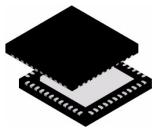
STM32W108HB, STM32W108CC STM32W108CB and STM32W108CZ

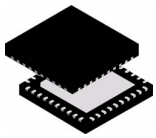
High-performance, IEEE 802.15.4 wireless system-on-chip with embedded Flash memory

Features

- Complete system-on-chip
  - 32-bit ARM® Cortex™-M3 processor
  - 2.4 GHz IEEE 802.15.4 transceiver & lower MAC
  - 128/192/256-Kbyte Flash, 8/12/16-Kbyte RAM memory
  - AES128 encryption accelerator
  - Flexible ADC, SPI/UART/I^2C serial communications, and general-purpose timers
  - 24 highly configurable GPIOs with Schmitt trigger inputs
- Industry-leading ARM® Cortex™-M3 processor
  - Leading 32-bit processing performance
  - Highly efficient Thumb®-2 instruction set
  - Operation at 6, 12 or 24 MHz
  - Flexible nested vectored interrupt controller
- Low power consumption, advanced management
  - Receive current (w/ CPU): 27 mA
  - Transmit current (w/ CPU, +3 dBm TX): 31 mA
  - Low deep sleep current, with retained RAM and GPIO: 400 nA/800 nA with/without sleep timer
  - Low-frequency internal RC oscillator for low-power sleep timing
  - High-frequency internal RC oscillator for fast (100 µs) processor start-up from sleep
- Exceptional RF performance
  - Normal mode link budget up to 102 dB; configurable up to 107 dB
  - -99 dBm normal RX sensitivity; configurable to -100 dBm (1% PER, 20 byte packet)
  - +3 dB normal mode output power; configurable up to +8 dBm



VFQFPN48
(7 x 7 mm)



VFQFPN40
(6 x 6 mm)

  - Robust WiFi and Bluetooth coexistence
- Innovative network and processor debug
  - Non-intrusive hardware packet trace
  - Serial wire/JTAG interface
  - Standard ARM debug capabilities: Flash patch & breakpoint; data watchpoint & trace; instrumentation trace macrocell
- Application flexibility
  - Single voltage operation: 2.1-3.6 V with internal 1.8 V and 1.25 V regulators
  - Optional 32.768 kHz crystal for higher timer accuracy
  - Low external component count with single 24 MHz crystal
  - Support for external power amplifier
  - Small 7x7 mm 48-pin VFQFPN package or 6x6 mm 40-pin VFQFPN package

Applications

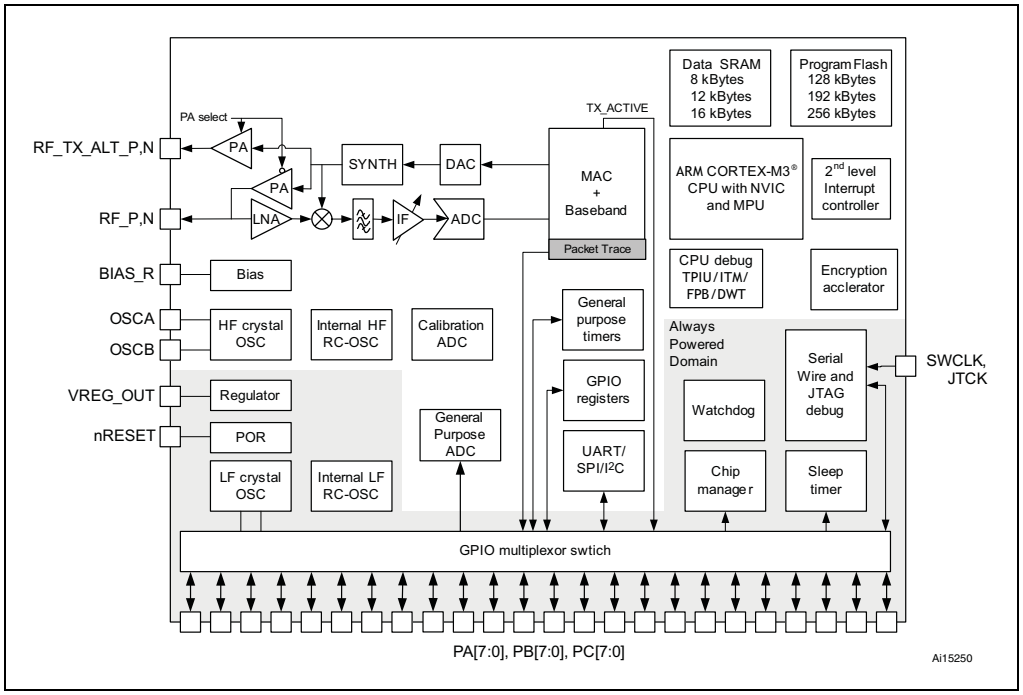
- Smart energy
- Building automation and control
- Home automation and control
- Security and monitoring
- ZigBee® Pro wireless sensor networking
- RF4CE products and remote controls
- 6LoWPAN and custom protocols

