT_RR	Watchdog Timer Reset Register

16.1.3.1 Offset 000h: WDT_CTL – Watchdog Timer Control

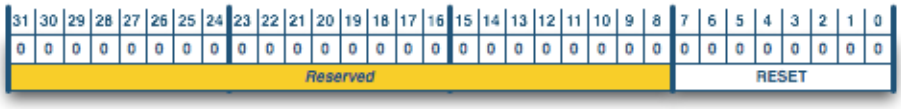


Bits	Type	Reset	Description
31:11	*RO*	*0*	*Reserved*
10	RW	0	**WDT Reset to PMU (RST_EN):** When set, the watchdog timer reset to the PMU is enabled.
09	RW	0	**WDT Interrupt (INT_EN):** When set, the watchdog timer interrupt is enabled.
08	RW	0	**WDT Enable (WDT_EN):** When set, the watchdog timer is enabled.
07:04	RW	0h	**Reset Period (RST_PERIOD):** If the CPU does not service the interrupt within this time period, the WDT sends a system reset request to the PMU. See Table 16-2
03:00	RW	0h	**Interrupt Period (INT_PERIOD):** Time period before sending an interrupt. See Table 16-2

Table 16-2: Time Period Values for INT_PERIOD and RST_PERIOD

Bits	Clocks	Bits	Clocks	Bits	Clocks	Bits	Clocks
Fh	2^{16}	Bh	2^{20}	7h	2^{24}	3h	2^{28}
Eh	2^{17}	Ah	2^{21}	6h	2^{25}	2h	2^{29}
Dh	2^{18}	9h	2^{22}	5h	2^{26}	1h	2^{30}
Ch	2^{19}	8h	2^{23}	4h	2^{27}	0h	2^{31}

16.1.3.2 Offset 004h: WDT_RR – Watchdog Timer Reset



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Reset (RESET):** If an A5h-5Ah sequence is written to WDT_RR before the reset period lapses, the interrupt timer is reset to its initial count value and counting resumes. • A5h: The first write value required to reset the WDT Timer prior to a system reset time-out. • 5Ah: The second write value required to reset the WDT Timer prior to a system reset time-out.

16.2 16-bit PWM Timers (Timers 3 to 0)

There are four 16-bit reloadable timers on APB that can be used for timing, event counting, or generation of pulse-width modulated (PWM) signals. The timers' features include:

- 16-bit reload counter
- Programmable pre-scalar with pre-scale values from 1 to 4096
- PWM output generation
- Capture, compare and capture/compare capability
- External input pin for timer input, clock gating, or capture signal. External input pin signal frequency is limited to a maximum of 1/4th the hclk frequency.
- Timer output pin
- Timer interrupt

16.2.1 Operation

The timers are 16-bit up-counter timers. Minimum time-out delay is set by loading the value 0001h into TxR and setting the TxCTL.PRES to "1". Maximum time-out delay is set by loading the value 0000h into TxR, and setting TxCTL.PRES to "4096". If the Timer reaches FFFFh, the timer rolls over to 0000h and continues counting. Operating Modes (TxCTL.TMODE)

16.2.1.1 One-Shot Mode

In this mode, the timer counts from Tx to TxR. The timer input is hclk. Upon reaching TxR, the timer generates an interrupt, and the count value in Tx is reset to 0001h. The timer is disabled and stops counting. The steps for configuring a timer for this mode are:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to "000", and set TxCTL.PRES. If using the timer output function, set the initial output level via TPOL:
2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = "1").

The timer period is given by the following equation:

$$\text{Oneshot Mode Timeout Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

16.2.1.2 Continuous Mode

In this mode, the timer counts from Tx to TxR. The timer input is hclk. Upon reaching TxR, the timer generates an interrupt, the Tx is reset to 0001h and counting resumes. The steps for configuring a timer for this mode are:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to "001", and set TxCTL.PRES. If using the timer output function, set the initial output level via TPOL.
2. Write to the Timer register to set the starting count value. This only affects the first pass in Continuous mode. After the first Timer Compare in Continuous mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = "1").

The timer period is given by the following equation:

$$\text{Continuous Mode Timeout Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an value other than 0001h is loaded into Tx, use the one-shot mode equation to determine the first time-out period.

16.2.1.3 Counter Mode

In Counter mode, the timer counts input transitions from a GPIO port pin. The timer input is taken from the GPIO Port pin Timer Input function. The TPOL bit in the Timer Control Register selects whether the count occurs on the rising edge (TPOL = "0") or the falling edge (TPOL = "1") of the Timer Input signal. In Counter mode, the pre-scalar is disabled. Any value assigned to PRES in Counter Mode will have no effect. The input frequency of the Timer Input signal must not exceed one-fourth the hclk frequency.

Upon reaching the Compare value stored in the Timer Compare register, the timer generates an interrupt, the count value in the Timer register is reset to 0001H and counting resumes. Also, if the Timer Output function is enabled in the GPIO, the Timer Output pin changes state from Low to High or from High to Low at Timer Compare.

The steps for configuring a timer for Counter mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to "010", and set TxCTL.PRES. Select either the rising or falling edge of the timer input signal for the count via TPOL:
2. Write to the Timer register to set the starting count value. This only affects the first pass in Counter mode. After the first Timer Compare in Counter mode, counting always begins at the reset value of 0001H. Generally, in Counter mode, the Timer register should be written with the value 0001H.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. See chapter covering GPIO.
7. Write to the Timer Control register to enable the timer (TEN = "1").

In Counter mode, the number of Timer Input transitions since the timer start is given by the following equation:

$$\text{Counter Mode Timer Input Transitions} = \text{Current Count Value} - \text{Start Value}$$

16.2.1.4 PWM Mode

In PWM mode, the timer outputs a Pulse-Width Modulator (PWM) output signal through a GPIO Port pin. The timer input is the hclk. The timer first counts up to the 16-bit PWM match value stored in the Timer PWM register. When the timer count value matches the PWM value, the timer output toggles. The timer continues counting until it reaches the Compare value stored in the Timer Compare register. Upon reaching the Compare value, the Timer Output signal toggles again, the timer generates an interrupt, the count value in the Timer register is reset to 0001H and counting resumes.

- **TxCTL.TPOL = "0":** Timer Output signal begins as a Low ("0") and then transitions to a High ("1") when the timer value matches the PWM value. The Timer Output signal returns to a Low ("0") after the timer reaches the Compare value, which is reset to 0001H.
- **TxCTL.TPOL = "1":** Timer Output signal begins as a High ("1") and then transitions to a Low ("0") when the timer value matches the PWM value. The Timer Output signal returns to a High ("1") after the timer reaches the Compare value, which is reset to 0001H.

The steps for configuring a timer for PWM mode and initiating the PWM operation are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to "010", and set TxCTL.PRES. Select the initial logic level and PWM high/low transition for the timer output function via TPOL:
2. Write to the Timer register to set the starting count value (typically 0001H). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of 0001H.

3. Write to the PWM register to set the Match value.
4. Write to the Timer Compare register to set the Compare value. The Compare value must be greater than the PWM value.
5. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
6. Configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)
7. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

The PWM period is given by the following equation:

$$PWM\ Period\ (s) = \frac{Reload\ Value \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

If an initial starting value other than 0001H is loaded into the Timer register, the One-Shot mode equation must be used to determine the first PWM time-out period.

If TPOL is set to “0”, the ratio of the PWM output High time to the total period is given by:

$$PWM\ Output\ High\ Time\ Ratio\ (\%) = \frac{Reload\ Value - PWM\ Value}{Reload\ Value} \times 100$$

If TPOL is set to “1”, the ratio of the PWM output High time to the total period is given by:

$$PWM\ Output\ High\ Time\ Ratio\ (\%) = \frac{PWM\ Value}{Reload\ Value} \times 100$$

16.2.1.5 Capture Mode

In Capture mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture count value is written to the Timer PWM register. The timer input is the hclk. The TPOL bit in the Timer Control register determines if the Capture occurs on a rising edge (TPOL = “0”) or a falling edge (TPOL = “1”) of the Timer Input signal. When the Capture event occurs, an interrupt is generated, and the timer continues counting.

The timer continues counting up to the 16-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt and counting continues.

The steps for configuring a timer for Capture mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “100”, and set TxCTL.PRES. Select the capture edge for the timer input via TPOL
2. Write to the Timer register to set the starting count value (typically 0001H).
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Capture mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$Capture\ Elapsed\ Time(s) = \frac{(Capture\ Value - Start\ Value) \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

16.2.1.6 Compare Mode

In Compare mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Compare register. The timer input is the hclk. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) upon Compare. When the Timer reaches FFFFh, the timer rolls over to 0000H and continues counting.

The steps for configuring a timer for Compare mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “101”, and set TxCTL.PRES. If using the timer output function, set the initial output level via TPOL:
2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)
6. Write to the Timer Control register to enable the timer and initiate counting (TEN = “1”).

In Compare mode, the hclk always provides the timer input. The Compare time is given by the following equation:

$$Compre\ Mode\ Time(s) = \frac{(Compare\ Value - Start\ Value) \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

16.2.1.7 Gated Mode

In Gated mode, the timer counts only when the Timer Input signal is in its active state (asserted), as determined by the TPOL bit in the Timer Control register. When the Timer Input signal is asserted, counting begins. A timer interrupt is generated when the Timer Input signal is de-asserted or a Timer Compare occurs.

The timer counts up to the 16-bit Compare value stored in the Timer Compare register. The timer input is the hclk. When reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0001H and counting resumes (assuming the Timer Input signal is still asserted). Also, if the Timer Output function is enabled (in the GPIO), the Timer Output pin changes state (from Low to High or from High to Low) at Timer Compare.

The steps for configuring a timer for Gated mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “110”, and set TxCTL.PRES:
2. Write to the Timer register to set the starting count value. This only affects the first pass in Gated mode. After the first timer reset in Gated mode, counting always begins at the reset value of 0001H.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.

If using the Timer Output function, configure the associated GPIO port pin for the Timer Output function. (See chapter covering GPIO herein.)

6. Write to the Timer Control register to enable the timer (TEN = “1”).
7. Assert the Timer Input signal to initiate the counting (High if TPOL = “0”; Low if TPOL = “1”).

16.2.1.8 Capture/Compare Mode

In Capture/Compare mode, the timer begins counting *after the first* desired external Timer Input transition occurs. The desired transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control Register. The timer input is the hclk.

Every subsequent desired transition (after the first) of the Timer Input signal captures the current count value. The Captured timer value is written to the Timer PWM register. When the Capture event occurs, an interrupt is generated, the count value in the Timer register is reset to 0001H, and counting resumes.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Compare register. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer register is reset to 0001H, and counting resumes.

The steps for configuring a timer for Capture/Compare mode and initiating the count are as follows:

1. Clear TxCTL.TEN, Write TxCTL.TMODE to “100”, and set TxCTL.PRES. Select the capture edge for the timer input via TPOL
2. Write to the Timer register to set the starting count value (typically 0001H).
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Configure the associated GPIO port pin for the Timer Input function.
6. Write to the Timer Control register to enable the timer (TEN = “1”).
7. Counting begins *after the first* desired transition of the Timer Input signal. No interrupt is generated by this first edge.

In Capture/Compare mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\text{Capture Elapsed Time (s)} = \frac{(\text{Capture Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

16.2.2 Timer Input/Output Polarity Bit Modes (TxCTL.TPOL)

TxCTL.TPOL is a function of TxCTL.TMODE, which can be set for 1 of the following 8 modes:

- **One-Shot Mode:** When the timer is disabled, the Timer Output signal is set to the value of TPOL. When the timer is enabled, the Timer Output signal is complemented (changes state from Low-to-High or High-to-Low) upon Timer Reload.
- **Continuous Mode:** When the timer is disabled, the Timer Output signal is set to the value of TPOL. When the timer is enabled, the Timer Output signal is complemented (changes state from Low-to-High or High-to-Low) upon Timer Reload.
- **Counter Mode:** When the timer is disabled, the Timer Output signal is set to the value of TPOL. When the timer is enabled, the Timer Output signal is complemented (changes state from Low-to-High or High-to-Low) upon Timer Reload.
  - **TPOL = “0”:** Count occurs on the rising edge of the Timer Input signal.
  - **TPOL = “1”:** Count occurs on the falling edge of the Timer Input signal.
- **PWM Mode**
  - **TPOL = “0”:** Timer Output is forced Low (“0”) when the timer is disabled. When enabled, the Timer Output is forced High (“1”) upon PWM count match and forced Low (“0”) upon Reload.
  - **TPOL = “1”:** Timer Output is forced High (“1”) when the timer is disabled. When enabled, the Timer Output is forced Low (“0”) upon PWM count match and forced High (“1”) upon Reload.
- **Capture Mode:**
  - **TPOL = “0”:** Count is captured on the rising edge of the Timer Input signal.
  - **TPOL = “1”:** Count is captured on the falling edge of the Timer Input signal.

- **Compare Mode:** When the timer is disabled, the Timer Output signal is set to the value of TPOL. When the timer is enabled, the Timer Output signal is complemented (changes state from Low-to-High or High-to-Low) upon Compare.
- **Gated Mode:**
  - o **TPOL = "0":** Timer counts when the Timer Input signal is High ("1") and interrupts are generated on the falling edge of the Timer Input or upon Timer Reload.
  - o **TPOL = "1":** Timer counts when the Timer Input signal is Low ("0") and interrupts are generated on the rising edge of the Timer Input or upon Timer Reload.
- **Capture/Compare Mode:**
  - o **TPOL = "0":** Counting begins after the first rising edge of the Timer Input signal. The current count is captured on subsequent rising edges of the Timer Input signal.
  - o **TPOL = "1":** Counting begins after the first falling edge of the Timer Input signal. The current count is captured on subsequent falling edges of the Timer Input signal.

16.2.3 Reading the Timer Count Values

The current count value in the timers can be read while the timer is counting (TEN = "1"). This capability has no effect on timer operation.

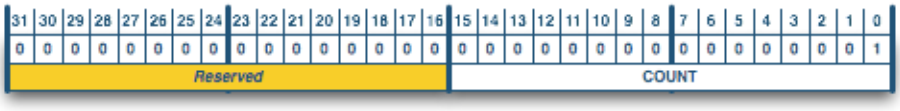
16.2.4 Timer Output Signal Operation

The timer output is toggled every time the counter is reloaded.

16.2.5 Registers (TMR0→FFFE3000h, TMR1→FFFE4000h, TMR2→FFFE5000h, TMR3→FFFE6000h)

Offset	Register	Description
000h	T16_x	Timer Register
004h	T16_x_R	Timer Compare Register
008h	T16_x_PWM	Timer PWM Register
00Ch	T16_x_INT	Timer Interrupt Register
010h	T16_x_CTL	Timer Control Register

16.2.5.1 Offset 000h: T16_x – Timer



Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0001h	**Count (COUNT):** Stores the current 16-bit timer count value. Writing to this register while the timer is enabled is not recommended. If this is written during counting, the 16-bit written value is placed in the counter at the next clock edge. The counter continues counting from the new value.

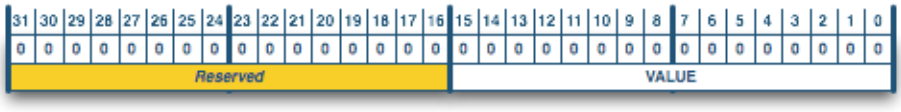
16.2.5.2 Offset 004h: T16_x_R – Timer Compare

Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*

15:00	RW	FFFFh	**Compare Value (VALUE):** Stores the current 16-bit Compare value, which is used to set the maximum count value which initiates a Timer Reload to 0001h.

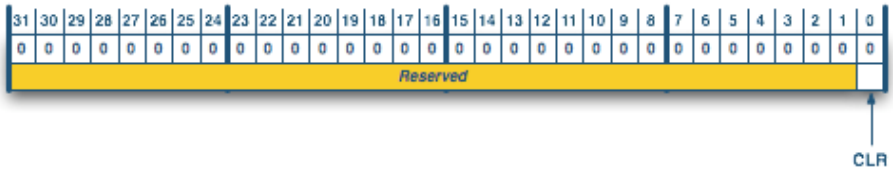
16.2.5.3 Offset 008h: T16_x_PWM – Timer PWM

This forms a 16-bit value that is compared to Tx.COUNT. When a match occurs, the PWM output changes state. The PWM output value is set in TxCTL.TPOL.



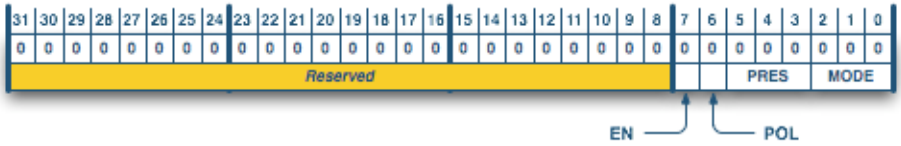
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	0000h	**PWM Value (VALUE):** Stores the 16-bit value that is compared to the current 16-bit timer count.

16.2.5.4 Offset 00Ch: T16_x_INT – Timer Interrupt



Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	RW1C	0	**Clear Interrupt (CLR):** When written to '1', clears the interrupt.

16.2.5.5 Offset 010h: T16_x_CTL – Timer Control



Bits	Type	Reset	Description
31:09	*RO*	*0*	*Reserved*
07	RW	0	**Enable (EN):** When set, the timer is enabled to count.
06	RW	0	**Input/Output Polarity (POL):** This is a function of the current operating mode of the timer. See section 16.2.2.
08 05:03	RW	0h	**Pre-scale Value (PRES):** The timer input clock is divided by 2^{PRES}, as shown below. • 0000: Divide by 1 • 0001: Divide by 2 • 0010: Divide by 4 • 0011: Divide by 8 • 0100: Divide by 16 • 0101: Divide by 32 • 0110: Divide by 64 • 0111: Divide by 128 • 1000: Divide by 256 • 1001: Divide by 512 • 1010: Divide by 1024 • 1011: Divide by 2048 • 1100: Divide by 4096 • 1101 - 1111: undefined

02:00	RW	000	**Mode (MODE):** See encodings below. • 000: One-Shot mode • 001: Continuous mode • 010: Counter mode • 011: PWM mode • 100: Capture mode • 101: Compare mode • 110: Gated mode • 111: Capture/Compare mode

16.3 32-bit Timers (Timers 8 to 4)

There are five 32-bit reloadable timers on the APB interface. The timers' features include:

- 32-bit reload counter
- Programmable pre-scalar with pre-scale values from 1 to 4096
- Cascadable counters
- Timer interrupt

16.3.1 Operation

The timers are 32-bit up-counter. Minimum time-out delay is set by loading the value 00000001h into TxR, and setting the pre-scale value to "1". Maximum time-out delay is set by loading the value 00000000h into TxR, and setting the pre-scale value to "4096". If the timer reaches FFFFFFFFh, it rolls over to 00000000h and continues counting.

16.3.2 Timer Operating Modes

The timers can be configured to operate in one of 4 modes via TxCTL.TMODE.

16.3.2.1 One-Shot Mode

In this mode, the timer counts from Tx to TxR using hclk. Upon reaching TxR, the timer generates an interrupt, the count value in Tx resets to 00000001h and the timer is disabled. The steps for configuring this mode are:

1. Clear TxCTL.TEN, write TxCTL.TMODE to "000", and set TxCTL.PRES.
2. Write to the Timer register to set the starting count value.
3. Write to the Timer Compare register to set the Compare value.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. Write to the Timer Control register to enable the timer and initiate counting (TEN = "1").

The timer period is given by the following equation:

$$\text{One Shot Mode Time Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

16.3.2.2 Continuous Mode

In this mode, the timer counts from Tx to TxR using hclk. Upon reaching TxR, the timer generates an interrupt, Tx is reset to 00000001h, and counting resumes. The steps for configuring this mode are:

1. Clear TxCTL.TEN, write TxCTL.TMODE to "001", and set TxCTL.PRES.
2. Write Tx. This affects the first count. After reaching TxR, the next count begins at 00000001h.
3. Write TxR.
4. If desired, enable the timer interrupt in the Interrupt Controller and set the timer interrupt priority by writing to the appropriate Configuration Table Register in the Interrupt Controller.
5. Write to the Timer Control register to enable the timer and initiate counting (TEN = "1").

The timer period is given by the following equation:

$$\text{Continuous Mode Time Out Period (s)} = \frac{\text{Reload Value} \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

If an initial starting value other than 00000001h is loaded into Tx, use the one-shot mode equation to determine the first time-out period.

16.3.2.3 Counter/Cascade Mode

In this mode, the timer counts time outs on the cascade input that comes from the previous timer. Timers 4 through 8 are sequentially tied from cascade output to cascade input and Timer 8's cascade output ties to Timer 4 cascade input. Any number of timers can be put in the cascade chain and the first timer can be any one of the 5 timers. The first timer in the chain can be set up to be in One-Shot, Continuous or Compare mode to have an output occur for the next cascade input. The rest of the timers in the chain must be in Counter/Cascade mode. In this mode, the pre-scalar is ignored.

For the timers using this mode in the chain, upon reaching TxR, the timer generates an interrupt, the count value in Tx is reset to 00000001h and counting resumes. To use the full 32-bits for the timer, 00000000h should be written to Tx. The steps for configuring a timer for Counter mode and initiating the count are as follows:

1. Clear TxCTL.TEN and write TxCTL.TMODE to "010".
2. Write Tx to set the starting count value. This affects the first pass. After reaching TxR, counting begins at the reset value of 00000001h.
3. Write TxR.
4. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
5. Set TxCTL.TEN.

The number of Cascade Input transitions since the timer start is given by the following equation:

$$Counter\ Mode\ Timer\ Input\ Transitions = Current\ Count\ Value - Start\ Value$$

16.3.2.4 Compare Mode

In this mode, Tx counts to TxR via hclk. Upon reaching TxR, an interrupt is generated, Tx is reset to 00000001h, and counting resumes. When the Timer reaches FFFFFFFFh, the timer rolls to 00000000h and continues counting. The steps for configuring a timer for this mode are:

1. Clear TxCTL.TEN and write TxCTL.TMODE to "101", and write TxCTL.PRES.
2. Write Tx and TxR.
3. If desired, enable the timer interrupt (via the Interrupt Mask Register in the Interrupt Controller) and set the timer interrupt priority (by writing to the appropriate Configuration Table Register in the Interrupt Controller herein).
4. Set TxCTL.TEN

The Compare time is given by the following equation:

$$Compare\ Mode\ Time\ (s) = \frac{(Compare\ Value - Start\ Value) \times Prescale}{System\ Clock\ Frequency\ (Hz)}$$

16.3.3 UART Mode

In UART mode, RxD of UART0, UART1, and UART2 are connected to timer inputs 4,5,6. Timer mode 7 (111b) can be used to measure the ASCII CR character 0x0D (13). The timer should be enabled to begin searching for an auto baud character. The timer enable bit will clear once an auto baud character has been measured.

To measure the auto baud character, the timer reload register should be setup with the maximum character time. This allows the timer to automatically reject break conditions which can be used to reset the serial interface. The timer measures the duration of 8 bit times. It starts measuring at the rising edge ending the start bit to the rising edge beginning the stop bit.

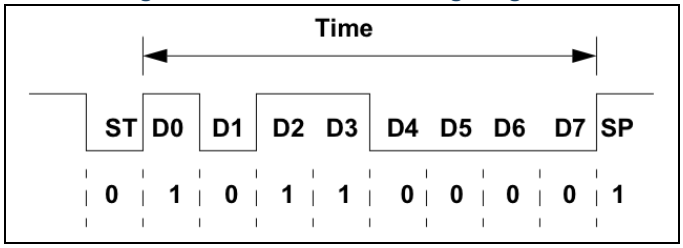
To obtain the baud rate, the timer count should be divided by 8 (shifted right by 3) to get the individual bit time. The bit count then needs to be divided by 16 (shifted right by 4) to obtain the baud rate. Rounding should be done to achieve maximum resolution. Since the timer is preset to 1, the resulting character count is one larger than the actual character time. The baud rate can be calculated as follows:

$$Baud\ Rate = \left(\left((Character\ Count - 1) \gg 6\right) + 1\right) \gg 1$$

Since the auto baud timer automatically rejects baud rates too slow, a break condition can be used as a way to reset the serial line. When software detects a break condition, it can place the timer in auto baud mode. The auto baud search hardware will wait until the RxD pin is idle (high) before beginning an auto baud measurement. Once the auto baud hardware sees the RxD pin go low for the start bit, it starts the counter. If the timer overflows, it assumes a break condition was detected and will reset. The timer will wait for the RxD pin to go high again before searching for another start bit character.

When the auto baud timer sees the rising edge at the end of the start bit, it will reset the timer and begin the character measurement. It will stop counting once it sees the beginning of the stop bit which is two rising edges of the RxD line later. If the counter overflows during this time, it automatically resets and waits for the RxD line to go idle again before searching for another start bit character. If the counter does not overflow, the timer enable bit will clear indicating an auto baud character was measured successfully. Figure 16-2 contains an illustration on auto baud timing.

Figure 16-2: Auto baud Timing Diagram



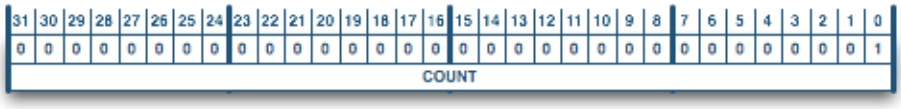
16.3.4 Reading the Timer Count Values

The current count value in the timers can be read while the timer is counting (TEN = “1”). The read has no effect on timer operation.

16.3.5 Registers (Base: TMR4→FFFE7000h, TMR5→FFFE8000h, TMR6→FFFE9000h, TMR7→FFFEA000h, TMR8→FFFEB000h)

Offset	Register	Description
000h	Tx	Timer Register
004h	TxR	Timer Compare Register
00Ch	TxINT	Timer Interrupt Register
010h	TxCTL	Timer Control Register

16.3.5.1 Offset 00h: Tx – Timer

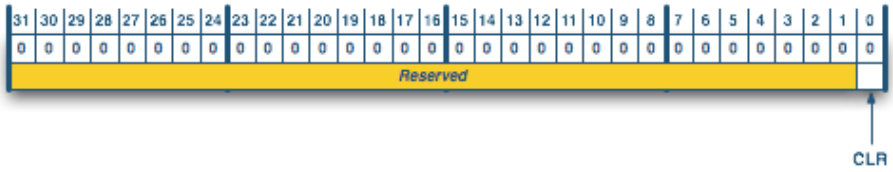


Bits	Type	Reset	Description
31:00	RW	00000000h	**Count (COUNT):** Stores the current 32-bit timer count value. Writing to this while the timer is enabled is not recommended. If written during counting, value is placed in the counter at the next clock edge. The counter continues counting from the new value.

16.3.5.2 Offset 04h: T32_x_R – Timer Compare

Bits	Type	Reset	Description
31:00	RW	FFFFFFFFh	**Compare Value (VALUE):** Stores the current 32-bit Compare value, which is used to set the maximum count value which initiates a Timer Reload to 0001h.

16.3.5.3 Offset 0Ch: T32_x_INT – Timer Interrupt



Bits	Type	Reset	Description
31:01	*RO*	*0*	*Reserved*
00	RW1C	0	**Clear Interrupt (CLR):** Clear the interrupt.

16.3.5.4 Offset 10h: T32_x_CTL – Timer Control

Bits	Type	Reset	Description
31:09	*RO*	*0*	*Reserved*
07	RW	0	**Timer Enable (EN):** When set, the timer is enabled to count.
06	*RO*	*0*	*Reserved*
08 05:03	RW	0h	**Pre-scale Value (PRES):** The timer input clock is divided by 2^{PRES}, as shown below. • 0000: Divide by 1 • 0001: Divide by 2 • 0010: Divide by 4 • 0011: Divide by 8 • 0100: Divide by 16 • 0101: Divide by 32 • 0110: Divide by 64 • 0111: Divide by 128 • 1000: Divide by 256 • 1001: Divide by 512 • 1010: Divide by 1024 • 1011: Divide by 2048 • 1100: Divide by 4096 • 1101 - 1111: undefined
02:00	RW	000	**Timer Mode (MODE):** See encoding below: • 000: One-Shot mode • 001: Continuous mode • 010: Counter mode • 101: Compare mode • 111: UART mode

Chapter 17: Universal Asynchronous Receiver/Transmitter (UART)

The UART is a fully 16550 compatible APB device that implements the logic required to support various asynchronous communications protocols. It implements two separate 16-byte-deep DMA-supported FIFOs for both transmission and reception. The UART module provides the following features:

- 5-, 6-, 7-, or 8-bit data transmission
- Even/odd or no parity bit generation and detection
- Start and stop bit generation and detection (supports up to two stop bits)
- Line break detection and generation
- Receiver overrun and framing errors detection
- Logic and associated I/O to provide modem handshake capability
- Multi-drop mode capable

There are 3 UARTs implemented in the Z32AN Series SoC. UART2 contains an additional infrared encoder/decoder block.

17.1 Functional Description

17.1.1 UART Transmitter

The transmitter block controls the data transmitted on TXD. It implements the FIFO, accessed through UARTx_THR, the transmit shift register, the parity generator, and logic for the transmitter to control the protocol.

Software writes a data byte to be transmitted into UARTx_THR. In FIFO mode, up to 16-bytes can be written at a time. Data from the FIFO is transferred to the transmit shift register at the appropriate time and transmitted out on TXD.

After reset, UARTx_THR is empty, UARTx_LSR.THRE is ‘1’ and an interrupt is generated (if enabled). Software can reset this interrupt by loading data into UARTx_THR.

The transmit shift register places the byte to be transmitted onto TXD serially, least-significant bit first. Control logic within the block adds the protocol bits to the byte being transmitted. The transmitter block obtains the parameters for the protocol from UARTx_LCR. TXD is set to ‘1’ if the transmitter is idle (it does not contain any data to be transmitted). The transmitter operates with the BRG clock. Data bits are placed on TXD once every 16 BRG clock cycles. The transmitter also implements a parity generator that attaches the parity bit to the byte, if programmed.

17.1.2 UART Receiver

The receiver block controls the data reception from RXD. It implements a receiver shift register, receiver line error condition monitoring logic and receiver data ready logic. It also implements the parity checker, and checks for overrun errors and break signals.

Software reads received data from this UARTx_RBR. The condition of UARTx_RBR is monitored via UARTx_LSR.DR, which is set to ‘1’ a data byte is transferred to UARTx_RBR from the receiver shift register. UARTx_LSR.DR is cleared when software reads all received data bytes. If the number of bits received is less than eight, the unused most significant bits of the data byte read are 0. In FIFO mode, up to 16 characters can be received before an overrun condition occurs. In addition to the receive data FIFO, 16 deep FIFOs exist for holding the parity error, framing error, and receive break condition flags.

The receiver uses the clock from the BRG for receiving the data. This clock must be 16x the appropriate baud rate. The receiver synchronizes the shift clock on the falling edge of the start bit. It then receives a complete byte according to the set parameters.

17.1.3 UART Modem Control

The modem control logic provides two outputs and four inputs for handshaking with the modem. Any change in the modem status inputs, except RI, is detected and an interrupt can be generated. For RI, an interrupt is generated only when the trailing edge of the RI is detected. The module also provides LOOP mode for self-diagnostics.

17.1.4 UART Interrupts

17.1.4.1 Transmitter Interrupt

A Transmitter Hold Register Empty interrupt is generated if there is no data available in the transmit holding register. A transmission complete interrupt is generated after the data in the shift register is sent. Both interrupts can be independently disabled using individual interrupt enable bits, or cleared by writing data into the UARTx_THR register.

17.1.4.2 Receiver Interrupts

A receiver interrupt can be generated by three possible sources. The first source, a receiver data ready, indicates that one or more data bytes are received and are ready to be read. This interrupt is generated if the number of bytes in the receiver FIFO is greater than or equal to the trigger level. If the FIFO is not enabled, the interrupt is generated if the receive buffer contains a data byte. This interrupt is cleared by reading the UARTx_RBR.

The second interrupt source is the receiver time-out. A receiver time-out interrupt is generated when there are fewer data bytes in the receiver FIFO than the trigger level and there are no READs and WRITEs to or from the receiver FIFO for four consecutive byte times. When the receiver time-out interrupt is generated, it is cleared only after emptying the entire receive FIFO.

The first two interrupt sources from the receiver (data ready and time-out) share an interrupt enable bit.

The third source of a receiver interrupt is a line status error, indicating an error in byte reception. This error can result from one of the below conditions:

- Incorrect received parity

For multi-drop mode, incorrect parity indicates detection of an address byte.

- Incorrect framing; that is, the stop bit is not detected by receiver at the end of the byte
- Receiver over run condition
- A BREAK condition being detected on the receive data input

An interrupt (due to one of the above) is cleared when the UARTx_LSR register is read.

A line status interrupt is activated (provided this interrupt is enabled) differently depending on whether the FIFO and DMA are enabled or not.

If the FIFO and DMA are not enabled, a line status interrupt from parity error, framing error or break condition is generated immediately when the character is received. The over run error occurs after a second character is received before the first has been read from the UARTx_RBR.

If the FIFO is enabled and DMA is not enabled the line status interrupt is activated when the READ pointer of the receiver FIFO points to the location of the FIFO that contains a byte with an error (parity, framing or break condition). The line status interrupt is activated due to an over run condition when the FIFO is full and another character is received.

The receive character which causes the overrun condition is not written into the FIFO. The interrupt is immediately cleared when the UARTx_LSR register is read. The ERR bit of the UARTx_LSR register is active as long as an erroneous byte due to parity, framing or break condition is present anywhere in the receiver FIFO.

If the FIFO and DMA are enabled, the line status interrupt is activated and the receive DMA request is deactivated when a byte with a parity, framing or break detect error enters the ‘trigger level’ portion of the receive FIFO. The line status interrupt will remain asserted while software alternately reads the UARTx_LSR and UARTx_RBR until the location with the error has been removed from the FIFO.

The specific error bits in the UARTx_LSR will only assert when the READ pointer points to the location of the FIFO that contains the byte with an error. To insure that all bytes with errors have been removed from the trigger level portion of the receive FIFO, software must loop on reading the LSR and RBR registers until the ERR bit is cleared. After the ERR bit is cleared, the DMA request is re-enabled. Any remaining valid data in the receive FIFO will then be handled by the DMA. An alternative is for software to assert the

CLRRXF bit of the UARTx_FCR register which will empty the receive data and error FIFO.

17.1.4.3 Modem Status Interrupt

The modem status interrupt is generated if there is any change in state of the modem status inputs to the UART. This interrupt is cleared when the processor reads UARTx_MSR.

17.2 UART Usage

Following is the standard sequence of events that occur in the Z32AN SOC using the UART:

1. Module Reset. Upon reset, internal registers are at their defaults, and the FIFOs are flushed.
2. Control Transfers
3. Data Transfers

17.2.1 Control Transfers

The data transfer baud rate is determined and the BRG is configured to generate a 16X clock frequency. Interrupts are disabled and the communication control parameters are programmed in UARTx_LCR. FIFO configuration is determined and the receive trigger levels are set in UARTx_FCR. Status registers, UARTx_LSR and UARTx_MSR, are cleared. Interrupts are enabled (except for the transmit interrupt) and the application is ready to use the module for transmission/reception.

17.2.2 Data Transfers

17.2.2.1 Control/Check Modem Status

To control and check modem status, software writes to UARTx_MCR and reading UARTx_MSR.

17.2.2.2 Data Transfers—Transmit with Interrupt

To transmit data, software enables the transmit interrupt. An interrupt is immediately expected in response. Software reads UARTx_IIR and determines that the interrupt occurs due to an empty UARTx_THR register, and writes data bytes to UARTx_THR.

The number of bytes that the application writes depends on whether or not the FIFO is enabled. If the FIFO is enabled, software may write 16 bytes at a time. If not, software can write one byte at a time. As a result of the first write, the interrupt is deactivated. When a new interrupt is generated, software repeats the process until it exhausts all data for transmission.

17.2.2.3 Data Transfers—Receive with Interrupt

The receiver is always enabled, checking for the start bit on RXD. When an interrupt is generated, software reads UARTx_IIR to determine the cause.

If the cause is a line status interrupt, software reads UARTx_LSR and reads the data byte. If the cause is a receive-data-ready condition, software alternately reads UARTx_LSR and UARTx_RBR registers and removes all received data bytes. It reads UARTx_LSR before reading the UARTx_RBR register to determine any errors in the received data.

17.2.3 Poll Mode Transfers

When interrupts are disabled, all data transfers are referred as poll mode transfers. In poll mode transfers, the application must continually poll the UARTx_LSR register to transmit or receive data without enabling the interrupts. The same holds true for the UARTx_MSR register. If the interrupts are not enabled, the data in the UARTx_IIR register cannot be used to determine the cause of interrupt.

17.2.4 DMA mode Transfers

DMA is enabled by setting UARTx_FCR.DMA to '1', disabling the corresponding interrupt(s) and selecting the UART transmit and/or receive devices. The assertion of DMA requests is controlled by the UARTx_FCR.TRIG.

A transmit DMA request is asserted when there are TRIG or fewer bytes in the transmit FIFO. The transmit DMA burst size must be set to less than or equal to 16-TRIG bytes to prevent overrunning the transmit FIFO. A receive DMA request is asserted when there are TRIG or more bytes in the receive FIFO. The receive DMA burst size can be set for up to TRIG bytes.

When receive DMA is enabled, UARTx_IER.LSIE must be set to catch errors that prevent DMA completion.

17.3 Baud Rate Generator (BRG)

The BRG consists of a 16-bit counter, two registers, and associated decoding logic. The initial value of the BRG is defined by UARTx_BRG_H and UARTx_BRG_L. At the rising edge of each hclk, the BRG decrements until 0001h. On the next hclk rising edge, it reloads the initial value from UARTx_BRG_H and UARTx_BRG_L and outputs a pulse to indicate the end-of-count. UART data rate can be calculated as follows:

$$\text{UART Data Rate (bps)} = \frac{\text{hclk Frequency}}{16 \times (\text{UART Baud Rate Generator Divisor})}$$

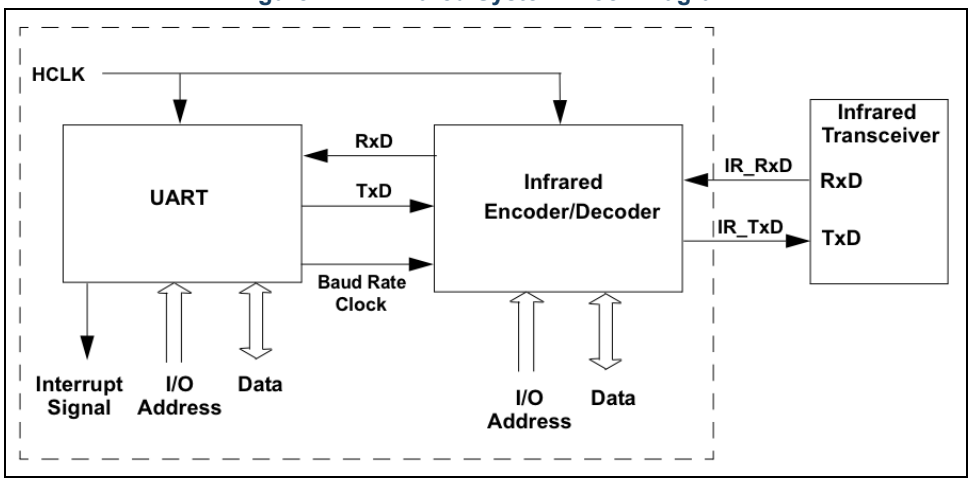
On RESET, the BRG divisor value resets to the smallest allowable value of 0002h. Therefore, the minimum BRG clock divisor ratio is 2. A write to either UARTx_BRG_H or UARTx_BRG_L causes both bytes to load into the BRG counter, restarting the count. The divisor registers can only be accessed if UARTx_LCR.DLAB is set to '1'. Following is the normal sequence of operations to configure the Baud Rate Generator:

1. Set UARTx_LCR[7] to 1 to enable access of the BRG divisor registers
4. Program the UARTx_BRG_L and UARTx_BRG_H registers
5. Clear UARTx_LCR[7] to 0 to disable access of the BRG divisor registers

17.4 Infrared Encoder/Decoder (UART2 only)

UART2 integrates an infrared encoder/decoder (endec) to allow easy communication between the Z32AN Series SoC and an IrDA Physical Layer Specification Version 1.3 compliant device, as shown in Figure 17-1.

Figure 17-1: Infrared System Block Diagram

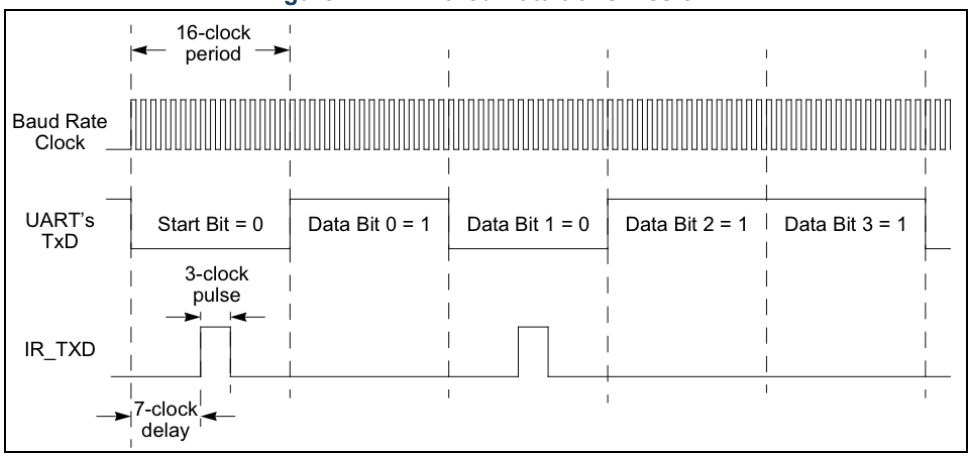


When enabled, transmit data from UART2 is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver. Likewise, data received from the infrared transceiver is decoded by the endec and passed to UART2. Communication is half-duplex meaning that simultaneous data transmission and reception is not allowed. The baud rate is set by the UART BRG and supports baud rates from 9600 bps to 115.2 kbps. Higher baud rates are possible, but do not meet IrDA specifications.