Contents

1 Description

The STM32W108 is a fully integrated System-on-Chip that integrates a 2.4 GHz, IEEE 802.15.4-compliant transceiver, 32-bit ARM® Cortex™-M3 microprocessor, Flash and RAM memory, and peripherals of use to designers of 802.15.4-based systems.

Figure 1. STM32W108 block diagram



The transceiver utilizes an efficient architecture that exceeds the dynamic range requirements imposed by the IEEE 802.15.4-2003 standard by over 15 dB. The integrated receive channel filtering allows for robust co-existence with other communication standards in the 2.4 GHz spectrum, such as IEEE 802.11 and Bluetooth. The integrated regulator, VCO, loop filter, and power amplifier keep the external component count low. An optional high performance radio mode (boost mode) is software-selectable to boost dynamic range.

The integrated 32-bit ARM® Cortex™-M3 microprocessor is highly optimized for high performance, low power consumption, and efficient memory utilization. Including an integrated MPU, it supports two different modes of operation: Privileged mode and Unprivileged mode. This architecture could be used to separate the networking stack from the application code and prevent unwanted modification of restricted areas of memory and registers resulting in increased stability and reliability of deployed solutions.

The STM32W108 has 128/192/256 Kbytes of embedded Flash memory and 8/12/16 Kbytes of integrated RAM for data and program storage. The STM32W108 HAL software employs an effective wear-leveling algorithm that optimizes the lifetime of the embedded Flash.

To maintain the strict timing requirements imposed by the ZigBee and IEEE 802.15.4-2003 standards, the STM32W108 integrates a number of MAC functions into the hardware. The MAC hardware handles automatic ACK transmission and reception, automatic backoff delay, and clear channel assessment for transmission, as well as automatic filtering of

received packets. A packet trace interface is also integrated with the MAC, allowing complete, non-intrusive capture of all packets to and from the STM32W108.

The STM32W108 offers a number of advanced power management features that enable long battery life. A high-frequency internal RC oscillator allows the processor core to begin code execution quickly upon waking. Various deep sleep modes are available with less than 1 µA power consumption while retaining RAM contents. To support user-defined applications, on-chip peripherals include UART, SPI, I^2C, ADC and general-purpose timers, as well as up to 24 GPIOs. Additionally, an integrated voltage regulator, power-on-reset circuit, and sleep timer are available.

1.1 Development tools

The STM32W108 implements both the ARM Serial Wire and JTAG debug interfaces. These interfaces provide real time, non-intrusive programming and debugging capabilities. Serial Wire and JTAG provide the same functionality, but are mutually exclusive. The Serial Wire interface uses two pins; the JTAG interface uses five. Serial Wire is preferred, since it uses fewer pins.

The STM32W108 also integrates the standard ARM system debug components: Flash Patch and Breakpoint (FPB), Data Watchpoint and Trace (DWT), and Instrumentation Trace Macrocell (DWT).

1.2 Overview

1.2.1 Functional description

The STM32W108 radio receiver is a low-IF, super-heterodyne receiver. The architecture has been chosen to optimize co-existence with other devices in the 2.4 GHz band (namely, WIFI and Bluetooth), and to minimize power consumption. The receiver uses differential signal paths to reduce sensitivity to noise interference. Following RF amplification, the signal is downconverted by an image-rejecting mixer, filtered, and then digitized by an ADC.