17.4.1 IR Transmit

UART2's TxD and Baud Rate Clock are used by the endec to generate IR_TXD that drives the infrared transceiver. Each UART bit is 16-clocks wide. If the data to be transmitted is a '1', IR_TXD remains '0' for the full 16-clock period. If the data to be transmitted is a logical 0, a 3-clock '1' pulse is output following a 7-clock '0' period, which is then followed by a 6-clock '0' pulse to complete the period. Data transmission is shown below. During data transmission, the IR receive function must be disabled by clearing IR_CTL.IRXEN to '0' to prevent transmitter to receiver cross-talk.

Figure 17-2: Infrared Data transmission

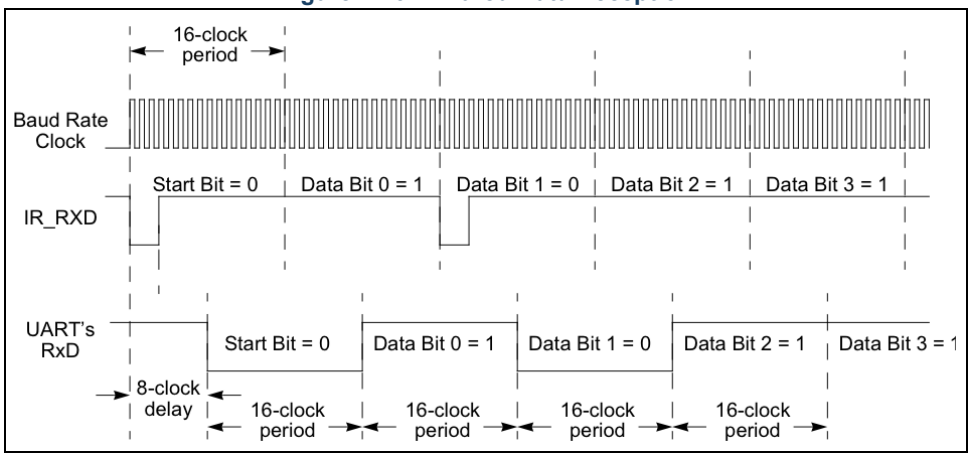


17.4.2 IR Receive

Data received on IR_RXD is decoded by the endec and passed to UART2. IR_CTL.IR_RXEN must be set to enable the receiver decoder. The SIR data format uses half duplex communication therefore the UART must not be allowed to transmit while the receiver decoder is enabled.

UART2 Baud Rate Clock generates the demodulated signal (RxD) that drives the UART. If the data to be received is a '1', IR_RXD remains '1' for the full 16-clock period. If the data to be received is a '0', a 3-clock '0' pulse is output following a 7-clock '1' period, which is then followed by a 6-clock '1' pulse to complete the full period. Data transmission is shown below.

Figure 17-3: Infrared Data Reception



17.4.3 IR Narrow Pulse Detection

The IR endec is designed to ignore pulses on IR_RXD which do not comply with IrDA pulse width specifications. Input pulses wider than 5 baud clocks (that is, 5/16 of a bit period) are always ignored, as this would be a violation of the maximum pulse width specified for any standard baud rate up to 115.2 kbps.

Minimum pulse width checking is optional, as using a slow hclk limits the ability to accurately measure narrow pulse widths near the IrDA specification minimum of 1.41 µs for the 2.4 kbps to 115.2 kbps rate range.

To enable checks of minimum input pulse width on IR_RXD, a non-zero value must be programmed into IR_CTL.NARROW_PULSE. This forms the most significant four bits of the 6-bit down-counter used to determine if an input pulse will be ignored as too narrow. The lower two counter bits are hard-coded to load with '11', resulting in a total down-count equal to ((IR_CTL.NARROW_PULSE * 4) + 3). To be accepted, input pulses must have a width greater than or equal to the down-count value times the hclk period. The following equation can be used to determine an appropriate setting for IR_CTL.NARROW_PULSE:

$$IR\_CTL.NARROW\_PULSE = INT\left[\frac{((F_{SYS} \times W_{MIN}) - 3)}{4}\right]$$

Where F_{SYS} is the frequency of the hclk and W_{MIN} is the minimum width of recognized input pulses. If this equation results in a value less than 1, IR_CTL.NARROW_PULSE must be set to 0h which enables edge detection and ensures that valid pulses wider than W_{MIN} are accepted. The field's maximum setting of Fh supports a W_{MIN} of 1.25 µs when F_{SYS} is 50 MHz

17.4.4 IR Jitter

Due to the inherent sampling of the received IR_RXD signal by the Bit Rate Clock, some jitter is expected on the first bit in any sequence of data. However, all subsequent bits in the received data stream are a fixed 16-clock periods wide.

17.4.5 IR Infrared Encoder/Decoder Signal Pins

The infrared encoder/decoder signal pins (TxD2 and RxD2) are multiplexed with General-Purpose I/O (GPIO) pins. The pins default to GPIO after reset and must be configured to use the IrDA function. For more information, see section Chapter 20:.

17.4.6 IR Loopback Testing

Internal loopback testing is enabled by setting IR_CTL.LOOP_BACK '1'. External loopback testing of the external IrDA transceiver can be accomplished by transmitting data from the UART while IR_CTL.IR_RXEN is set to '1'.

17.5 Registers (Base: UART0→FFFE000h, UART1→FFFE100h, UART2→FFFE200h)

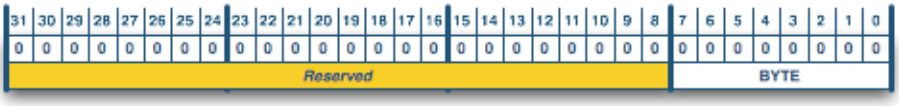
17.5.1 Baud Rate Generator Registers

Offset	Register	Description
000h	UARTx_BRG_L	UART Baud Rate Generator Register—Low Byte
004h	UARTx_BRG_H	UART Baud Rate Generator Register—High Byte

17.5.1.1 Offset 000h: UARTx_BRG_L – UART Baud Rate Generator Low Byte

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Byte (BYTE):** These bits represent the Low-byte of the 16-bit Baud Rate Generator divider value. The complete BRG divisor value is {UARTX_BRG_H, UARTX_BRG_L}.

17.5.1.2 Offset 004h: UARTx_BRG_H – UART Baud Rate Generator High Byte



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Byte (BYTE):** These bits represent the High-byte of the 16-bit Baud Rate Generator divider value. The complete BRG divisor value is {UARTX_BRG_H, UARTX_BRG_L}.

17.5.2 UART Registers

Offset	Register	Description
000h	UARTx_THR	UART Transmit Holding Register
000h	UARTx_RBR	UART Receive Buffer Register
004h	UARTx_IER	UART Interrupt Enable Register
008h	UARTx_IIR	UART Interrupt Identification Register
008h	UARTx_FCR	UART FIFO Control Register
00Ch	UARTx_LCR	UART Line Control Register
010h	UARTx_MCR	UART Modem Control Register
014h	UARTx_LSR	UART Line Status Register
018h	UARTx_MSR	UART Modem Status Register
01Ch	UARTx_SPR	UART Scratch Pad Register
020h	UARTx_IR_CTL	Infrared Control (register exists in UART2 only)

17.5.2.1 Offset 000h: UARTx_THR – UART Transmit Holding Registers

These registers share the same address space as UARTx_RBR and UARTx_BRG_L.

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	WO	00h	**Transmit data byte (TXD):** If less than eight bits are programmed for transmission, the lower bits of the byte written to this register are selected for transmission. The transmit FIFO is mapped at this address. You can write up to 16 bytes for transmission at one time to this address if the FIFO is enabled by the application. If the FIFO is disabled, this buffer is only one byte deep

17.5.2.2 Offset 000h: UARTx_RBR – UART Receive Buffer Registers

This register shares the same address space as UARTx_THR and UARTx_BRG_L.

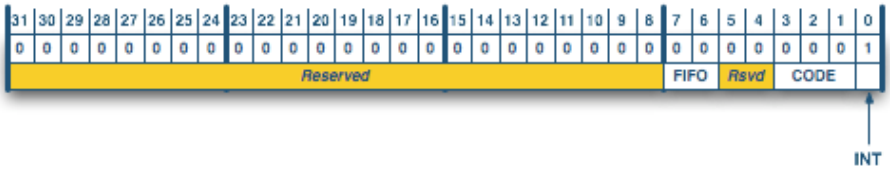
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	WO	00h	**Receive data byte (RXD):**

17.5.2.3 Offset 004h: UARTx_IER – UART Interrupt Enable Registers

This register shares the same addresses as UARTx_BRG_H.

Bits	Type	Reset	Description
31:05	*RO*	*0*	*Reserved*
04	RW	0	**Transmit Complete (TCIE):** When set, transmission complete interrupt is generated when both the transmit hold register and the transmit shift register are empty.
03	RW	0	**Modem Edge Detect (MIIE):** When set, an interrupt on edge detect of status inputs is enabled.
02	RW	0	**Line Status (LSIE):** When set, line status interrupt is enabled for receive data errors: incorrect parity bit received, framing error, overrun error, or break detection.
01	RW	0	**Transmit (TIE):** When set, an interrupt is generated when the transmit FIFO/buffer is empty indicating no more bytes available for transmission.
00	RW	0	**Receive (RIE):** When set, an interrupt is generated if the FIFO/buffer contains data ready to be read or if the receiver times out.

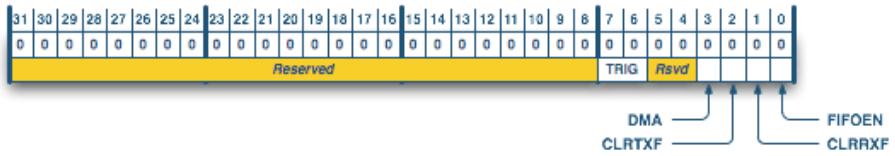
17.5.2.4 Offset 008h: UARTx_IIR – UART Interrupt Identification Registers



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:06	RO	0	FSTS • 00 = FIFO is disabled. • 10 = Receive FIFO is disabled (Multi-drop Mode). • 11 = FIFO is enabled.
05:04	*RO*	*0*	*Reserved*
03:01	RO	0	**Interrupt Status Code (INTSTS):** The code indicated in these three bits is valid only if INTBIT is 0. If two internal interrupt sources are active and their respective enable bits are High, only the higher priority interrupt is seen by the application. The lower-priority interrupt code is indicated only after the higher-priority interrupt is serviced.
00	RO	1	**Interrupt Bit (INTBIT):** 0 = There is an active interrupt source within the UART. 1 = There is not an active interrupt source within the UART.

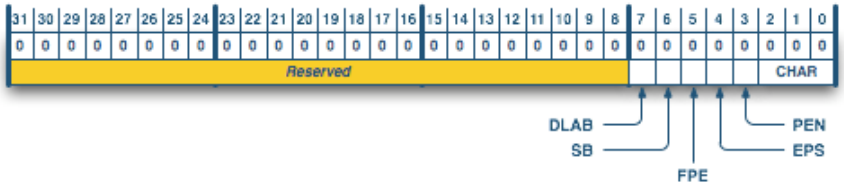
Bits	Priority	Interrupt Type
011	Highest	Receiver Line Status
010	Second	Receive Data Ready or Trigger Level
110	Third	Character Time-out
101	Fourth	Transmission Complete
001	Fifth	Transmit Buffer Empty
000	Lowest	Modem Status

17.5.2.5 Offset 008h: UARTx_FCR – UART FIFO Control Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:06	WO	00	**Trigger Level (TRIG):** See below: • 00 = Receive data interrupt/DMA request is generated when there is 1 byte in the receive FIFO. Transmit DMA request is generated when there is 1 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 01 = Receive data interrupt/DMA request is generated when there are 4 bytes in the FIFO. Transmit DMA request is generated when there are 4 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 10 = Receive data interrupt/DMA request is generated when there are 8 bytes in the FIFO. Transmit DMA request is generated when there are 8 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled. • 11 = Receive data interrupt/DMA request is generated when there are 14 bytes in the FIFO. Transmit DMA request is generated when there are 14 or fewer bytes in the transmit FIFO. Valid only if FIFO is enabled.
05:04	*RO*	*0*	*Reserved*
03	WO	0	**DMA Enable (DMA):** When set, DMA is enabled.
02	WO	0	**Clear Transmit FIFO (CLRTXF):** Writes of '1' clear the transmit FIFO and resets the transmit FIFO pointers. Valid only if the FIFOEN is set. Writes of '0' have no effect.
01	WO	0	**Clear Receive FIFO (CLRRXF):** Writes of '1' clear the receive FIFO, clear the receive error FIFO, and resets the receive FIFO pointers. Valid only if FIFOEN is set. Writes of '0' have no effect.
00	WO	0	**FIFO Enable (FIFOEN):** 0 = Transmit and receive FIFOs are disabled. Transmit and receive buffers are only 1 byte deep. 1 = Transmit and receive FIFOs are enabled. The receive FIFO will not be enabled during Multi-drop Mode (UARTx_MCR.MDM is set).

17.5.2.6 Offset 00Ch: UARTx_LCR – UART Line Control Registers

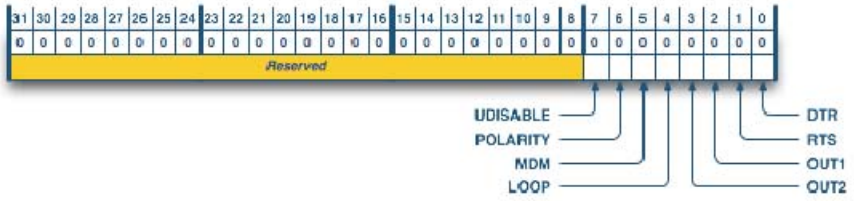


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Baud Rate Generator Access (DLAB):** When set, access to UARTx_BRG_L and UARTx_BRG_H is enabled.
06	RW	0	**Send Break (SB):** When set, UART sends continuous zeroes on the transmit output from the next bit boundary. Transmit data in the transmit shift register is ignored. After writing this bit to '1', TxD becomes '0' after the bit boundary is reached. and the transmit FIFO is cleared. Any new data written to the transmit FIFO during a break must be written only after UARTx_LSR.THRE becomes '1'. New data is transmitted after this bit is cleared for the next BRG edge.
05	RW	0	**Force Parity Error (FPE):** When set, forces a parity error. When set and PEN is set, an incorrect parity bit is transmitted with the data byte.
04	RW	0	**Even Parity Select (EPS):** When cleared, uses odd parity for transmission. When set, uses even parity for transmission. In multi-drop mode:
03	RW	0	**Parity Enable (PEN):** When set, parity bit transmit and receive is enabled for every character.
02:00	RW	000	**UART Character Parameter Selection (CHAR).** See below.

UARTx_MCR.MDM	EPS	Parity Type
0	0	Odd
0	1	Even
1	0	Space
1	1	Mark. EPS resets to '0' after the first character is sent.

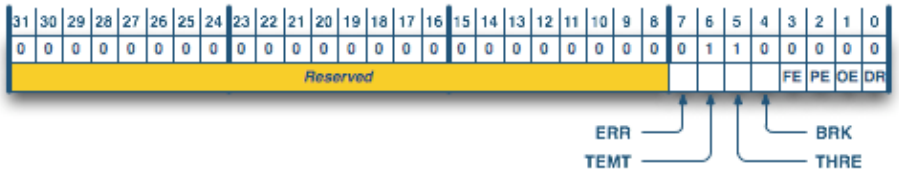
CHAR	Tx/Rx Data Bits	Stop Bits (Tx Stop Bits)
000	5	1
001	6	1
010	7	1
011	8	1
100	5	2
101	6	2
110	7	2
111	8	2

17.5.2.7 Offset 010h – UARTx_MCR – UART Modem Control Registers



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**UART Disable (UDISABLE):** When set, UART FIFOs, state machines and status bits are cleared. Setting this bit prior to updating control register settings ensures corrupted characters will not be sent/received while updating control or baud rate settings.
06	RW	0	**Polarity (POLARITY):** When set, invert polarity of TxD and RxD.
05	RO	0	**Multi-drop Mode (MDM):** When set, multi-drop mode is enabled.
04	RO	0	**Loop Back Mode (LOOP):** When set, Loop Back mode is enabled. Receive data is disconnected connected to internal transmit data. The modem status input ports are disconnected and the four bits of the modem control register are connected as modem status inputs. The two modem control output ports (RTS and DTR) are set to their inactive state.
03	RO	0	**Out 2 (OUT2):** In loop back mode, this bit is connected to UARTx_MSR.DCD.
02	RO	0	**Out 1 (OUT1):** In loop back mode, this bit is connected to UARTx_MSR.RI.
01	RO	0	**Request to Sent (RTS):** In Normal operation, the RTS output port is the inverse of this bit. In LOOP BACK mode, this bit is connected to the CTS bit in the UART Status Register.
00	RO	0	**Data Terminal Ready (DTR):** In Normal operation, the DTR output port is the inverse of this bit. In LOOP BACK mode, this bit is connected to the DSR bit in the UART Status Register.

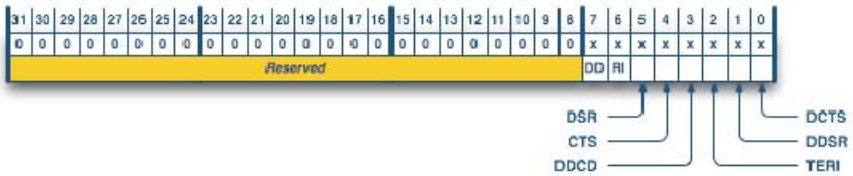
17.5.2.8 Offset 014h: UARTx_LSR – UART Line Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RO	0	**Error (ERR):** When set, an error is detected in the FIFO. There is at least 1 parity, framing or break indication error in the FIFO. Reset when register read and there are no more bytes with error status in the FIFO.
06	RO	1	**Transmitter Empty (TEMT):** When set, UARTx_THR/FIFO and transmit shift register are empty; and the transmitter is idle. This bit cannot be set during the BREAK condition. This bit only becomes 1 after the BREAK command is removed.
05	RO	1	**Transmit Holding Register Empty (THRE):** When set to '1', UARTx_THR/FIFO is empty. This bit cannot be set to 1 during the BREAK condition. This bit only becomes 1 after the BREAK command is removed.
04	RO	0	**Break (BI):** When set the receiver detects a BREAK condition on the receive line. This bit is 1 if the duration of BREAK condition on the receive data is longer than one character transmission time, the time depends on the programming of UARTx_LSR. In case of FIFO only one null character is loaded into the receiver FIFO with the framing error. The framing error is revealed whenever that particular data is read from the receiver FIFO.
03	RO	0	**Framing Error (FE):** When set, top character of the FIFO has a framing error. This bit is set to 1 when the stop bit following the data/parity bit is logic 0.
02	RO	0	**Parity Error (PE):** When set, character at the top of the receive FIFO has a parity error.
01	RO	0	**Overrun Error (OE):** When set, overrun error is detected. If FIFO is not enabled, the data in UARTx_RBR was not read before the next character was transferred into UARTx_RBR. If FIFO is enabled, the FIFO was full when an additional character was received by the receiver shift register. The character in the receiver shift register is not put into the receiver FIFO. Reset when register is read.
00	RO	0	**Data Ready (DR):** If FIFO is not enabled, set to '1' when a character is transferred into UARTx_RBR from the receiver shift register. If FIFO is enabled, set to '1' when a character is received and transferred to the receiver FIFO.

17.5.2.9 Offset 018h: UARTx_MSR – UART Modem Status Register

DDCD, TERI, DDSR, and DCTS are cleared to '0' when this register is read.

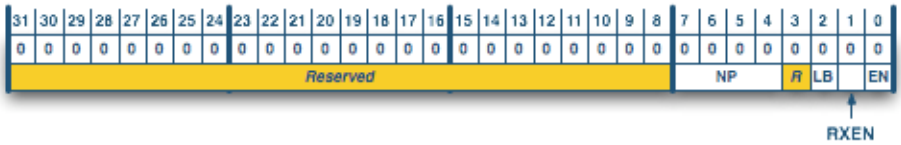


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RO	Undef	**Data Carrier Detect (DD):** In normal mode, this bit reflects the inverted state of the DCDx input pin. In loop back mode, this bit reflects the value of the UARTx_MCR.OUT2.
06	RO	Undef	**Ring Indicator (RI):** In normal mode, this bit reflects the inverted state of the RIx input pin. In loop back mode, this bit reflects the value of the UARTx_MCR.OUT1.
05	RO	Undef	**Data Set Ready (DSR):** In normal mode, this bit reflects the inverted state of the DSRx input pin. In loop back mode, this bit reflects the value of the UARTx_MCR.DTR.
04	RO	Undef	**Clear to Send (CTS):** In normal mode, this bit reflects the inverted state of the CTSx input pin. In loop back mode, this bit reflects the value of the UARTx_MCR.RTS.
03	RO	Undef	**Delta Status Change of DCD (DDCD):** Set to '1' when DCDx pin changes state.
02	RO	Undef	**Trailing Edge Change on RI (TERI):** Set to '1' when a falling edge is detected on the RIx pin.
01	RO	Undef	**Delta Status Change of DSR (DDSR):** Set when the DSRx pin changes state.
00	RO	Undef	**Delta Status Change of CTS (DCTS):** Set when the CTSx pin changes state.

17.5.2.1 Offset 01Ch: UARTx_SPR – UART Scratch Pad Register

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Scratchpad (SPR):** Available for use as a general-purpose scratchpad register.

17.5.2.2 Offset 020h: UARTx_IR_CTL (UART2 only)



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:04	RW	0h	**Narrow Pulse Width Factor (NARROW_PULSE):** Narrow pulse detection value. See 17.4.3.
03	*RO*	*0*	*Reserved*
02	RW	0	**Loop Back Mode (LOOP_BACK):** When set, internal LOOP BACK mode is enabled. IR_TXD is inverted and connected to IR_RXD.
01	RW	0	**IR Receive Enable (IR_RXEN):** When set, IR_RXD data is passed to UART2 RxD. When cleared, IR_RXD data is ignored.
00	RW	0	**IR Enable (IR_EN):** When set, infrared encoder/decoder is enabled.

Chapter 18: Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is an APB device whose synchronous interface allows several devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-fourth the hclk frequency
- Error detection
- Mode collision detection
- Dedicated Baud Rate Generator
- 4 x 16 Transmit and Receive FIFOs
- Transmit and Receive DMA Support

SPI can be configured as either a master (in single or multi-master systems) or slave as shown below.

Figure 18-1: SPI Configured as a Master in a Single Master, Single Slave System

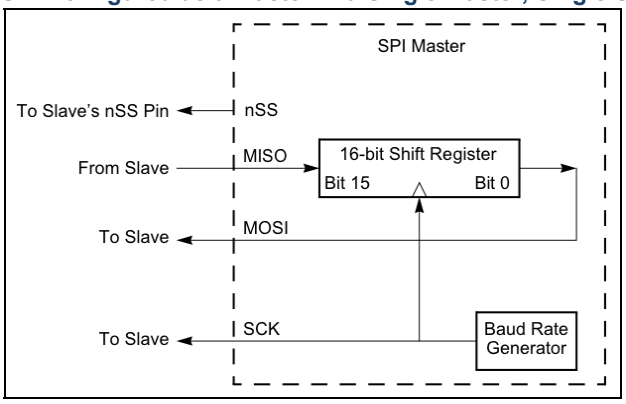


Figure 18-2: SPI Configured as a Master in a Single Master, Multiple Slave system

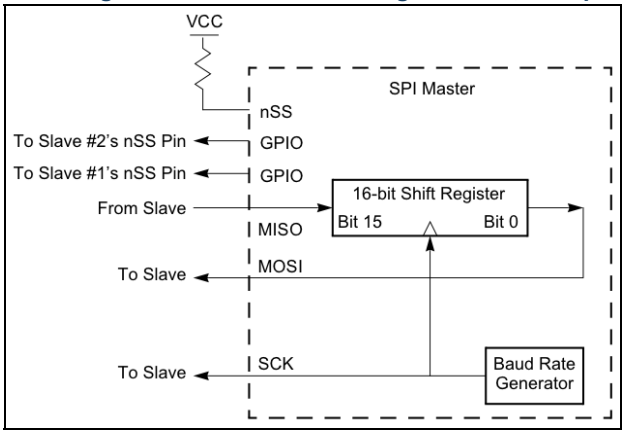
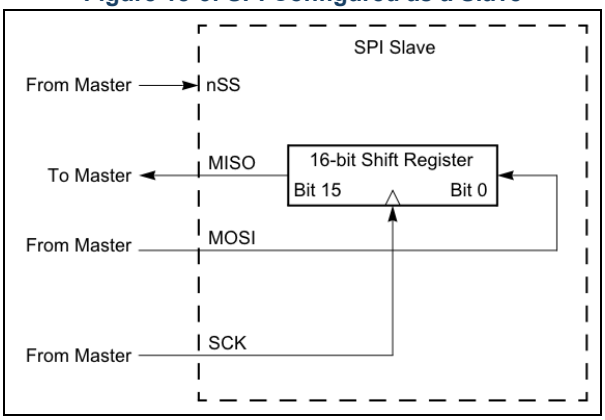


Figure 18-3: SPI Configured as a Slave



18.1 Operation

The SPI is a full-duplex, synchronous, character-oriented channel that supports a four-wire interface (serial clock, transmit, receive, and Slave select). The SPI block consists of a transmit/receive shift register (supported by FIFOs), a Baud Rate (clock) Generator, and a control unit.

During an SPI transfer, data is sent and received simultaneously by both the Master and the Slave SPI devices. Separate signals are required for data and serial clock. When an SPI transfer occurs, a multi-bit (selectable from 1 to 16-bit) character is shifted out on one data pin and a multi-bit character is simultaneously shifted in on a second data pin. A 16-bit shift register in the Master and another 16-bit shift register in the Slave are connected as a circular buffer with the most significant bit (bit 15) sent first.

The SPI contains two 4x16 FIFOs to support the transmit and receive directions. New data is moved automatically from the transmit FIFO into the shift register at the start of every new SPI transfer as long as there is data in the transmit FIFO. At the end of every SPI transfer, the data is then moved from the shift register to the receive FIFO.

18.2 Signals

The SPI signals are reset to GPIO inputs. An external pull-up resistor must be used to prevent floating input signals especially for the clock and slave select.

18.2.1 Master-In/Slave-Out (MISO)

This pin is configured as input of the master and output of the slave. It is one of the two lines that transfer serial data, with the most significant bit sent first. The MISO pin of a slave device is placed in a high-impedance state if the Slave is not selected. When SPI is not enabled, this signal is in a high-impedance state.

18.2.2 Master-Out/Slave-In (MOSI)

This pin is configured as an output in of the master and an input of the slave. It is one of the two lines that transfer serial data, with the most significant bit sent first. When SPI is not enabled, this signal is in a high-impedance state.

18.2.3 Serial Clock (SCK)

This is used to synchronize data through MOSI and MISO. In master mode, the SPI's Baud Rate Generator creates the serial clock. The Master drives the serial clock out its own SCK pin to the Slave's SCK pin. When the SPI is configured as a Slave, the SCK pin is an input and the clock signal from the Master synchronizes the data transfer between the Master and Slave devices. Slave devices ignore the SCK signal, unless the nSS pin is asserted. When configured as a Slave, the SPI block requires a minimum SCK period of greater than or equal to 8 times the hclk period.

The Master and Slave are each capable of exchanging a character of data during a sequence of NUMBITS clock cycles (SPI_MOD.NUMBITS). In both master and slave devices, data is shifted on one edge of the

SCK and is sampled on the opposite edge where data is stable. Edge polarity is determined by the SPI phase and polarity control.

18.2.4 Slave Select (nSS)

This active low signal selects a slave device. In a system with multiple slaves, the master provides separate nSS signals to each slave. nSS must be low prior to all data communication to and from the slave low for the full duration of each character transferred. It can stay low during the transfer of multiple characters or may de-assert between each character.

- **Single Master SPI System**: nSS is configured as an output by setting SPI_MOD.SSIO to '1'. For communication between the master and slave devices, nSS controls the nSS input pin on one of the Slave devices via SPI_MOD.SSV. GPIO output pins must be employed to select additional SPI Slave devices when there are multiple Slaves.
- **Slave SPI System**: nSS configured as an input by clearing SPI_MOD.SSIO to '0'.
- **Multi-Master SPI Systems:** nSS pin must be configured as an input by clearing SPI_MOD.SSIO to '0'. Other GPIO output pins are employed to select slave devices. When acting as the master, if nSS goes low indicating another master is selecting this device as a slave, SPI_STAT.COL is set, indicating a collision. The block is switched between master and slave via SPI_CTL.MMEN.

18.3 Clock Phase and Polarity Control

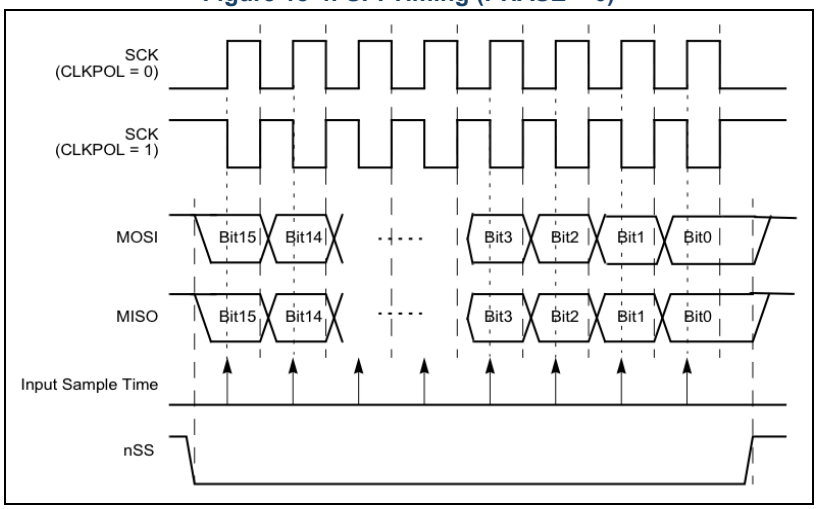
The SPI supports four combinations of clock phase and polarity using SPI_CTL.CLKPOL and SPI_CTL.PHASE, as shown in the table below. For proper data transmission, the clock phase and polarity must be identical for the SPI master and slave. The master always places data on the MOSI line a half-cycle before the clock edge (SCK signal), in order for the slave to latch the data.

PHASE	CLKPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

18.3.1 Transfer Format (SPI_CTL.PHASE = 0)

Figure 18-4 shows the timing diagram for 16-bit transfer in which SPI_CTL.PHASE is cleared. The two SCK waveforms show both polarities of SPI_CTL.CLKPOL. In the case of multi-character transfers with nSS remaining asserted between characters, the output data will change at the end of the Bit0 (final clock edge) to reflect the output value for Bit15 of the next character.

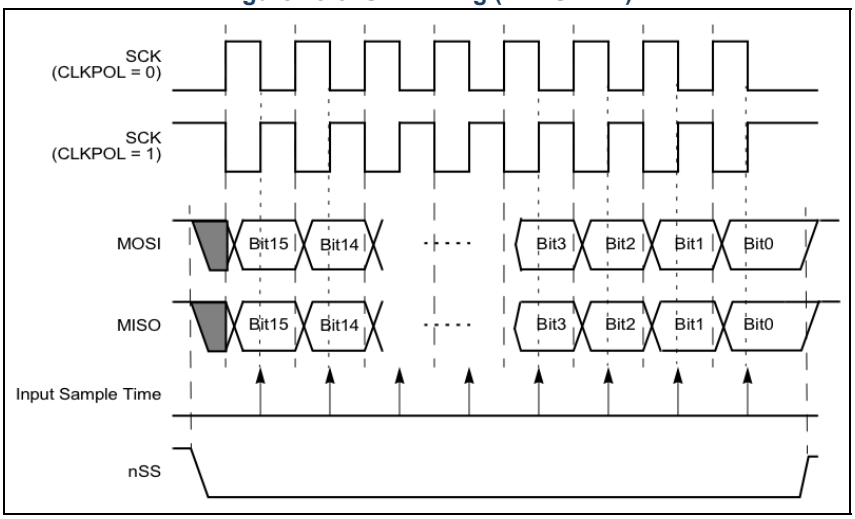
Figure 18-4: SPI Timing (PHASE = 0)



18.3.2 Transfer Format (SPI_CTL.PHASE = 1)

Figure 18-5 shows the timing diagram for an SPI transfer in which SPI_CTL.PHASE is set to 1. Both polarities of SPI_CTL.CLKPOL are shown. In the case of multi-character transfers with nSS remaining asserted between characters, the Bit0 output data will remain stable until the clock edge which starts Bit15 of the next character or until nSS de-asserts at the end of the transfer.

Figure 18-5: SPI Timing (PHASE = 1)



18.4 Multi-Master Operation

In a multi-master SPI system, all SCK pins are tied together, all MOSI pins are tied together, and all MISO pins are tied together. All SPI pins must then be configured in open-drain mode by setting SPI_CTL.WOR to prevent bus contention. At one time, only one SPI device is configured as the Master and all other SPI devices on the bus are configured as Slaves.

The Master enables a single Slave by asserting the nSS pin on that Slave only. Then, the single Master drives data out its SCK and MOSI pins to the Slaves' SCK and MOSI pins (including those which are not enabled). The enabled Slave drives data out its MISO pin to the Master's MISO pin.

For a Master device operating in a multi-master system, if the nSS pin is configured as an input and is driven low by another master, SPI_STAT.COL is set to '1', indicating a multi-master collision (mode fault error condition).

18.5 Slave Operation

The SPI block is configured for Slave mode operation by:

- Setting SPI_CTL.SPIEN to '1'
- Clearing SPI_CTL.MMEN to '0'
- Clearing SPI_MOD.SSIO to '0'

The IRQE, PHASE, CLKPOL, and WOR bits in SPI_CTL and SPI_MOD.NUMBITS must be set to be consistent with the other SPI devices. SPI_CTL.STR can be used to force a startup interrupt. SPI_CTL.BIRQ and SPI_MOD.SSV are not used in slave mode. The SPI BRG is not used in slave mode.

If the slave has data to send to the master, the data must be written to SPI_DAT before the transaction starts (first edge of SCK when nSS is asserted). If SPI_DAT is not written prior to the slave transaction, MISO outputs whatever value was written last into SPI_DAT. Due to the delay resulting from synchronization of the SPI input signals to hclk, the maximum SCK baud rate that can be supported in slave mode is hclk divided by 8. This rate is controlled by the SPI master.

18.6 Error Detection

SPI contains error detection logic to recognize when communication errors have occurred. If SPI_CTL.IRQE is set to ‘1’, one or more of the error conditions asserting generates an interrupt. SPI_STAT indicates which error has been detected. Writing a ‘1’ to these bits in SPI_STAT clears the interrupt condition.

18.6.1 Transmit Overrun

A transmit overrun error indicates that a write to the SPI Data register was attempted when the internal transmit FIFO was full in either Master or Slave modes. An overrun sets SPI_STAT.TOVR to ‘1’.

18.6.2 Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master collision). The mode fault is detected when an enabled Master’s nSS input pin is asserted low. A mode fault sets SPI_STAT.COL to ‘1’.

18.6.3 Slave Mode Abort

In Slave mode of operation, if the nSS pin de-asserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs, the ABT bit is set in the SPI_STA. The next time nSS asserts, the MISO pin outputs SPI_DAT[15], regardless of where the previous transaction left off. Writing 1 to ABT bit clears this error flag.

18.6.4 Receive Overrun

A receive overrun error indicates a write to the receive FIFO occurred when the internal receive FIFO was full (in either Master or Slave modes). An overrun sets the ROVR bit in the SPI Status register to 1. Writing 1 to ROVR bit clears this error flag.

18.7 SPI Interrupts

When SPI_CTL.IRQE is set, SPI generates an interrupt when one of the following interrupt conditions occurs:

- A data interrupt occurs when the transmit character has been fully moved out of the shift register and the Transmit FIFO is empty (in either Master or Slave mode). Since transmit and receive are always interlocked, there is no need for a separate receive interrupt.

  If either SPI_DMA.RXDMA or SPI_DMA.TDMA is set, the data interrupt is not asserted, however error interrupts will still occur. To start the data transfer process, an SPI interrupt can be forced by software writing SPI_CTL.STR to ‘1’.

- If any of error conditions occur, the corresponding error bit and IRQ are set in SPI_STAT and an interrupt is asserted. The error status bits and IRQ must be cleared at the same time by writing 1 to those bits.
- If SPI is disabled, an interrupt can be generated by a Baud Rate Generator time-out. This timer function must be enabled by setting SPI_CTL.BIRQ to ‘1’.

The interrupt condition is indicated in SPI_STAT.IRQ.

18.8 SPI Baud Rate Generator (BRG)

In master mode, the BRG creates a lower frequency serial clock (SCK) for data transmission synchronization between the Master and the external Slave. The input to the Baud Rate Generator is the hclk. The SPI Baud Rate register is a 16-bit reload value, BRG[15:0], for the SPI Baud Rate Generator. The reload value must be greater than or equal to 0002H for SPI operation (maximum baud rate is hclk frequency divided by 4). The SPI baud rate is calculated using the following equation (for the special case BRG = 0x0000 substitute 2**16 for BRG in the equation):

$$\text{SPI Baud Rate (bps)} = \frac{\text{hclk Frequency (Hz)}}{2 \times BRG[15:0]}$$

When the SPI is disabled, the Baud Rate Generator can function as a CONTINUOUS mode 16-bit timer with interrupt on time-out.

Follow the steps below to configure Baud Rate Generator as a timer with interrupt on time-out:

1. Clear SPI_CTL.SPIEN to ‘0’
2. Load the appropriate 16-bit count value into the SPI Baud Rate register, BRG[15:0]
3. Set SPI_CTL.BIRQ to ‘1’

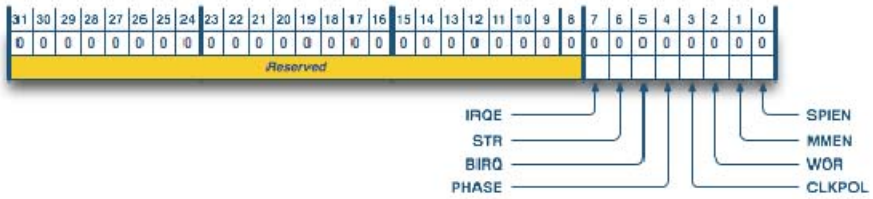
18.9 SPI Registers (Base: SPI0→FFFEE00h, SPI1→FFFEF000h)

Offset	Register	Description
000h	SPI_DAT	SPI Data Register
004h	SPI_CTL	SPI Control Register
008h	SPI_STA	SPI Status Register
00Ch	SPI_MOD	SPI Mode Register
010h	SPI_DIAG	SPI Diagnostic State Register
014h	SPI_BRG	SPI Baud Rate Register
018h	SPI_DMA	SPI DMA Register

18.9.1 Offset 00h: SPI_DAT – SPI Data Register

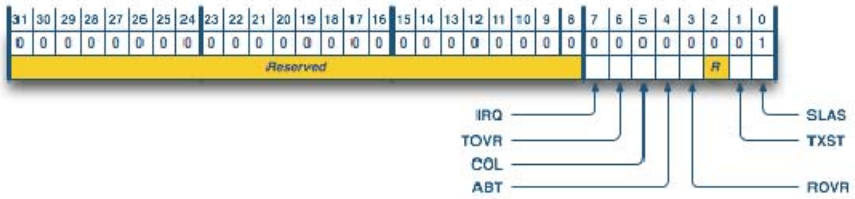
Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	Undef	**Data (DATA):** Stores outgoing (transmit) data and incoming (received) data. With SPI configured as a master, writing data to this register initiates transmission. With SPI configured as a slave, writing data to this register loads the shift register in preparation for the next data transfer with the external master. Data is shifted out starting with bit 15. The last bit received will reside in bit position 0. When the character length is less than 16 bits (as set by SPI_MOD.NUMBITS), the transmit character must be left justified. A received character of less than 16 bits is right justified (last bit received will be in bit position 0).

18.9.2 Offset 04h: SPI_CTL – SPI Control Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Interrupt Request Enable (IRQE):** When cleared, SPI interrupts are disabled. When set, SPI interrupts are enabled. If transmit or receive DMA is enabled, the transmit data complete interrupt is disabled, but other interrupt sources are enabled.
06	RW	0	**Start an SPI Interrupt Request (STR):** When cleared, this value has no effect. When set, sets SPI_STA.IRQ. This bit is cleared by writing 0 to this bit or clearing SPI_STA.IRQ.
05	RW	0	**BRG Timer Interrupt Request (BIRQ):** When cleared, if SPIEN = 0, disables the Baud Rate Generation timer function. If SPIEN = 1, this bit has no effect. When set, if SPIEN = 0, enables the Baud Rate Generation timer function and time-out interrupt. If SPIEN = 1, this bit has no effect.
04	RW	0	**Phase Select (PHASE):** Sets the phase relationship of the data to the clock (see section 18.3).
03	RW	0	**Clock Polarity (CLKPOL):** When set, SCK idles to '1' after character transmission/reception.
02	RW	0	**Wire-OR (Open-Drain) Mode Enable (WOR):** When set, all 4 SPI signal pins (nSS, SCK, MISO, MOSI) configured for open-drain function. This typically used for multi-master/ multi-slave configurations.
01	RW	0	**SPI Master Mode Enable (MMEN):** When set, SPI is configured as a master. When cleared, SPI is configured as a slave.
00	RW	0	**SPI Enable (SPIEN):** When set, SPI operation is enabled.

18.9.3 Offset 08h: SPI_STA – SPI Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW1C	0	**Interrupt Request (IRQ):** When cleared, an interrupt request is not pending. When set, an interrupt request is pending. If one or more of the error condition status bits is asserted, clear those bits at the same time this bit is cleared.
6	RW1C	0	**Transmit Overrun (TOVR):** When set, transmit overrun error has occurred.
5	RW1C	0	**Collision (COL):** When set, a multi-master collision (mode fault) has been detected.
4	RW1C	0	**Slave Mode Transaction Abort (ABT):** When set, slave mode transaction abort detected.
3	RW1C	0	**Receive Overrun (ROVR):** When set, receive overrun error has occurred.
02	*RO*	*0*	*Reserved*
01	RO	0	**Transmit Status (TXST):** When set, data transmission currently in progress.
00	RO	1	**Slave Select (SLAS):** When set, if SPI enabled as a slave, indicates slave is not selected. This bit only has effect when SPI is enabled as a slave.