The radio transmitter uses an efficient architecture in which the data stream directly modulates the VCO frequency. An integrated power amplifier (PA) provides the output power. Digital logic controls Tx path and output power calibration. If the STM32W108 is to be used with an external PA, use the TX_ACTIVE or nTX_ACTIVE signal to control the timing of the external switching logic.

The integrated 4.8 GHz VCO and loop filter minimize off-chip circuitry. Only a 24 MHz crystal with its loading capacitors is required to establish the PLL local oscillator signal.

The MAC interfaces the on-chip RAM to the Rx and Tx baseband modules. The MAC provides hardware-based IEEE 802.15.4 packet-level filtering. It supplies an accurate symbol time base that minimizes the synchronization effort of the software stack and meets the protocol timing requirements. In addition, it provides timer and synchronization assistance for the IEEE 802.15.4 CSMA-CA algorithm.

The STM32W108 integrates an ARM® Cortex-M3 microprocessor, revision r1p1. This industry-leading core provides 32 bit performance and is very power efficient. It has excellent code density using the ARM® Thumb 2 instruction set. The processor can be operated at 12 MHz or 24 MHz when using the crystal oscillator, or at 6 MHz or 12 MHz when using the integrated high frequency RC oscillator.

The STM32W108 has 128/192/256 Kbytes of Flash memory, 8/12/16 Kbytes of SRAM on-chip, and the ARM configurable memory protection unit (MPU).

The STM32W108 contains 24 GPIO pins shared with other peripheral or alternate functions. Because of flexible routing within the STM32W108, external devices can use the alternate functions on a variety of different GPIOs. The integrated Serial Controller SC1 can be configured for SPI (master or slave), I^2C (master-only), or UART operation, and the Serial Controller SC2 can be configured for SPI (master or slave) or I^2C (master-only) operation.

The STM32W108 has a general purpose ADC which can sample analog signals from six GPIO pins in single-ended or differential modes. It can also sample the regulated supply VDD_PADSA, the voltage reference VREF, and GND. The ADC has two selectable voltage ranges: 0 V to 1.2 V (normal) and 0.1 V to 0.1 V below the high voltage supply (high). The ADC has a DMA mode to capture samples and automatically transfer them into RAM. The integrated voltage reference for the ADC, VREF, can be made available to external circuitry. An external voltage reference can also be driven into the ADC.

The STM32W108 contains four oscillators: a high frequency 24 MHz external crystal oscillator, a high frequency 12 MHz internal RC oscillator, an optional low frequency 32.768 kHz external crystal oscillator, and a 10 kHz internal RC oscillator.

The STM32W108 has an ultra low power, deep sleep state with a choice of clocking modes. The sleep timer can be clocked with either the external 32.768 kHz crystal oscillator or with a 1 kHz clock derived from the internal 10 kHz RC oscillator. Alternatively, all clocks can be disabled for the lowest power mode. In the lowest power mode, only external events on

GPIO pins will wake up the chip. The STM32W108 has a fast startup time (typically 100 µs) from deep sleep to the execution of the first ARM® Cortex-M3 instruction.

The STM32W108 contains three power domains. The always-on high voltage supply powers the GPIO pads and critical chip functions. Regulated low voltage supplies power the rest of the chip. The low voltage supplies are be disabled during deep sleep to reduce power consumption. Integrated voltage regulators generate regulated 1.25 V and 1.8 V voltages from an unregulated supply voltage. The 1.8 V regulator output is decoupled and routed externally to supply analog blocks, RAM, and Flash memories. The 1.25 V regulator output is decoupled externally and supplies the core logic.

The digital section of the receiver uses a coherent demodulator to generate symbols for the hardware-based MAC. The digital receiver also contains the analog radio calibration routines and controls the gain within the receiver path.

In addition to 2 general-purpose timers, the STM32W108 also contains a watchdog timer to ensure protection against software crashes and CPU lockup, a 32-bit sleep timer dedicated to system timing and waking from sleep at specific times and an ARM® standard system event timer in the NVIC.