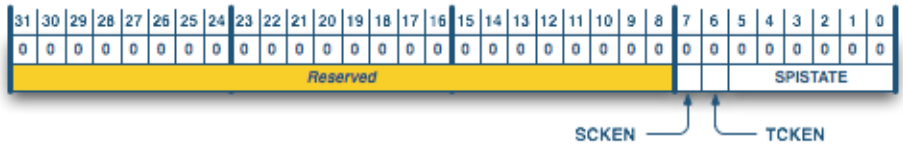
18.9.4 Offset 0Ch: SPI_MOD – SPI Mode Register

Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
06	RW	0	**Diagnostic Mode Control (DIAG):** For manufacturing test mode. Setting this bit allows the Baud Rate Counter value to be read via the SPI_BRG. DIAG = 0 allows reading of SPI_BRG to return the value in the SPI_BRG. When set, reading SPI_BRG will return bits [15:0] of the SPI Baud Rate Counter.
05:02	RW	0h	**Number of Data Bits per Character to Transfer (NUMBITS):** Contains the number of bits to shift for each character transfer. See SPI_DAT for information on valid bit positions when the character length is less than 16-bits.
01	RW	0	**Slave Select I/O (SSIO):** When cleared, nSS pin configured as an input. When set, nSS pin configured as an output (Master Mode Only).
00	RW	0	**Slave Select Value (SSV):** When cleared, if SSIO=1 and SPI is configured as a master, asserts the active slow nSS pin to an external Slave. When set, if SSIO=1 and SPI is configured as a master, de-asserts the active Low nSS pin to an external Slave.

Value	Data Bits per Character	Bits	Data Bits per Character
0000	16	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

18.9.5 Offset 10h: SPI_DIAG – SPI Diagnostic State Register

The register provides observability of internal SPI state.

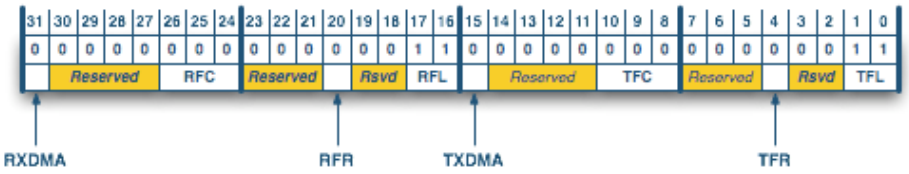


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RO	0	**Shift Clock Enable (SCKEN):** When cleared, internal Shift Clock Enable is de-asserted. When set, internal Shift Clock Enable is asserted (shift register will update on the next hclk).
06	RO	0	**Transmit Clock Enable (TCKEN):** When cleared, internal transmit clock enable is de-asserted. When set, internal transmit clock enable is asserted. When asserted, the serial data out will update on the next hclk (MOSI or MISO).
05:00	RO	000000	**SPI State Machine (SPISTATE):** The current state of the internal SPI State Machine.

18.9.6 Offset 14h: SPI_BRG – SPI Baud Rate Register

Bits	Type	Reset	Description
31:16	*RO*	*0*	*Reserved*
15:00	RW	FFFFh	**Baud Rate Reload Value (BRG):** See section 18.8.

18.9.7 Offset 18h: SPI_DMA – SPI DMA Register



Bits	Type	Reset	Description
31	RW	0	**DMA Receive Enable (RXDMA):** When cleared, disable RX DMA requests. When set, enable RX DMA requests
30:27	*RO*	*0*	*Reserved*
26:24	RO	000	**Receive FIFO Count (RX_FIFO_CNT):** Indicates the number of entries in the RX FIFO. • 000: RxFIFO empty (0 entries) • 001: RxFIFO contains 1 entry • 010: RxFIFO contains 2 entries • 011: RxFIFO contains 3 entries • 100: RxFIFO contains 4 entries
23:21	*RO*	*0*	*Reserved*
20	WO	0	**Clear Receive FIFO (RX_FIFO_CLR):** When written to '1', resets the Rx FIFO. Writes of '0' have no effect. Always reads as '0'.
19:18	*RO*	*0*	*Reserved*
17:16	RW	11	**Receive FIFO Level (RX_FIFO_LEVEL):** Configures the number of filled RxFIFO entries before activating an RxDMA request. • 00: Request RxDMA when RxFIFO contains 1 entry • 01: Request RxDMA when RxFIFO contains 2 entries • 10: Request RxDMA when RxFIFO contains 3 entries • 11: Request RxDMA when RxFIFO contains 4 entries
15	RW	0	**DMA Transmit Enable (TXDMA):** When cleared, disable TX DMA requests. When set, enable TX DMA requests
14:11	*RO*	*0*	*Reserved*
10:08	RO	000	**Transmit FIFO Count (TX_FIFO_CNT):** Indicates the number of entries in the TX FIFO. • 000: TxFIFO empty (0 entries) • 001: TxFIFO contains 1 entry • 010: TxFIFO contains 2 entries • 011: TxFIFO contains 3 entries • 100: TxFIFO contains 4 entries
07:05	*RO*	*0*	*Reserved*
04	WO	0	**Transmit FIFO Clear (TX_FIFO_CLR):** When written to '1', resets the Tx FIFO. Writes of '0' have no effect. Always reads as '0'.
03:02	*RO*	*0*	*Reserved*
01:00	RW	11	**Transmit FIFO Level (TX_FIFO_LEVEL):** Configures the number of free (empty) TxFIFO entries before generating a DMA request. • 00: Request TxDMA when TxFIFO has 1 free entry • 01: Request TxDMA when TxFIFO has 2 free entries • 10: Request TxDMA when TxFIFO has 3 free entries • 11: Request TxDMA when TxFIFO has 4 free entries

Chapter 19: Universal Serial Bus (USB)

The USB controller is an AHB controller that can be configured as either a host or device and supports the On-The-Go protocol. Features:

- Complies with USB specification rev2.0
- Complies with On The Go (OTG) Supplement to the USB2.0 specification
- Supports 12Mbps and 1.5Mbps serial data transmission
- USB2.0 for integration in OTG Dual Role Device applications
- 32-bit DMA engine
- Supports all device classes

19.1 Buffer Descriptor Table

A Buffer Descriptor Table (BDT) in system memory manages USB endpoint communications. It resides on a 512-byte boundary pointed to by the BDT Page Registers. Every endpoint direction has two eight-byte entries, allowing software to process one descriptor while hardware is processing the other descriptor.

Software manages buffers by updating the BDT. A semaphore mechanism is used to distinguish who is allowed to update the BDT. OWN is cleared when the descriptor entry is owned by software, and set when owned by the controller. The descriptor also contains pointers to where the data buffer resides in system memory. The buffer descriptor provides endpoint control information for the controller and software. The controller uses the descriptor to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- Release OWN upon packet completion
- No address increment (FIFO Mode)
- Data toggle synchronization enable
- How much data is to be transmitted or received
- Where the buffer resides in system memory

Software uses the data stored in the BDs to determine:

- Who owns the buffer in system memory
- Data0 or Data1 PID
- The received TOKEN PID
- How much data was transmitted or received
- Where the buffer resides in system memory

The format for the Buffer Descriptor is shown in Figure 19-1.

Figure 19-1: Buffer Descriptor Entry

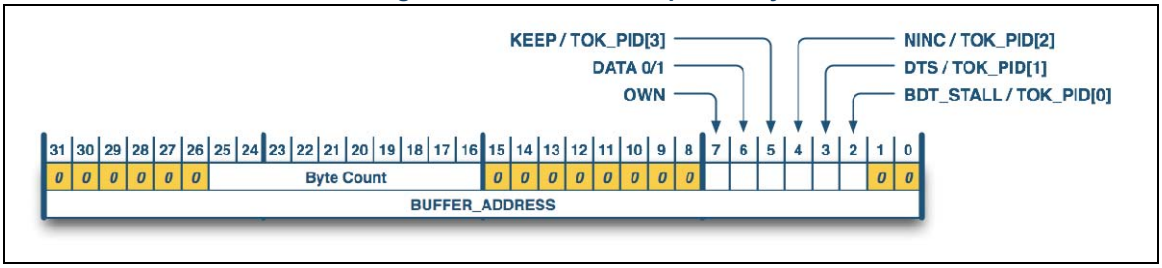


Table 19-1 : Buffer Descriptor - Word 0

Bits	Description
31:26	*000000*
25:16	**BYTE_COUNT:** 10-bit Byte Count. The USB block updates this field upon the completion of a receive with the byte count of the data received.
15:08	*00000000*
07	**OWN:** When set, the controller has exclusive access to the descriptor. When cleared, software has exclusive access to the descriptor. Hardware clears this bit when it completes a token, except when KEEP = '1'.
06	**DATA0_1:** DATA0 (= '0') or a DATA1 (= '1') was transmitted or received. Unchanged by hardware.
05	**KEEP:** When set, once OWN is set the descriptor remains owned by hardware. This bit is typically set for isochronous endpoints that are feeding a FIFO. NINC is normally also set when this bit is set to prevent address increment. If set, this bit is unchanged by the USB block, otherwise bit 3 of the current token PID is written back in to the descriptor by hardware.
04	**NINC:** This bit disables DMA engine address increment. This forces the DMA engine to read or write from the same address. This bit is typically set with KEEP for isochronous endpoints that are interfacing to a FIFO. If KEEP=1 this bit is unchanged by the controller, otherwise bit 2 of the current token PID is written back in to the BD by the controller.
03	**DTS:** When set, enables the controller to perform Data Toggle Synchronization. If KEEP is set this bit is unchanged by the controller, otherwise bit 1 of the current token PID is written back in to the descriptor by the controller.
02	**BDT_STALL:** When set, the controller issues a STALL handshake if a token is received that uses this descriptor. The descriptor is not consumed (OWN remains and the rest of the table is unchanged). If KEEP=1 this bit is unchanged by the controller, otherwise bit 0 of the current token PID is written back in to the descriptor by the controller.
05:02	**TOKEN_PID:** This is written back to the descriptor by the controller when a transfer completes. The values written back are the token PID values from the USB specification: 1h for an OUT, 9h for an IN, or Dh for a SETUP. In host mode this field is used to report the last returned PID or a transfer status indication. The possible values returned are: 3h (DATA0), Bh (DATA1), 2h (ACK), Eh (STALL), Ah (NAK), 0h (Bus Timeout), or Fh (Data Error).
01:00	*00*

Table 19-2 : Buffer Descriptor - Word 1

Bits	Description
31:00	**BUFFER_ADDRESS:** Data buffer address in system memory.

19.2 Receive vs. Transmit

The controller can function as a USB device or host, and may switch modes of operation under software control. The same data paths and buffer descriptors are used for the transmission and reception of data. A USB block centric nomenclature describes the direction of data transfer between the SoC and USB. Rx/receive is used to describe transfers that move data from USB to memory, and Tx/transmit is used to describe transfers that move data from memory to USB.

	Rx	Tx
Device	OUT or Setup	IN
Host	IN	Out or Setup

19.3 Buffer Descriptor Addressing

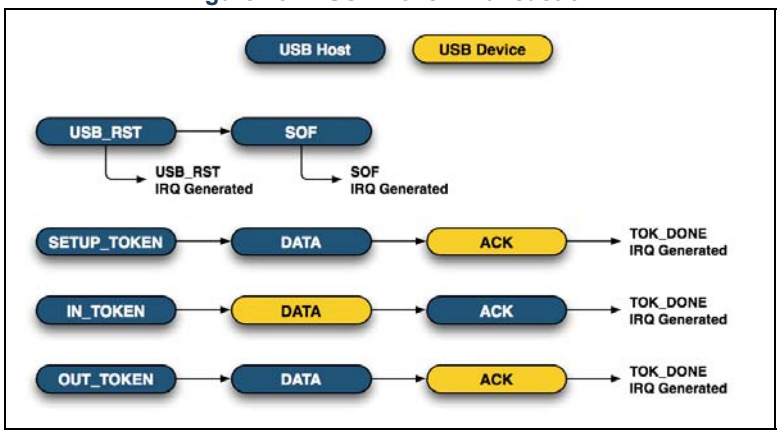
When the controller receives a token on an enabled endpoint it uses its DMA controller to interrogate the BDT. The USB block reads the corresponding endpoint descriptor entry and determines if it owns the descriptor and corresponding buffer in memory. To compute the entry point in the BDT, the USB_BDT_PAGE registers are concatenated with the current endpoint and the TX/ODD fields to form a 32-bit address, shown below:

Bits	Field	Description
31:24	BDT_PAGE	USB_BDT_PAGE3 Register
23:16		USB_BDT_PAGE2 Register
15:09		USB_BDT_PAGE1 Register
08:05	END_POINT	End Point field from the USB TOKEN
04	IN	1 for an TX transmit transfers and 0 for an RX receive transfers
03	ODD	This bit is maintained within the USB block SIE. It corresponds to the buffer that is currently in use. The buffers are used in a ping-pong fashion.
02	Reserved	Set to ‘000’

19.4 USB Transaction

When the controller moves data, it computes the BDT address. Once the BDT has been read, and if OWN = ‘1’, the controller DMAs the data to or from the buffer pointed to by the ADDR field of the descriptor. When the token is complete, the controller updates the descriptor and clears OWN if KEEP is ‘0’. USB_ISTAT.TDONE is set. Figure 19-2 shows how a USB token is processed.

Figure 19-2: USB Token Transaction



The controller has two sources for the DMA overrun: receive FIFO overflow (transient), or the packet received may be larger than the negotiated *MaxPacket* size (software bug). In the FIFO overflow case, the

controller responds with NAK or BTO as appropriate for the class of transaction. USB_ESTAT.DMA is set. Depending on USB_IEN and USB_EEN, an interrupt may be generated. In device mode the BDT is not written back nor is USB_ISTAT.TDONE set. In host mode the USB_ISTAT.TDONE is set and the TOK_PID field of the BDT will be "Fh" to indicate the error.

In the oversized packet case, the controller responds with ACK (for non-isochronous packets). Data written to memory is clipped to *MaxPacket* size, and ERR_STAT.DMA and USB_ISTAT.TDONE are set. The TOK_PID field of the BDT will not be "1111" because the DMA error is not due to latency. The packet length field written back to the BDT will be the *MaxPacket* value to represent the length of the clipped data actually written to memory.

19.5 Host Mode Operation

Setting USB_CTRL.HOST enables host mode. When enabled, only endpoint zero is used. All other endpoints must be disabled by software.

19.5.1 Discover a Connected Device

1. Set USB_CTRL.HOST to ‘1’. Pull down resistors enabled, pull-up disabled. SOF generation begins. SOF counter loaded with 12,000. Eliminate noise on the USB by disabling Start of Frame packet generation by clearing USB_CTRL.USBE to ‘0’.
2. Set USB_IEN.ATTACH to ‘1’, and wait for an attach.
3. Check USB_CTRL.J and USB_CTRL.SE0. If J is ‘0’ the device is low speed.
4. If low speed, set USB_ADDR.LS and USB_EP0_CTRL.NOHUB to ‘1’.
5. Perform a reset (USB_CTRL.RESET to ‘1’, and then ‘0’ after 10ms).
6. Enable SOF to keep device from going to suspend by setting USB_CTL.USB_EN = ‘1’.
7. Start enumeration of device.

19.5.2 Perform a Control Transaction to Device

1. Discover a connected device (see above flow)
2. Set USB_EP0_CTL.TXE, USB_EP0_CTL.RXE, and USB_EP0_CTL.HSHK.
3. Place a copy of the device framework setup command in a memory buffer. See Chapter 9 of the USB 2.0 specification for information on the device framework command set.
4. Initialize current TX EP0 BDT to transfer the 8-byte device framework command (i.e. a GET DEVICE DESCRIPTOR).
5. Set the BDT command word to 0x00080080 – Byte count to 8, own bit to 1
6. Set the BDT buffer address field to the start address of the 8 byte command buffer.
7. Set the address of the device in USB_ADDR. After the bus reset the device address will be zero. It is set to some other value (usually 1) by the Set Address device framework command.
8. Write USB_TOKEN with a SETUP to Endpoint 0 default control pipe (D0h). This initiates a setup token followed by a data packet. The device handshake put in the BDT PID field after the packet completes, and USB_ISTAT.TDONE is set to ‘1’.
9. Set up a buffer in memory for the data.
10. Initialize the current (even or odd) TX EP0 BDT to transfer the data.
11. Set the BDT command word to 0x004000C0 – Byte count to the length of the data buffer in this case 64, own bit to 1, Data toggle to Data1.
12. Set the BDT buffer address field to the start address of the data buffer
13. Write USB_TOKEN with an IN or OUT token to EP0, an IN token for a GET DEVICE DESCRIPTOR command (90h). This initiates an IN token followed by a data packet from the device. When packet completes BDT is written and USB_ISTAT.TOK.DNE is set to ‘1’.
14. To initiate the Status phase of the setup transaction set up a buffer in memory to receive or send the zero length status phase data packet.
15. Initialize the current (even or odd) TX EP0 BDT to transfer the status data.
16. Set the BDT command byte count to 0, OWN to 1, Data toggle to Data0.
17. Set the BDT buffer address field to the start address of the data buffer
18. Write USB_TOKEN with an IN or OUT token to Endpoint 0 the target device default control pipe, an OUT token for a GET DEVICE DESCRIPTOR command (10h). This sends an OUT token followed by a zero length data packet from the host. When packet completes the BDT is written with the handshake form the device and USB_ISTAT.TDONE is set to ‘1’.

19.5.3 Send a Full Speed Bulk Data to Target Device

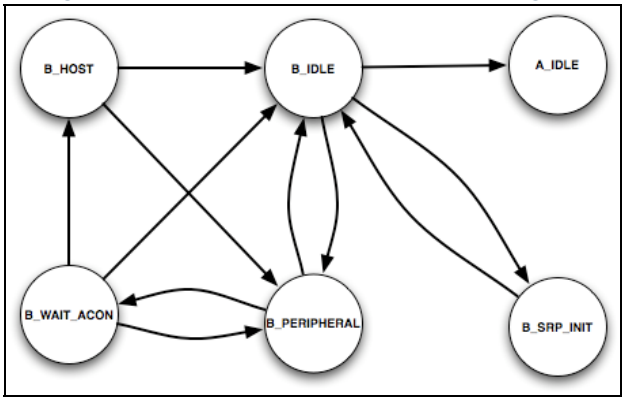
1. Complete all steps discover to connect and configure a device. Write USB_ADDR with the address of the target device.
2. Write the ENDPT0 to 0x1D to enable transmit and receive transfers with handshaking enabled.
3. Setup the Even TX EP0 BDT to transfer up to 64 bytes.
4. Set the device address of the target device in USB_ADDR.
5. Write USB_TOKEN with an OUT token to the desired endpoint. The write to this register will trigger the USB block transmit state machines to begin transmitting the TOKEN and the data.
6. Setup the Odd TX EP0 BDT to transfer up to 64 bytes.
7. Write USB_TOKEN with an OUT.
8. Wait for USB_ISTAT.TDONE to be set, indicating the transfer has completed. If the target NAKed, the controller retries indefinitely without unless USB_EP0_CTRL.RETRY_DIS is set. If set, the handshake (ACK, NAK, STALL or ERROR) is returned in the BDT PID field. If a STALL, the packet must be de-queued and the error condition in the device cleared. If a RESET occurs the target has detached.
9. Once USB_ISTAT.TDONE set, the BDTs can be examined and the next packet queued.

19.6 On-The-Go operation

19.6.1 OTG Dual Role “B” Device Operation

A device is considered a “B” device if it connected to the bus with a USB Type-B or Mini-B cable.

Figure 19-3: Dual Role "B" Device Flow Diagram

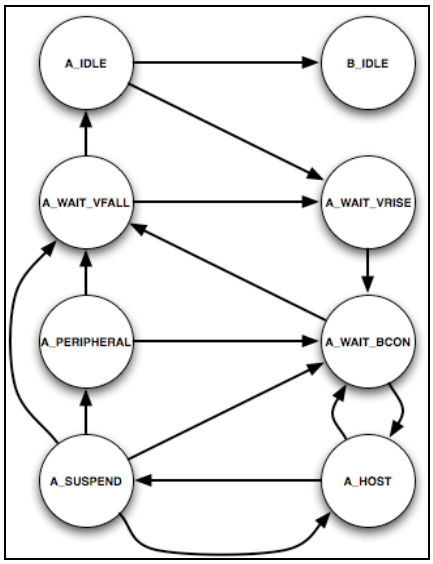


State	Action
B_IDLE	If ID Interrupt, type A cable plugged in. Goto A_IDLE
	If BSESSV Interrupt, “A” device has turned on VBUS and will begin a session, turn on DPH and goto B_PERIPHERAL
	If “B” application wants the bus and Bus is Idle for 2ms and BSESSE set, “B” device can perform an SRP, Pulse CHRG_VBUS Pulse DPH 5-10 ms and goto B_SRP_INIT
B_SRP_INIT	If ID Interrupt or SRP Done (SRP must be done in less than 100 ms), goto B_IDLE
B_PERIPHERAL	If HNP enabled and the bus is suspended and “B” wants the bus, the “B” device can become the host, turn off DPH and goto WAIT_ACON
B_WAIT_ACON	If “A” connects, that is, an attach interrupt is received, turn on host mode, and goto B_HOST
	If ID Interrupt or BSESSV Interrupt, if the cable changes or if VBUS goes away, host doesn’t support us, goto B_IDLE
	If 3.125 ms expires or if a Resume occurs, goto B_PERIPHERAL
B_HOST	If ID Interrupt or BSESSV Interrupt. if cable changes or V_{BUS} goes away, host doesn’t support us, goto B_IDLE.
	If “B” application is done or “A” disconnects, goto B_PERIPHERAL

19.6.2 OTG Dual Role “A” Device Operation

A dual role “A” device will operate as the following flow diagram and state description table illustrates.

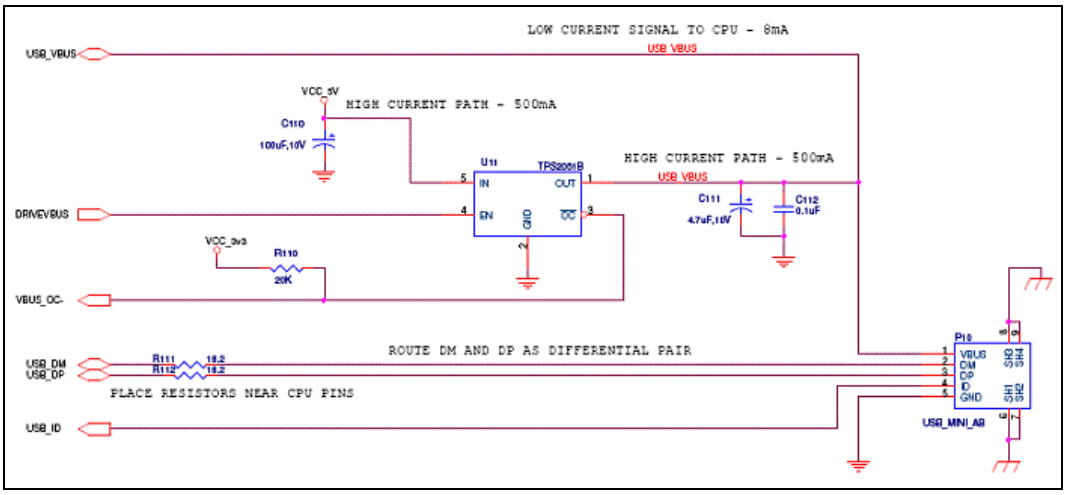
Figure 19-4: Dual Role "A" Device Flow Diagram



State	Action
A_IDLE	If ID Interrupt, cable was unplugged or type B cable attached. Goto B_IDLE
	If the “A” application wants to use the bus or if the “B” device is doing an SRP as indicated by either an A_SESSV Interrupt or Attach or Port Status Change Interrupt check data line for 5 –10 ms pulsing, Go to A_WAIT_VRISE, turn on DRV_VBUS
A_WAIT_VRISE	If ID Interrupt or if AVBUSV is false after 100 ms, cable has been changed or the “A” device cannot support the current required from the “B” device, Go to A_WAIT_VFALL, turn off DRV_VBUS
	If AVBUSV, interrupt, Go to A_WAIT_FALL, turn off DRV_VBUS
A_WAIT_BCON	200 ms without Attach or ID Interrupt, Go to A_WAIT_FALL, turn off DRV_VBUS
	AVBUSV Interrupt and “B” device attaches, Go to A_HOST, turn on Host Mode
A_HOST	Enumerate Device determine OTG Support
	If AVBUSV/ Interrupt or “A” device is done and doesn’t think he wants to do something soon or the “B” device disconnects, Go to A_WAIT_VFALL. Turn off Host Mode and DRV_BUS
	If the “A” device is finished with session or it wants to allow “B” device to take bus, Goto A_SUSPEND
	ID Interrupt or the “B” device disconnects, Goto A_WAIT_BCON
A_SUSPEND	If ID Interrupt, or if 150 ms “B” disconnect timeout (This timeout value could be longer.) or if AVBUSV\ Interrupt, Goto A_WAIT_VFALL, urn off DRV_VBUS
	If HNP enabled, and “B” disconnects in 150 ms then “B” device is becoming the host, Goto A_PERIPHERAL, urn off Host Mode
	If “A” wants to start another session, goto A_HOST
A_PERIPHERAL	If ID Interrupt or if AVBUSV interrupt, Goto A_WAIT_VFALL, turn off DRV_VBUS.
	If 3 –200 ms of Bus Idle, Go to A_WAIT_BCON, Turn on Host Mode
A_WAIT_VFALL	If ID Interrupt or (A_SESSV/ & b_conn/), Go to A_IDLE

19.7 External Configuration

The diagram below shows a typical hook up of the Z32AN to a USB connector.



19.8 Registers (Base → FFFBD000h)

Offset	Address	Description
000h	USB_PER_ID	Peripheral ID Register
004h	USB_ID_COMP	Peripheral ID complement Register
008h	USB_REV	Peripheral revision register
00Ch	USB_ADD_INFO	Peripheral additional info register
010h	USB_OTG_ISTAT	OTG Interrupt Status Register
014h	USB_OTG_IEN	OTG Interrupt Control Register
018h	USB_OTG_STAT	OTG Status Register
01Ch	USB_OTG_CTRL	OTG Control register
080h	USB_ISTAT	Interrupt status register
084h	USB_IEN	Interrupt enable register
088h	USB_ESTAT	Error interrupt status register
08Ch	USB_EEN	Error interrupt enable register
090h	USB_STAT	Status register
094h	USB_CTRL	Control register
098h	USB_ADDR	Address register
09Ch	USB_BDT_PAGE1	Buffer Descriptor Table Page Register 1
0A0h	USB_FRAMEL	Frame number low register
0A4h	USB_FRAMEH	Frame number high register
0A8h	USB_TOKEN	Token register
0ACh	USB_SOFT	SOF threshold register
0B0h	USB_BDT_PAGE2	Buffer Descriptor Table Page Register 2
0B4h	USB_BDT_PAGE3	Buffer Descriptor Table Page Register 3
0C0h – 0FCh	USB_ENDPT_CTRL	Endpoint control register

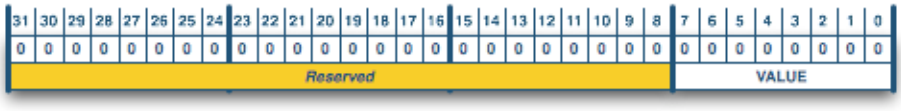
19.8.1 Offset 000h: USB_PER_ID – Peripheral ID Register

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RO	04h	**Configuration number (ID):** Peripheral is the USB block FS/LS USB Core.

19.8.2 Offset 004h: USB_ID_COMP – Peripheral ID Compliment Register

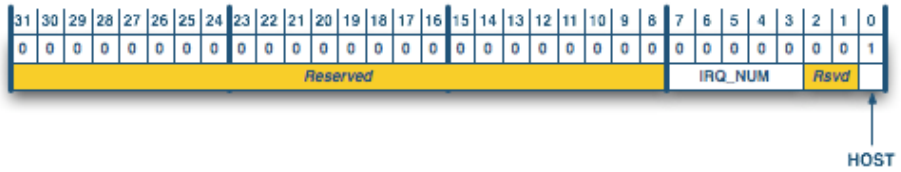
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RO	FBh	**Compliment of Configuration number (NID):** This number is set to 0xFB and indicates that the peripheral is the USB block FS/LS USB Core.

19.8.3 Offset 008h: USB_REV – Peripheral Revision Register



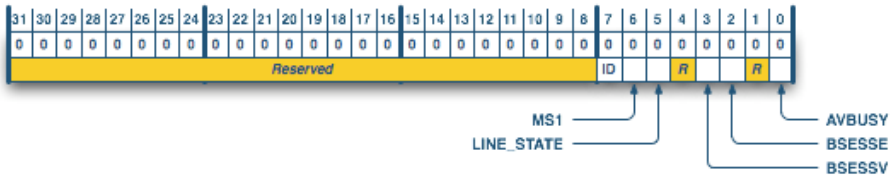
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RO	00h	***Revision (REV):*** Revision number of the USB block USB 2.0 Core

19.8.4 Offset 00Ch: USB_ADD_INFO – Peripheral Additional Info Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:03	RO	00000	**Interrupt Number (IRQ_NUM):** Interrupt Request Number assigned by the tools.
02:01	*RO*	*0*	*Reserved*
00	RO	1	**Host Mode (HOST):** When set, host mode is enabled.

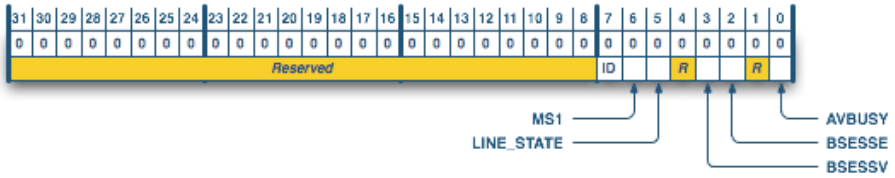
19.8.5 Offset 010h: USB_OTG_ISTAT – OTG Interrupt Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW1C	0	**ID:** This bit is set when a change in the ID Signal from the USB connector is sensed.
06	RW1C	0	**MS1:** This bit is set when the 1 millisecond timer expires.
05	RW1C	0	**LINE_STATE:** Set when USB_CTRL.SE0 and USB_CTRL.J are stable for 1ms, and the value is different from the last time that line state was stable. Set on transitions between SE0 and J, SE0 and K and J and K. Useful for detecting reset, resume, connect and data line pulse signaling.
04	*RO*	*0*	*Reserved*
03	RW1C	0	**BSESSV:** Set when a change in V_{BUS} is detected.
02	RW1C	0	**BSESSE:** Set when a change in VBUS is detected on a “B” device.
01	*RO*	*0*	*Reserved*
00	RW1C	0	**AVBUSV:** This bit is set when a change in VBUS is detected on an “A” device.

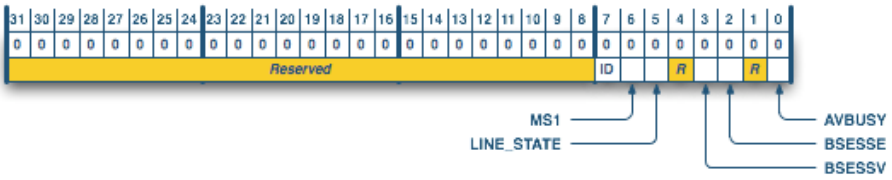
19.8.6 Offset 014h: USB_OTG_IEN – OTG Interrupt Control Register

This register enables the corresponding interrupt status bits defined in USB_OTG_ISTAT.



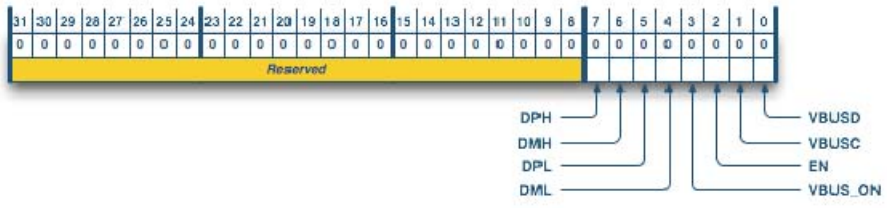
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**ID:** Enables USB_OTG_ISTAT.ID to generate an interrupt
06	RW	0	**MS1:** Enables USB_OTG_ISTAT.MS1 to generate an interrupt.
05	RW	0	**LINE_STATE:** Enables USB_OTG_ISTAT.LINE_STATE to generate an interrupt.
04	*RO*	*0*	*Reserved*
03	RW	0	**BSESSV:** Enables USB_OTG_ISTAT.BSESSV to generate an interrupt.
02	RW	0	**BSESSE:** Enables USB_OTG_ISTAT.BSESSE to generate an interrupt.
01	*RO*	*0*	*Reserved*
00	RW	0	**AVBUSV:** Enables USB_OTG_ISTAT.AVBUSV to generate an interrupt.

19.8.7 Offset 018h: USB_OTG_STAT – OTG Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**ID:** When set, either no cable is attached or a Type B cable has been plugged into the USB connector. When cleared, a Type A cable has been plugged into the USB connector.
06	RW	0	**MS1:** This bit is reserved for the 1ms count, but is not useful to software.
05	RW	0	**LINE_STATE:** Set when USB_CTRL.J and USB_CTRL.SE0 have been static for the previous 1ms, and can be considered de-bounced.
04	*RO*	*0*	*Reserved*
03	RW	0	**BSESSV:** Set when V_{BUS} is above the "B" session valid threshold.
02	RW	0	**BSESSE:** Set when V_{BUS} is below the "B" session end threshold.
01	*RO*	*0*	*Reserved*
00	RW	0	**AVBUSV:** This bit will be set when V_{BUS} is above the "A" VBUS Valid threshold.

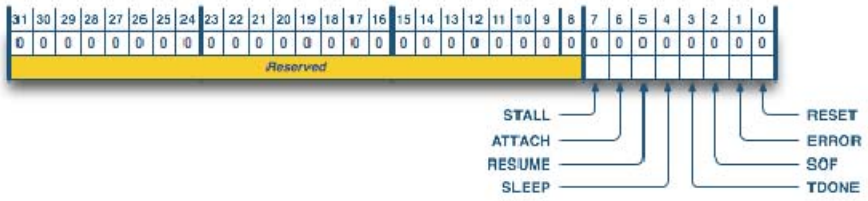
19.8.8 Offset 01Ch: USB_OTG_CTL – OTG Control Register



"

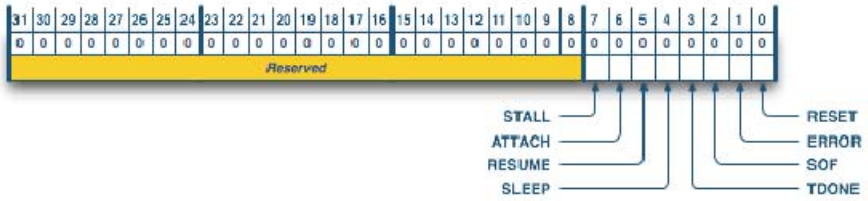
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**D+ High (DPH):** When set, enables a pull-up resistor on D+.
06	RW	0	**D- High (DMH):** When set, enables a pull-up resistor on D-.
05	RW	0	**D+ Low (DPL):** When set, enables a pull-down resistor on D+.
04	RW	0	**D- Low (DML):** When set, enables a pull-down resistor on D-.
03	RW	0	**V_{BUS} On (VBUS_ON):** When set, this will turn on V_{BUS}
02	RW	0	**On the Go Enable (EN):** Only valid if USB_CTRL.EN is set. When set, pull-up and pull-down controls in this register are used. When cleared and USB_CTRL.HOST is cleared, D+ pull up is engaged. When cleared and USB_CTRL.HOST is set, D+ and D- pull down resistors are engaged if the controller is enabled and in host mode.
01	RW	0	**V_{BUS} Charge (VBUSC):** When set, charges VBUS through a resistor.
00	RW	0	**V_{BUS} Discharge (VBUSD):** When set, discharges VBUS through a resistor.

19.8.9 Offset 080h: USB_ISTAT – Interrupt Status Register



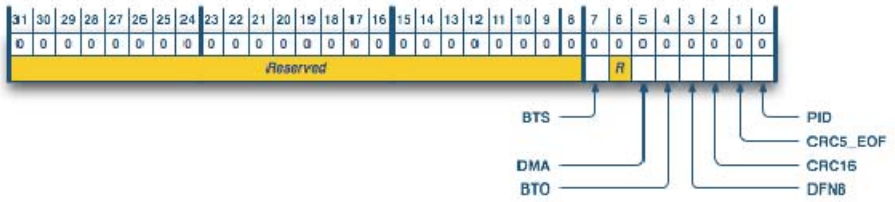
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW1C	0	**Stall (STALL):** In device mode, set when a STALL ACK sent by the controller. In host mode, set when controller detects a STALL ACK during handshake of a USB transaction.
06	RW1C	0	**Attach (ATTACH):** Only valid in host mode. Set when there have been no transitions on USB for 2.5µs and current bus state is not SE0, indicating attachment of a USB peripheral.
05	RW	0	**Resume (RESUME):** Set when a K-state is observed on D+/D- for 2.5usec.
04	RW	0	**Sleep (SLEEP):** This bit is set if the controller has detected a constant idle on the bus for 3 ms. The sleep timer is reset by activity on the bus.
03	RW1C	0	**Token Done (TDONE):** Set when the current token is complete. Clearing this bit causes USB_STAT to be cleared or the STAT holding register to be loaded into USB_STAT.
02	RW	0	**Start of Frame Token (SOF):** In device mode, this bit is set if the controller has received an SOF token. In host mode this bit is set when the SOF threshold is reached.
01	RW	0	**Error (ERROR):** Set when any error condition in USB_ESTAT occurred.
00	RW	0	**Reset (RESET):** Set when the controller detects a USB reset for 2.5 µs. It is not asserted again until the USB reset condition has been removed, and then re-asserted.

19.8.10 Offset 084h: USB_IEN – Interrupt Enable Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Stall (STALL):** Enables USB_ISTAT.STALL to generate an interrupt.
06	RW	0	**Attach (ATTACH):** Enables USB_ISTAT.ATTACH to generate an interrupt.
05	RW	0	**Resume (RESUME):** Enables USB_ISTAT.RESUME to generate an interrupt.
04	RW	0	**Sleep (SLEEP):** Enables USB_ISTAT.SLEEP to generate an interrupt.
03	RW	0	**Token Done (TDONE):** Enabled USB_ISTAT.TDONE to generate an interrupt.
02	RW	0	**Start of Frame Token (SOF):** Enables USB_ISTAT.SOF to generate an interrupt.
01	RW	0	**Error (ERROR):** Enables USB_ISTAT.ERROR to generate an interrupt.
00	RW	0	**Reset (RESET):** Enables USB_ISTAT.RESET to generate an interrupt.

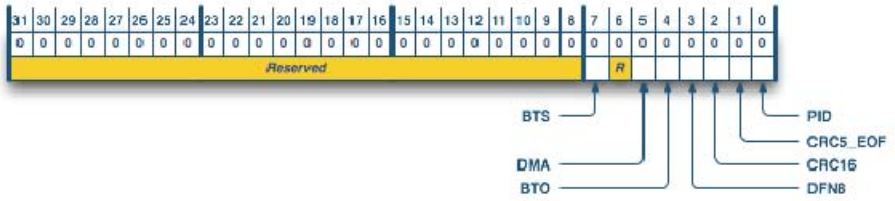
19.8.11 Offset 088h: USB_ESTAT – Error Interrupt Status Register



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Bit Stuff (BTS):** When set, a bit stuff error detected, and the packet rejected.
06	*RO*	*0*	*Reserved*
05	RW	0	**DMA (DMA):** Occurs if there is a transmit data underflow, a receive data overflow, or if a data packet is larger than the buffer allocated in the BDT. In this last case data is truncated as it is put into memory.
04	RW	0	**Turnaround Timeout (BTO):** Set if more that 16-bit times are counted from the previous EOP before a transition from IDLE.
03	RW	0	**DFN8:** When set, the data field received was not 8-bits.
02	RW	0	**CRC16 (CRC16):** When set, the data packet was rejected due to a CRC16 error.
01	RW	0	**CRC5/EOF:** In device mode, indicates a CRC5 error in the token packet. In host mode, the controller is transmitting or receiving data and the SOF counter has expired.
00	RW	0	**PID (PID):** The PID check field failed.

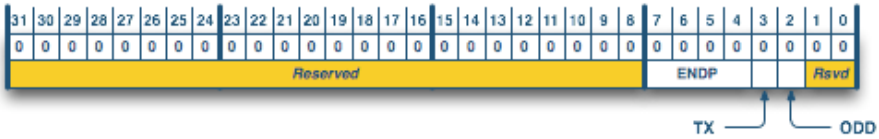
19.8.12 Offset 08Ch: USB_EEN – Error Interrupt Enable Register

Setting any of these bits enables the respective error interrupt source USB_ESTAT.



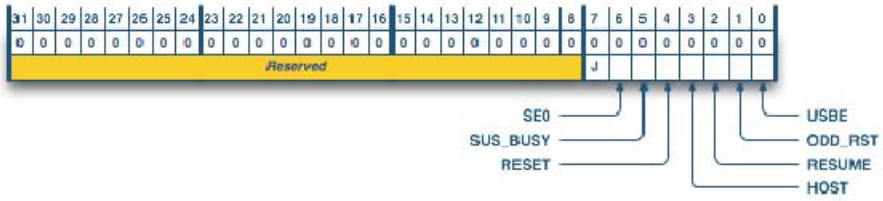
Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**BTS Error (BTS):** Enables USB_ESTAT.BTS to generate an interrupt.
06	*RO*	*0*	*Reserved*
05	RW	0	**DMA (DMA):** Enables USB_ESTAT.DMA to generate an interrupt.
04	RW	0	**Turnaround Timeout (BTO):** Enables USB_ESTAT.BTO to generate an interrupt.
03	RW	0	**DFN8 (DFN8):** Enables USB_ESTAT.DFN8 to generate an interrupt.
02	RW	0	**CRC16 (CRC16):** Enables USB_ESTAT.CRC16 to generate an interrupt.
01	RW	0	**CRC5/EOF (CRC5_EOF):** Enables USB_ESTAT.CRC5_EOF to generate an interrupt.
00	RW	0	**PID (PID):** Enables USB_ESTAT.PID to generate an interrupt.