The STM32W108 integrates hardware support for a Packet Trace module, which allows robust packet-based debug.

Note: The STM32W108 is not pin-compatible with the previous generation chip, the SN250, except for the RF section of the chip. Pins 1-11 and 45-48 are compatible, to ease migration to the STM32W108.

1.2.2 ARM® Cortex™-M3 core

The STM32W108 integrates the ARM® Cortex™-M3 microprocessor, revision r1p1, developed by ARM Ltd, making the STM32W108 a true system-on-a-chip solution. The ARM® Cortex-M3 is an advanced 32-bit modified Harvard architecture processor that has separate internal program and data buses, but presents a unified program and data address space to software. The word width is 32 bits for both the program and data sides. The ARM® Cortex-M3 allows unaligned word and half-word data accesses to support efficiently-packed data structures.

The ARM® Cortex-M3 clock speed is configurable to 6 MHz, 12 MHz, or 24 MHz. For normal operation 12 MHz is preferred over 24 MHz due to its lower power consumption. The 6 MHz operation can only be used when radio operations are not required since the radio requires an accurate 12 MHz clock.

The ARM® Cortex-M3 in the STM32W108 has also been enhanced to support two separate memory protection levels. Basic protection is available without using the MPU, but the usual operation uses the MPU. The MPU protects unimplemented areas of the memory map to prevent common software bugs from interfering with software operation. The architecture could also separate the networking stack from the application code using a fine granularity RAM protection module. Errant writes are captured and details are reported to the developer to assist in tracking down and fixing issues.

2 Documentation conventions

Table 1. Description of abbreviations used for bitfield access

Abbreviation	Description(1)
Read/Write (rw)	Software can read and write to these bits.
Read-only (r)	Software can only read these bits.
Write only (w)	Software can only write to this bit. Reading returns the reset value.
Read/Write in (MPU) Privileged mode only (rws)	Software can read and write to these bits only in Privileged mode. For more information, please refer to *RAM memory protection on page 30* and *Memory protection unit on page 30*.

1. The conditions under which the hardware (core) sets or clears this field are explained in details in the bitfield description, as well as the events that may be generated by writing to the bit.