19.8.13 Offset 090h: USB_STAT – USB Status Register



Bits	Type	Reset	Description
31:08	RO	0	*Reserved*
07:04	RO	0	**Endpoint (ENDP):** Encodes the endpoint address that received or transmitted the previous token. This allows the CPU to determine which BDT entry was updated by the last USB transaction.
03	RO	0	**Transmit (TX):** When set, indicates the last BDT updated was for a transmit. When cleared, the last transaction was a receive data transfer.
02	RO	0	**Odd (ODD):** When set, last buffer descriptor update was in the odd bank of the BDT.
01:00	RO	0	*Reserved*

19.8.14 Offset 094h: USB_CTRL – USB Control Register

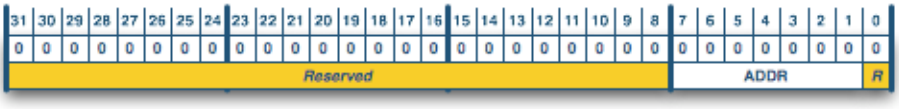


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**J-State (J):** USB differential receiver JSTATE. Polarity is affected by USB_ADDR.LS.
06	RW	0	**Single Ended Zero (SE0):** Live USB Single Ended Zero signal.
05	RW	0	**Transmit Suspend (device) / Token Busy (host) (SUS_BSY):** When set as a device, the controller has disabled packet transmission and reception. Clearing this bit allows the controller to continue token processing. This bit is set when a Setup Token is received. As a host, the controller is executing a USB token and no more token commands must be written.
04	RW	0	**Reset (RESET):** When set, the controller performs USB reset signaling. Only valid in host mode. Software must set this bit to '1' for the required amount of time and then clear it back to '0'. For more information on RESET signaling see the USB specification.
03	RW	0	**Host Enable (HOST):** When set, enables the controller to operate as a host.
02	RW	0	**Resume (RESUME):** When set, the controller executes resume signaling. Software must set this bit for the required amount of time and then clear it to 0. If HOST is set the controller appends a Low Speed End of Packet to the resume signaling when RESUME is cleared.
01	RW	0	**Even/Odd Reset (ODD_RST):** Setting this clears all BDT ODD ping/pong bits.
00	RW	0	**USB Enable (USBE):** Setting this bit enables the controller and resets all ODD bits to the BDT. If HOST is set, SOFs are generated by the controller.

19.8.15 Offset 098h: USB_ADDR – USB Address Register

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Low Speed Enable (LS):** When set, next command written to the token register shall be performed at low speed.
06:00	RW	0	**Address (ADDR):** Address controller decodes in device mode/transmits in host mode.

19.8.16 Offset 09Ch: USB_BDT_PAGE1 – Buffer Descriptor Table Page Register #1

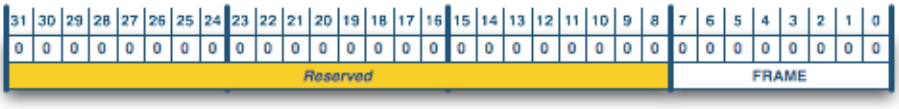


Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:01	RW	00h	**Address (ADDR):** Bits 15:09 of the base address of the BDT.
00	*RO*	*0*	*Reserved*

19.8.17 Offset 0A0h: USB_FRAMEL – USB Frame Number Register Low

Bits	Type	Reset	Description
31:07	*RO*	*0*	*Reserved*
07:00	RW	00h	**Frame (FRAME):** Low order bits of the frame number, combined with USB_FRAMEH.FRAME.

19.8.18 Offset 0A4h: USB_FRAMEH – USB Frame Number Register High



Bits	Type	Reset	Description
31:04	*RO*	*0*	*Reserved*
03:00	RW	00h	**Frame (FRAME):** High order bits of the frame number, combined with USB_FRAMEL.FRAME.

19.8.19 Offset 0A8h: USB_TOKEN – USB Token Register

Once written, the controller starts a transaction to USB_ADDR and endpoint control register 0.

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:04	RW	0h	**Endpoint (EP):** Endpoint for the token command.
3:0	RW	0h	**PID (PID):** Token type that will be executed by the VSUB. Valid tokens are: • 1h: OUT token • 9h: IN token • Dh: SETUP token

19.8.20 Offset 0ACh: USB_SOFT – USB SOF Threshold Register

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Count (CNT):** Only valid in host mode. Sets the number of byte times *before* SOF to stop initiating token packet transactions.

19.8.21 Offset 0B4h: USB_BDT_PAGE2 – Buffer Descriptor Table Page Register #2

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Address (ADDR):** Bits 23:16 of the base address of the BDT.

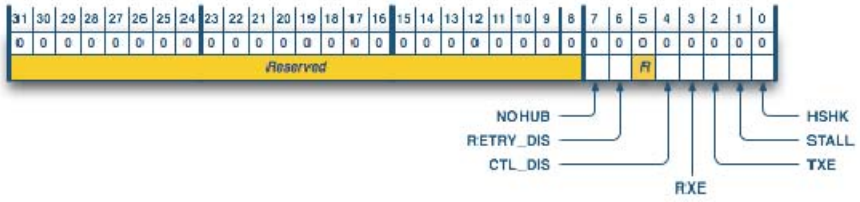
19.8.22 Offset 0B8h: USB_BDT_PAGE3 – Buffer Descriptor Table Page Register #3

Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07:00	RW	00h	**Address (ADDR):** Bits 31:24 of the base address of the BDT.

19.8.23 USB_ENDPTn_CTRL – Endpoint "N" Control Registers

Offset	Endpoint	Offset	Endpoint	Offset	Endpoint	Offset	Endpoint
C0h	0	D0h	4	E0h	8	F0h	12
C4h	1	D4h	5	E4h	9	F4h	13
C8h	2	D8h	6	E8h	10	F8h	14
CCh	3	DCh	7	ECh	11	FCh	15

These registers contain endpoint control bits for each of the 16 endpoints available. Offsets for each endpoint are above, and a description of the register is shown below.



Bits	Type	Reset	Description
31:08	*RO*	*0*	*Reserved*
07	RW	0	**Host Without Hub (NOHUB):** Host mode only. Only present in the control register for endpoint 0. When set, allows the host to communicate to a directly connected low speed device. When cleared host produces the PRE_PID then switches to low speed signaling when sending a token to a low speed device.
06	RW	0	**Retry Disable (RETRY_DIS):** Host mode only. Only present in the control register for endpoint 0. When set, host does not retry NAKed transactions. The BDT PID field will be updated with the NAK PID, and the token done interrupt is set. When cleared, NAKed transactions are retried by hardware. This bit must be set when the host is attempting to poll an interrupt endpoint.
05	*RO*	*0*	*Reserved*
04	RW	0	**Control Disable (CTL_DIS):** See Table 19-3.
03	RW	0	**Receive Enable (RXE):** See Table 19-3.
02	RW	0	**Transmit Enable (RXE):** See Table 19-3.
01	RW	0	**Stall (STALL):** When set, an endpoint is stalled. This bit has priority over all other bits in this register, but is only valid if TXE=1 or RXE=1. Any access to this endpoint will cause the controller to return a STALL handshake.
00	RW	0	**Handshake (HSHK):** When set, the endpoint performs handshaking during a transaction to this endpoint. Not set for an isoch endpoint.

Table 19-3: Endpoint Enable / Direction Control

CTL_DIS	RXE	TXE	Endpoint Enable/Direction Control
X	0	0	Disable Endpoint
X	0	1	Enable Endpoint for TX transfers only
X	1	0	Enable Endpoint for RX transfers only
1	1	1	Enable Endpoint for both RX and TX transfers

Chapter 20: General-Purpose Input/Output (GPIO)

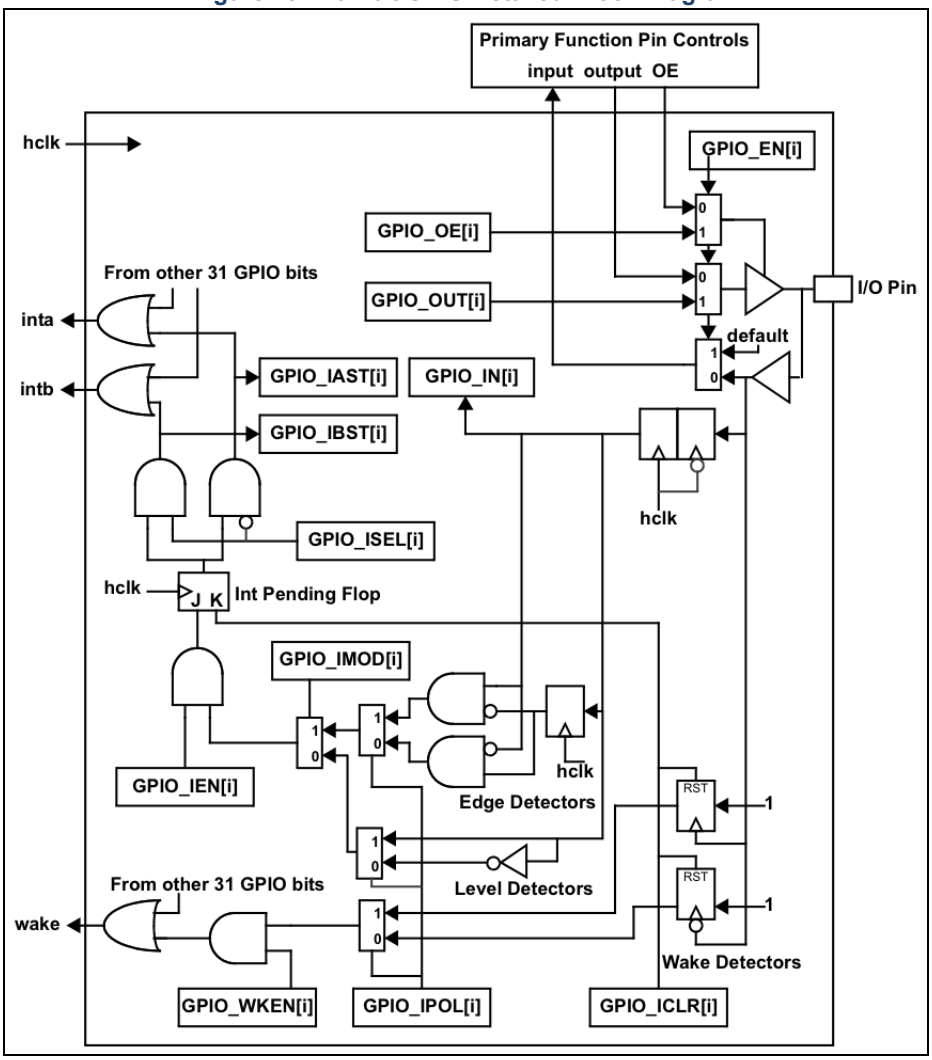
Three APB based 32-bit General-Purpose Input/Output (GPIO) modules are included for a total of up to 76 GPIO pins. The GPIO module enables direct I/O control of the GPIO pins. Most pins are shared with a primary pin function and can be used secondarily as GPIOs when the primary pin function is unused. Though this multiplexing between primary and secondary functions is usually static, it may also be done dynamically.

The primary features of the GPIO module includes:

- Multiplexing between primary and secondary (GPIO) functions
- Configuration of each GPIO pin as input, output or I/O
- Open-Drain capability on GPIO0[15:0]
- Up to six independent interrupts:
- Two interrupts can be generated from each 32-bit GPIO module.
- Any GPIO pin can be assigned to either interrupt.
- Each GPIO can generate an interrupt for a High or Low level or on rising or falling edge.
- Configuration to wake the PMU on rising or falling edge of any GPIO pin

20.1 GPIO Configuration

Figure 20-1: 32-bit GPIO Detailed Block Diagram



20.2 Multiplexed Pins

Pin Name	Primary Function		GPIO Function		After Reset	
	Name	Dir	Name	Default	Function	Dir
TxREQ	TxREQ	I	GPIO_2[8]	0	GPIO_2[8]	I
TxACK	TxACK	O	GPIO_2[9]	N/A	GPIO_2[9]	I
RxREQ	RxREQ	I	GPIO_2[10]	0	GPIO_2[10]	I
RxACK	RxACK	O	GPIO_2[11]	N/A	GPIO_2[11]	I
nCS[6]	nCS[6]	O	GPIO_0[16]	N/A	nCS[6]	O
nCS[7]	nCS[7]	O	GPIO_0[17]	N/A	nCS[7]	O
nCS[8]	nCS[8]	O	GPIO_0[18]	N/A	nCS[8]	O
nCS[9]	nCS[9]	O	GPIO_0[19]	N/A	nCS[9]	O
MA[20]	MA[20]	O	GPIO_0[20]	N/A	MA[20]	O
MA[21]	MA[21]	O	GPIO_0[21]	N/A	MA[21]	O
MA[22]	MA[22]	O	GPIO_0[22]	N/A	MA[22]	O
MA[23]	MA[23]	O	GPIO_0[23]	N/A	MA[23]	O
READY	READY	I	GPIO_0[26]	0	GPIO_0[26]	I
PWM/TCLK[0]	PWM/TCLK[0]	I/O	GPIO_0[27]	PWM/TCLK[0]	GPIO_0[27]	I
PWM/TCLK[1]	PWM/TCLK[1]	I/O	GPIO_0[28]	PWM/TCLK[1]	GPIO_0[28]	I
PWM/TCLK[2]	PWM/TCLK[2]	I/O	GPIO_0[29]	PWM/TCLK[2]	GPIO_0[29]	I
PWM/TCLK[3]	PWM/TCLK[3]	I/O	GPIO_0[30]	PWM/TCLK[3]	GPIO_0[30]	I
RxD[2]	RxD[2]	I	GPIO_0[31]	1	GPIO_0[31]	I
TxD[2]	TxD[2]	O	GPIO_1[0]	N/A	GPIO_1[0]	I
nCTS[0]	nCTS[0]	I	GPIO_1[1]	0	GPIO_1[1]	I
nCTS[1]	nCTS[1]	I	GPIO_1[2]	0	GPIO_1[2]	I
nCTS[2]	nCTS[2]	I	GPIO_1[3]	0	GPIO_1[3]	I
nRTS[0]	nRTS[0]	O	GPIO_1[4]	N/A	GPIO_1[4]	I
nRTS[1]	nRTS[1]	O	GPIO_1[5]	N/A	GPIO_1[5]	I
nRTS[2]	nRTS[2]	O	GPIO_1[6]	N/A	GPIO_1[6]	I
nDCD[0]	nDCD[0]	I	GPIO_1[7]	0	GPIO_1[7]	I
nDTR[0]	nDTR[0]	O	GPIO_1[8]	N/A	GPIO_1[8]	I
nDSR[0]	nDSR[0]	I	GPIO_1[9]	0	GPIO_1[9]	I
nRI[0]	nRI[0]	I	GPIO_1[10]	1	GPIO_1[10]	I
MISO[0]	MISO[0]	I/O	GPIO_1[11]	0	GPIO_1[11]	I
MISO[1]	MISO[1]	I/O	GPIO_1[12]	0	GPIO_1[12]	I
MOSI[0]	MOSI[0]	I/O	GPIO_1[13]	0	GPIO_1[13]	I
MOSI[1]	MOSI[1]	I/O	GPIO_1[14]	0	GPIO_1[14]	I
SCK[0]	SCK[0]	I/O	GPIO_1[15]	0	GPIO_1[15]	I
SCK[1]	SCK[1]	I/O	GPIO_1[16]	0	GPIO_1[16]	I
nSS[0]	nSS[0]	I/O	GPIO_1[17]	1	GPIO_1[17]	I
nSS[1]	nSS[1]	I/O	GPIO_1[18]	1	GPIO_1[18]	I
SC_CLK[0]	SC_CLK[0]	O	GPIO_1[19]	N/A	GPIO_1[19]	I
SC_CLK[1]	SC_CLK[1]	O	GPIO_1[20]	N/A	GPIO_1[20]	I
SC_IO[0]	SC_IO[0]	I/O	GPIO_1[21]	1	GPIO_1[21]	I
SC_IO[1]	SC_IO[1]	I/O	GPIO_1[22]	1	GPIO_1[22]	I
SC_nALARM	SC_nALARM	I	GPIO_1[23]	1	GPIO_1[23]	I
SC_SCK	SC_SCK	O	GPIO_1[24]	N/A	GPIO_1[24]	I
SC_MOSI	SC_MOSI	O	GPIO_1[25]	N/A	GPIO_1[25]	I
SC_MISO	SC_MISO	I	GPIO_1[26]	0	GPIO_1[26]	I
SC_nSS[0]	SC_nSS[0]	O	GPIO_1[27]	N/A	GPIO_1[27]	I
SC_nSS[1]	SC_nSS[1]	O	GPIO_1[28]	N/A	GPIO_1[28]	I
LCD_E	LCD_E	O	GPIO_1[29]	N/A	GPIO_1[29]	I
LCD_RS	LCD_RS	O	GPIO_1[30]	N/A	GPIO_1[30]	I
LCD_RnW	LCD_RnW	O	GPIO_1[31]	N/A	GPIO_1[31]	I
LCD_D[0]	LCD_D[0]	I/O	GPIO_2[0]	0	GPIO_2[0]	I
LCD_D[1]	LCD_D[1]	I/O	GPIO_2[1]	0	GPIO_2[1]	I
LCD_D[2]	LCD_D[2]	I/O	GPIO_2[2]	0	GPIO_2[2]	I
LCD_D[3]	LCD_D[3]	I/O	GPIO_2[3]	0	GPIO_2[3]	I
LCD_D[4]	LCD_D[4]	I/O	GPIO_2[4]	0	GPIO_2[4]	I
LCD_D[5]	LCD_D[5]	I/O	GPIO_2[5]	0	GPIO_2[5]	I
LCD_D[6]	LCD_D[6]	I/O	GPIO_2[6]	0	GPIO_2[6]	I
LCD_D[7]	LCD_D[7]	I/O	GPIO_2[7]	0	GPIO_2[7]	I

20.3 Registers (Base: GPIO0→FFFF5000h, GPIO1→FFFF6000h, GPIO2→FFFF7000h)

In all cases, the bits of these correspond to a single GPIO pin: bit 0 corresponds to GPIO_0[0], bit 1 to GPIO_0[1], etc. Unless otherwise specified, the default state for each bit in these registers is '0'.

Offset	R/W	Description
00h	RW	**Enable (GPIO_EN):** When set, the secondary (GPIO) function is enabled. This register can be written directly or by using GPIO_EN_SET and GPIO_EN_CLR. • GPIO0 default value: FF00FFFFh • GPIO1 default value: FFFFFFFFh • GPIO2 default value: 00000FFFh
04h	WO	**Enable Set (GPIO_EN_SET):** Writing a '1' sets GPIO_EN.
08h	WO	**Enable Clear (GPIO_EN_CLR):** Writing a '1' clears GPIO_EN.
0Ch	RW	**Output Enable (GPIO_OE):** When '0', disable output. When '1', enable output. This can be written directly or by using GPIO_OE_SET and GPIO_OE_CLR.
10h	WO	**Output Enable Set (GPIO_OE_SET):** Writing a '1' sets GPIO_OE.
14h	WO	**Output Enable Clear (GPIO_OE_CLR):** Writing a '1' clears the GPIO_OE.
18h	RW	**Output (GPIO_OUT):** When '0', drive '0' on GPIO output. When '1', drive '1' on GPIO output. This can be written directly or by using GPIO_OUT_SET and GPIO_OUT_CLR. GPIO_OE and GPIO_EN must be active.
1Ch	WO	**Output Set (GPIO_OUT_SEL):** Writing a '1' sets GPIO_OUT.
20h	WO	**Output Clear (GPIO_OUT_CLR):** Writing a '1' clears GPIO_OUT.
24h	RO	**Input (GPIO_IN):** '0' means pin is a '0'. '1' means pin is a '1'. Always represents the state of the pin, even if GPIO_EN is inactive. Reset value is undefined.
28h	RW	**Interrupt Mode (GPIO_IMOD):** '0' = level triggered. '1' = edge triggered.
2Ch	RW	**Polarity (GPIO_IPOL):** '0' means interrupts latched on "falling" condition. '1' means interrupts A/B are latched on a "rising" condition
30h	RW	**Interrupt Channel Select (GPIO_ISEL):** When set, use interrupt channel B. When cleared, use interrupt channel A.
34h	RW	**Interrupt Enable (GPIO_IEN):** When set, allows the Interrupt Pending Flop to be set. Disabling this bit does not mask an existing pending interrupt from driving an interrupt into the INTC.
38h	WO	**Interrupt Enable Set (GPIO_IEN_SET):** Writing a '1' sets GPIO_IEN.
3Ch	WO	**Interrupt Enable Clear (GPIO_IEN_CLR):** Writing a '1' clears GPIO_IEN.
40h	RO	**Interrupt Channel-A Status (GPIO_IAST):** When set, outstanding channel A interrupt associated with this pin. Use GPIO_ICLR to clear.
44h	RO	**Interrupt Channel-B Status (GPIO_IBST):** When set, outstanding channel B interrupt associated with this pin. Use GPIO_ICLR to clear.
48h	WO	**Interrupt Clear (GPIO_ICLR):** When written to '1', clears Interrupt Pending Flop and Edge Detector Flops. Clearing Interrupt Pending Flop clears the corresponding bit in GPIO_IAST and GPIO_IBST.
4Ch	RW	**Wake (GPIO_WKEN):** When set, enables PMU wake for pin on an edge condition.
50h	WO	**Wake Enable Set (GPIO_WKEN_SET):** Writing a '1' sets GPIO_WKEN.
54h	WO	**Wake Enable Clear (GPIO_WKEN_CLR):** Writing a '1' clears GPIO_WKEN.
58h	RW	**Open-Drain register (GPIO_OPEN_D):** This only applies to GPIO0 [15:0]. When set, enables open-drain mode.

20.4 Using Output

A GPIO pin can always be used as an input. The state of the pin can be sampled through GPIO_IN even when configured to operate as a primary function (GPIO_EN = '0') or when configured as an output (GPIO_OE = '1').

To be used as an output, GPIO_EN must be set. Once set, the pin can be forced to drive a value on the pin by setting GPIO_OE. With GPIO_OE=1, setting GPIO_OUT to 0 drive a '0', while GPIO_OUT=1 drives a '1'. The pin can be tri-stated by clearing GPIO_OE. GPIO_EN, GPIO_OE, and GPIO_OUT can be modified either by writing to them directly or by writing to the associated SET and CLR registers. This allows bits to be changed without a Read-Modify-Write operation.

20.5 Configuring Interrupts

Below is the suggested sequence for the initial configuration of GPIO interrupts. This can be done for any number of GPIO bits. The GPIO Interrupt function can be used independent of the GPIO_EN and GPIO_OE settings.

1. Disable interrupts by clearing GPIO_IEN. This prevents the Interrupt Pending Flop from being set but does not mask an already pending interrupt in the flop. See JK-flop labeled Interrupt Pending Flop in Figure 20-1. GPIO_IEN can also be cleared by writing to GPIO_IEN_CLR.
2. Clear any pending interrupts by writing to GPIO_ICLR. This clears the Interrupt Pending Flop.
3. Write to GPIO_IMOD to select level/edge mode.
4. Write to GPIO_IPOL to select either rising/falling trigger level.
5. Write to GPIO_ISEL to select either A/B channel. This provides two priority levels within the interrupt controller.
6. Write to GPIO_IEN to re-enable interrupts. Once set, the Interrupt Pending Flop can be set and interrupts are active and ready. GPIO_IEN can also be set by writing to GPIO_IEN_SET.
7. Configure the Interrupt Controller as given section Chapter 6:. You may want to do this last step prior to setting the GPIO_IEN bits.

20.6 Handling Interrupts

The recommended sequence of GPIO interrupt handling is provided below:

1. Based upon GPIO_ISEL, read either GPIO_IAST or GPIO_IBST to determine the source(s).
2. Perform application specific interrupt tasks associated with the bit(s).
3. Clear the Interrupt Pending Flop by writing to GPIO_ICLR. This also clears and re-arms the rising and falling edge trigger flops.
4. Signal and End-of-Interrupt to the interrupt controller. See section Chapter 6: for more details
5. Return from the ISR.

20.7 Wake Function

Below are suggested steps to enable wake. Like interrupts, wake is independent of GPIO_EN and GPIO_OE:

1. Set the polarity (rising or falling edge) by writing to GPIO_IPOL. The Wake function relies upon rising and falling edge detectors to detect and latch the transition on the GPIO pin. These flops operate asynchronously and do not require the GPIO clock to be active.
2. Clear the edge detector flops by writing to the GPIO_ICLR register.
3. Activate the GPIO Wake function by writing to the GPIO_WKEN or GPIO_WKEN_SET register.
4. Configure the PMU to wake upon GPIO per section Chapter 3:.

Chapter 21: Electrical Characteristics

The data in this chapter is pre-qualification and pre-characterization and is subject to change.

21.1 Absolute Maximum Ratings

Stresses greater than those listed below may cause permanent damage to the device. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. For improved reliability, tie unused inputs to one of the supply voltages (V_{DDIO} or V_{SS}).

Table 21-1: Absolute Maximum Ratings

Parameter		Min	Max	Units
Operating Temperature	Standard Temp Full Chip	0	+70	^{0}C
	Extended Temp Full Chip	–40	+85	^{0}C
	Battery Only	–40	+125	^{0}C
Storage temperature		–65	+150	^{0}C
Voltage on any pin with respect to V_{SS}[2]	Standard	–0.3	+4.1	V
	5V-Tolerant	–0.3	+5.5	V
Voltage on power pin with respect to V_{SS}	V_{DDIO}	–0.3	+4.1	V
	V_{DD}	–0.3	+2.4	V
	V_{BAT}	–0.3	+4.1	V
	V_{DDRTC}	–0.3	+4.1	V
	V_{DDPLL}	–0.3	+2.4	V
	V_{DDMCR}	–0.3	+4.1	V
	V_{DDADC}	–0.3	+4.1	V
	V_{DDUSB}	–0.3	+4.1	V
Maximum current on input and/or inactive output pin[2]		–10	+10	µA
Maximum output current from active output pin	nWEL, nWEU, nOE, MA[24:0], MD[15:0]	–8	+8	mA
	Other pins	–4	+4	mA
Total power dissipation[1]		—	900	mW

1. Min and max power estimated with max clock rate and 20% toggle rate on all flops and I/Os.
2. At maximum voltage, max current for the pin cannot be exceeded.

21.2 DC Characteristics

Symbol	Parameter	T_A = –40°C to 85°C Min	Max	Units	Conditions
V_{DDIO}	I/O Supply Voltage	3.0	3.6	V	
V_{DD}	Core Supply Voltage	1.71	1.89	V	
V_{BAT}	Battery Supply Voltage	2.36	3.6	V	
V_{DDRTC}	RTC Supply Voltage	3.03	3.6	V	
V_{DDPLL}	PLL Supply Voltage	1.71	1.89	V	
V_{DDMCR}	MCR Supply Voltage	3.0	3.6	V	
V_{DDADC}	ADC Supply Voltage	3.0	3.6	V	
V_{DDUSB}	USB Supply Voltage	3.0	3.6	V	
V_{IL}	Low Level Input Voltage	—	0.8	V	Excludes RTCXI, RSTIN, and CLKXI
		—	0.6	V	Includes RTCXI, RSTIN, and CLKXI
V_{IH}	High Level Input Voltage	2.0	—	V	Excludes RTCXI, RSTIN, and CLKXI
		2.2	—	V	Includes RTCXI, RSTIN, and CLKXI
V_{OL}	Low Level Output Voltage	—	0.4	V	I_{OL} = 4mA; V_{DDIO} = 3.3V
V_{OH}	High Level Output Voltage	2.4	—	V	I_{OH} = –4mA; V_{DDIO} = 3.3V
C_{PAD}	Pad Capacitance	5.0	5.0	pF	
C_{XIN}	XIN Pad Capacitance	5.0	5.0	pF	
C_{XOUT}	XOUT Pad Capacitance	5.5	5.5	pF	
I_{IL}	Input Leakage Current	–10	+10	µA	V_{DDIO} = 3.6V; V_{IN} = V_{DDIO} or V_{SS}[1]
I_{TL}	Tristate Leakage Current	–10	+10	µA	V_{DD} = 3.6V
I_{PU}	Weak Pull-up Current	–60	-175	µA	
I_{CCS}	Supply Currents in STOP with USB powered down	130	130	µA	No I/O switching, all blocks and oscillator disabled. V_{DD} +V_{DDPLL}
		40	40	µA	V_{DDIO} + V_{DDADC} + V_{DDMCR}+ V_{DDUSB} + V_{DDRTC}
I_{CCI}	Supply Currents in IDLE	61	61	mA	No I/O switching, all blocks enabled. V_{DD} +V_{DDPLL}
		10	10	mA	V_{DDIO} + V_{DDADC}+ V_{DDMCR}+ V_{DDUSB} + V_{DDSEC}
I_{CCA}	Supply Currents Active	110	110	mA	Normal activity at Typical conditions using CPU, Analog, and 16 I/O switching at ~20MHz. V_{DD} +V_{DDPLL}
		25	25	mA	V_{DDIO} +V_{DDADC} + V_{DDMCR}+ V_{DDUSB} + V_{DDSEC}
I_{CCMHA}	Supply Currents High Active	150	150	mA	High activity at typ conditions using CPU, MCR, SmartCard with external interface and SDRAM. V_{DD} +V_{DDPLL}
		45	45	mA	V_{DDIO} + V_{DDADC} + V_{DDMCR}+ V_{DDUSB} + V_{DDSEC}
I_{CCB}	Supply Current from Battery (backup mode, V_{BAT} only)	18	18	µA	RTC running. (V_{BAT} only)
	Supply Current from Battery (with system power on)	18	18	µA	Battery Backup with RTC running.(V_{BAT} > AV_{DD_RTC})
		1.2	1.2	µA	

1. This condition excludes all pins that have on-chip pull-ups, when driven Low.

21.3 AC Characteristics

Table 21-2: AC Characteristics

Symbol	Parameter	V_{DDIO} = 3.0 – 3.6V, V_{DDCORE} = 1.71–1.89V, T_A = –40°C to 85°C		Units	Conditions
		Min	Max		
F_{CLK}	System Clock Frequency	—	200	MHz	fclk
		—	100	MHz	hclk, CLKOUT
F_{CLKXI}	Crystal Frequency	14.0	40.0	MHz	Without USB
		24.0	24.0	MHz	Required for USB operation
T_{CLKXI}	CLKXI Clock Period	25.0	—	ns	$T_{CLKXI} = 1/F_{CLKXI}$
T_{XINH}	CLKXI High Time	40%T_{CLKXI}	60%T_{CLK_XI}	ns	T_{CLKXI} = 25ns
T_{XINL}	CLKXI Low Time	40%T_{CLKXI}	60%T_{CLK_XI}	ns	T_{CLKXI} = 25ns
T_{XINR}	CLKXI Rise Time	—	2	ns	T_{CLKXI} = 25ns
T_{XINF}	CLKXI Fall Time	—	2	ns	T_{CLKXI} = 25ns

Table 21-3: Power-On Reset Electrical Characteristics and Timing

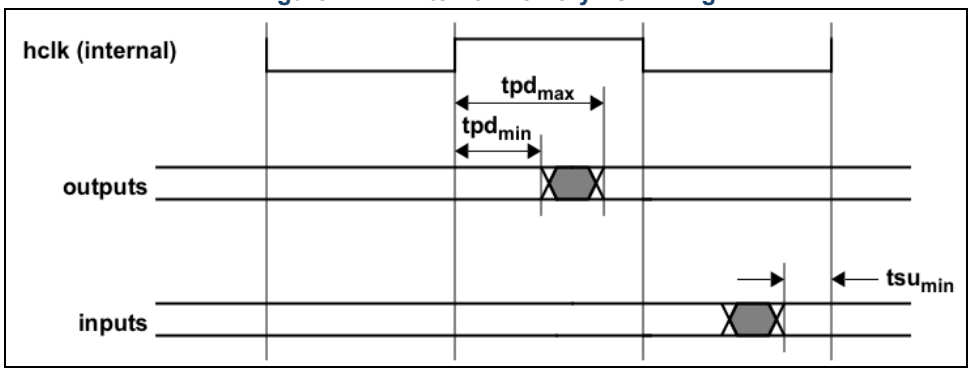
Symbol	Parameter	T_A = –40°C to 85°C		Units	Conditions
		Min	Max		
V_{POR}	POR Voltage Threshold for Core Power	1.62	1.71	V	After UV POR TRIM. $V_{DD} = V_{POR}$
V_{OVR}	Over Voltage Reset Threshold for Core Power	1.89	1.98	V	After OV POR TRIM. $V_{DD} = V_{POR}$
	V_{POR} hysteresis	0	60	mV	
	Starting V_{DD} voltage to ensure valid Power-On Reset.	—	—	V	
T_{PORD}	Power-On Reset Delay		10	µs	
T_{POR}	Power-On Reset Digital Delay			clocks	Cycles after Crystal is stable.

Table 21-4: Analog-to-Digital Converter Electrical Characteristics

Symbol	Parameter	V_{DDA} = 3.0 – 3.6V, T_A = 0°C to 85°C			Units	Conditions
		Min	Typ	Max		
	Full Scale Input Range	0	—	V_{REF}	V	
	Maximum Sample Rate		—	45	kSps	
	Data Latency	—	12	—	cycles	
	Input Clock Frequency	—	540	—	kHz	
	Resolution	—	10	—	bits	
	Differential Nonlinearity (DNL)	–1.0	—	1.0	LSB	
	Integral Nonlinearity (INL)	–2.0	—	2.0	LSB	
	DC Offset Error		±2		% FS	Percent of Full Scale
V_{REF}	ADC Reference Voltage	—	V_{DDADC}	—	V	Should filter for best performance

21.4 External Memory Timing

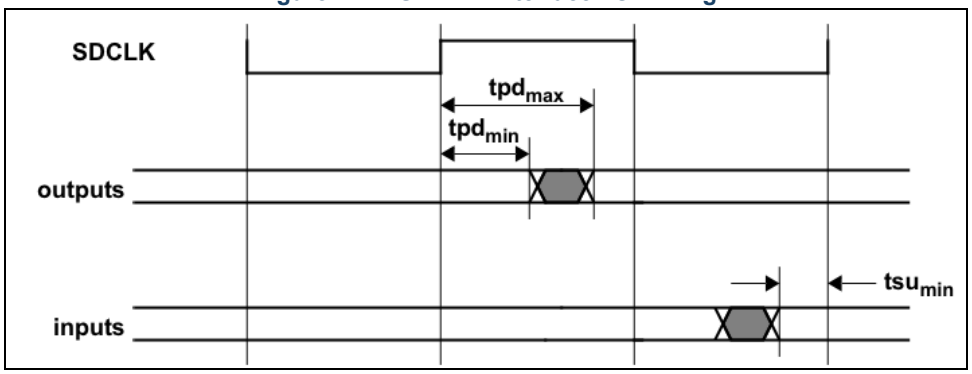
Figure 21-1: External Memory I/O Timing



Parameter	Description	Symbol	Pins	Min	Max
Output Delay	Delay from internal hclk to output pin	t_{PD}	nCS[9:0], nWEU, nWEL, nOE	1.92ns	4.72ns
			MA[12:0], MD[15:0]	1.83ns	4.72ns
Setup Time	Setup time from input stable to internal hclk	t_{SU}	MD[15:0]	5.78ns	—

21.5 SDRAM Timing

Figure 21-2: SDRAM Interface I/O Timing

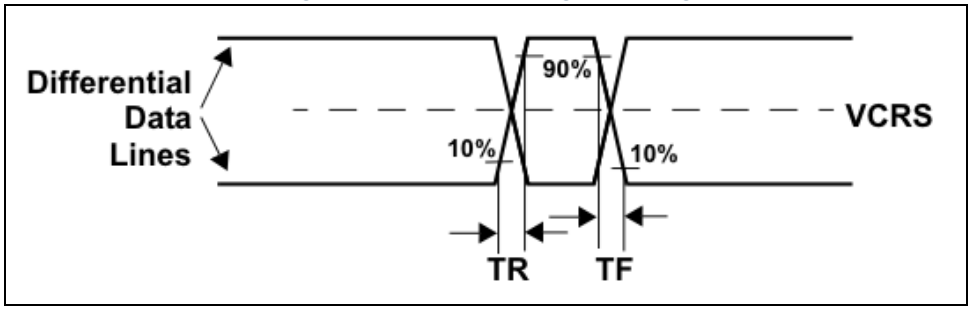


Parameter	Description	Symbol	Pins	Min	Max
Output Delay	Delay from rising edge of SDCLK to output valid	t_{PD}	CKE, nCS[1], nRAS, nCAS, nWE, DQM[1:0]	1.92ns	6.06ns
			MA[12:0], MD[15:0]	1.91ns	6.85ns
Setup Time	Setup time from input valid to rising edge of SDCLK	t_{SU}	MD[15:0]	3.49ns	—

21.6 USB Electrical and Timing

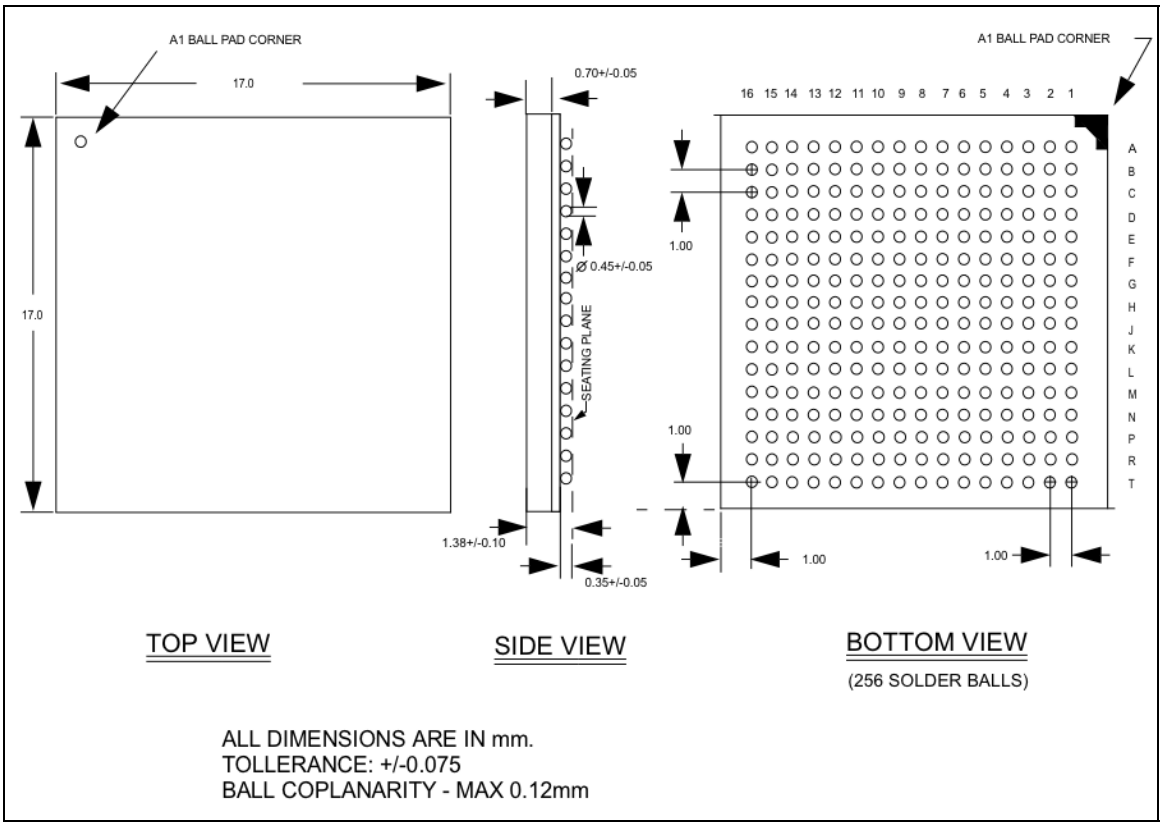
The Z32AN Series conforms to the Universal Serial Bus 2.0 standard specification.

Figure 21-3: USB Data Signal Timing



Symbol	Parameter	T_A = –40°C to 85°C		Units	Conditions
		Min	Max		
I_{CCUSB}	Supply Currents in STOP with USB Suspend	—	—	µA	No I/O switching, all blocks and oscillator disabled. USB Transceivers configured to wake with activity.
t_R	Time for data to rise from 10% to 90% of rail values	4.0	20.0	ns	USB Full speed
t_F	Time for data to fall from 90% to 10% of rail values	4.0	20.0	ns	USB Full speed

Chapter 22: Packaging



22.1 Soldering Information

All Zilog products designated with as “Green” are RoHS compliant.

22.2 Top Mark

The Top Mark contains 4 lines, as follows:

- Line 1: Zilog Logo and ARM Logo.
- Line 2: Denotes the SOC part number. Example: Z32AN12NW200XG
- Line 3: Denotes Customer ID or special lot number and type of lot (i.e. engineering sample).
- Line 4: Date code: Year (YY), week (XX), and lot coding (BB).
- Line 5: Country of origin.

Chapter 23: Ordering Information

The table below lists the available packages for Z32AN Series ARM SOC and provides a brief description of each product.

Part Number	RAM (KB)	Smart Card Reader IF	Magnetic Card Reader	I/O	UARTs	SPI	ADC Inputs	USB	Timers	RTC	256 BGA package	-40°C to 85°C Temp Range
Z32AN00NW200SG	64	0	0	76	3	2	4	2	9	Yes	X	X
Z32AN01NW200SG	64	0	1	76	3	2	4	2	9	Yes	X	X
Z32AN10NW200SG	64	2	0	76	3	2	4	2	9	Yes	X	X

Please contact your local Zilog® sales offices for assistance in ordering the part(s) required. The product represented by this document is newly introduced and Zilog has not completed the full characterization of the product. The document states what Zilog knows about this product at this time, but additional features or non-conformance with some aspects of the document may be found, either by Zilog or its customers in the course of further application and characterization work. In addition, Zilog cautions that delivery may be uncertain at times, due to start-up yield issues. For more information, please visit www.zilog.com.

Chapter 24: Customer Support

For answers to technical questions about the Z32AN, documentation, or any other issues, please contact techsupport@zilog.com.