3 Pinout and pin description

Figure 2. 48-pin VFQFPN pinout

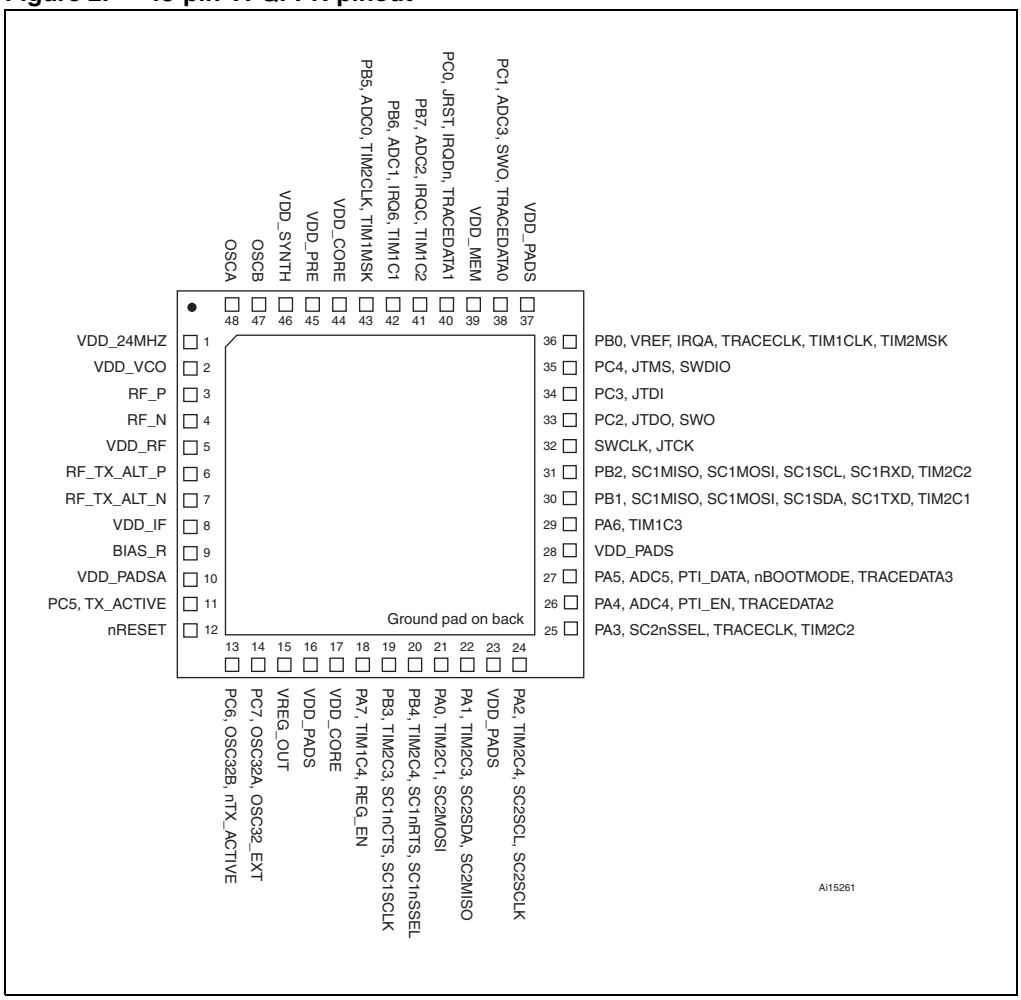


Figure 3. 40-pin VFQFPN pinout

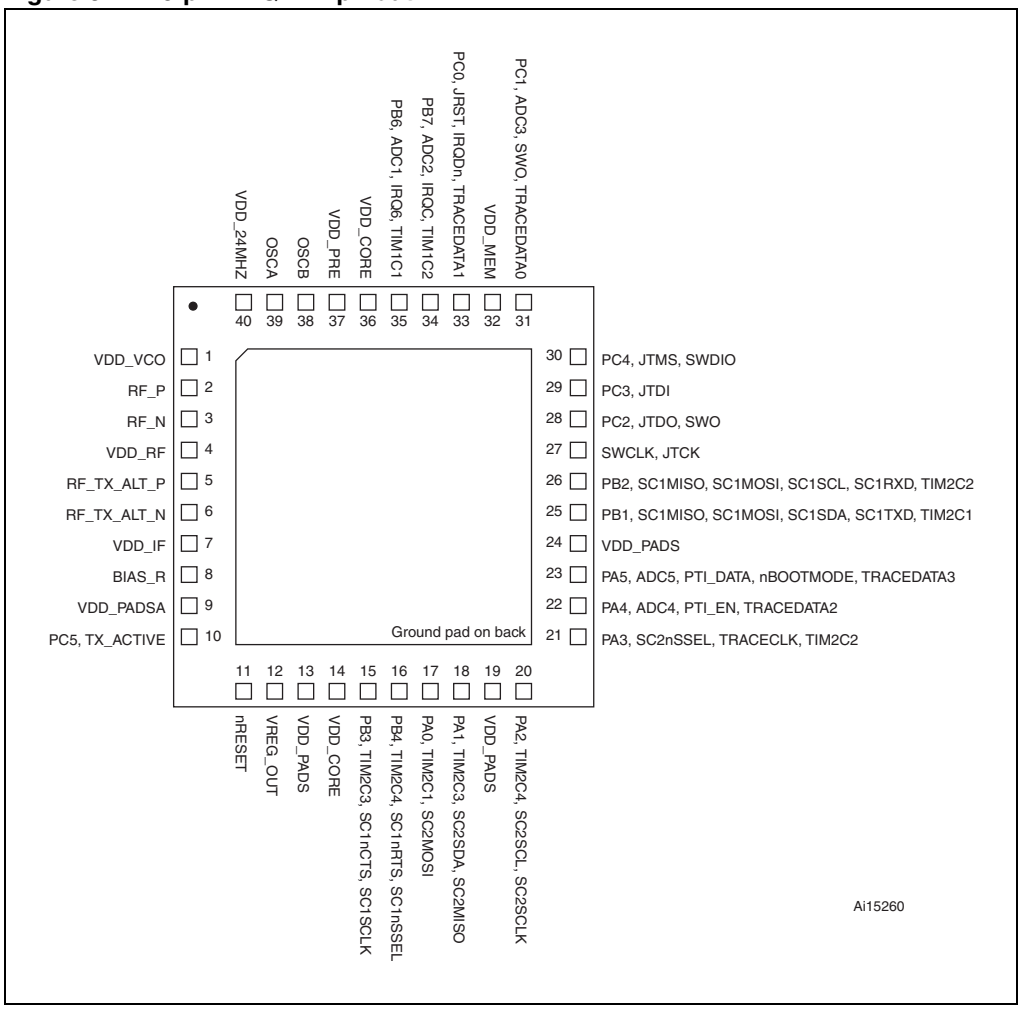


Table 2. Pin descriptions

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
1	40	VDD_24MHZ	Power	1.8V high-frequency oscillator supply
2	1	VDD_VCO	Power	1.8V VCO supply
3	2	RF_P	I/O	Differential (with RF_N) receiver input/transmitter output
4	3	RF_N	I/O	Differential (with RF_P) receiver input/transmitter output
5	4	VDD_RF	Power	1.8V RF supply (LNA and PA)
6	5	RF_TX_ALT_P	O	Differential (with RF_TX_ALT_N) transmitter output (optional)
7	6	RF_TX_ALT_N	O	Differential (with RF_TX_ALT_P) transmitter output (optional)
8	7	VDD_IF	Power	1.8V IF supply (mixers and filters)

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
9	8	BIAS_R	I	Bias setting resistor
10	9	VDD_PADSA	Power	Analog pad supply (1.8V)
11	10	PC5	I/O	Digital I/O
		TX_ACTIVE	O	Logic-level control for external Rx/Tx switch. The STM32W108 baseband controls TX_ACTIVE and drives it high (VDD_PADS) when in Tx mode. Select alternate output function with GPIO_PCCFGH[7:4]
12	11	nRESET	I	Active low chip reset (internal pull-up)
13		PC6	I/O	Digital I/O
		OSC32B	I/O	32.768 kHz crystal oscillator Select analog function with GPIO_PCCFGH[11:8]
		nTX_ACTIVE	O	Inverted TX_ACTIVE signal (see PC5) Select alternate output function with GPIO_PCCFGH[11:8]
14		PC7	I/O	Digital I/O
		OSC32A	I/O	32.768 kHz crystal oscillator. Select analog function with GPIO_PCCFGH[15:12]
		OSC32_EXT	I	Digital 32 kHz clock input source
15	12	VREG_OUT	Power	Regulator output (1.8 V while awake, 0 V during deep sleep)
16	13	VDD_PADS	Power	Pads supply (2.1-3.6 V)
17	14	VDD_CORE	Power	1.25 V digital core supply decoupling
18		PA7	I/O High current	Digital I/O. Disable REG_EN with GPIO_DBGCFG[4]
		TIM1_CH4	O	Timer 1 Channel 4 output Enable timer output with TIM1_CCER Select alternate output function with GPIO_PACFGH[15:12] Disable REG_EN with GPIO_DBGCFG[4]
			I	Timer 1 Channel 4 input. (Cannot be remapped.)
		REG_EN	O	External regulator open drain output. (Enabled after reset.)

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
19	15	PB3	I/O	Digital I/O
		TIM2_CH3 (see Pin 22)	O	Timer 2 channel 3 output Enable remap with TIM2_OR[6] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PBCFGL[15:12]
			I	Timer 2 channel 3 input. Enable remap with TIM2_OR[6].
		UART_CTS	I	UART CTS handshake of Serial Controller 1 Enable with SC1_UARTCFG[5] Select UART with SC1_MODE
		SC1SCLK	O	SPI master clock of Serial Controller 1 Either disable timer output in TIM2_CCER or disable remap with TIM2_OR[6] Enable master with SC1_SPICFG[4] Select SPI with SC1_MODE Select alternate output function with GPIO_PBCFGL[15:12]
			I	SPI slave clock of Serial Controller 1 Enable slave with SC1_SPICFG[4] Select SPI with SC1_MODE
20	16	PB4	I/O	Digital I/O
		TIM2_CH4 (see also Pin 24)	O	Timer 2 channel 4 output Enable remap with TIM2_OR[7] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PBCFGH[3:0]
			I	Timer 2 channel 4 input. Enable remap with TIM2_OR[7].
		UART_RTS	O	UART RTS handshake of Serial Controller 1 Either disable timer output in TIM2_CCER or disable remap with TIM2_OR[7] Enable with SC1_UARTCFG[5] Select UART with SC1_MODE Select alternate output function with GPIO_PBCFGH[3:0]
		SC1nSSEL	I	SPI slave select of Serial Controller 1 Enable slave with SC1_SPICFG[4] Select SPI with SC1_MODE

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
21	17	PA0	I/O	Digital I/O
		TIM2_CH1 (see also Pin 30)	O	Timer 2 channel 1 output Disable remap with TIM2_OR[4] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PACFGL[3:0]
			I	Timer 2 channel 1 input. Disable remap with TIM2_OR[4].
		SC2MOSI	O	SPI master data out of Serial Controller 2 Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[4] Enable master with SC2_SPICFG[4] Select SPI with SC2_MODE Select alternate output function with GPIO_PACFGL[3:0]
			I	SPI slave data in of Serial Controller 2 Enable slave with SC2_SPICFG[4] Select SPI with SC2_MODE
22	18	PA1	I/O	Digital I/O
		TIM2_CH3 (see also Pin 19)	O	Timer 2 channel 3 output Disable remap with TIM2_OR[6] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PACFGL[7:4]
			I	Timer 2 channel 3 input. Disable remap with TIM2_OR[6].
		SC2SDA	I/O	I^2C data of Serial Controller 2 Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[6] Select I^2C with SC2_MODE Select alternate open-drain output function with GPIO_PACFGL[7:4]
		SC2MISO	O	SPI slave data out of Serial Controller 2 Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[6] Enable slave with SC2_SPICFG[4] Select SPI with SC2_MODE Select alternate output function with GPIO_PACFGL[7:4]
			I	SPI master data in of Serial Controller 2 Enable slave with SC2_SPICFG[4] Select SPI with SC2_MODE
23	19	VDD_PADS	Power	Pads supply (2.1-3.6V)

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
24	20	PA2	I/O	Digital I/O
		TIM2_CH4 (see also Pin 20)	O	Timer 2 channel 4 output Disable remap with TIM2_OR[7] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PACFGL[11:8]
			I	Timer 2 channel 4 input. Disable remap with TIM2_OR[7].
		SC2SCL	I/O	I^2C clock of Serial Controller 2 Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[7] Select I^2C with SC2_MODE Select alternate open-drain output function with GPIO_PACFGL[11:8]
		SC2SCLK	O	SPI master clock of Serial Controller 2 Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[7] Enable master with SC2_SPICFG[4] Select SPI with SC2_MODE Select alternate output function with GPIO_PACFGL[11:8]
			I	SPI slave clock of Serial Controller 2 Enable slave with SC2_SPICFG[4] Select SPI with SC2_MODE
25	21	PA3	I/O	Digital I/O
		SC2nSSEL	I	SPI slave select of Serial Controller 2 Enable slave with SC2_SPICFG[4] Select SPI with SC2_MODE
		TRACECLK (see also Pin 36)	O	Synchronous CPU trace clock Either disable timer output in TIM2_CCER or enable remap with TIM2_OR[5] Enable trace interface in ARM core Select alternate output function with GPIO_PACFGL[15:12]
		TIM2_CH2 (see also Pin 31)	O	Timer 2 channel 2 output Disable remap with TIM2_OR[5] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PACFGL[15:12]
			I	Timer 2 channel 2 input. Disable remap with TIM2_OR[5].

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
26	22	PA4	I/O	Digital I/O
		ADC4	Analog	ADC Input 4. Select analog function with GPIO_PACFGH[3:0].
		PTI_EN	O	Frame signal of Packet Trace Interface (PTI). Disable trace interface in ARM core. Select alternate output function with GPIO_PACFGH[3:0].
		TRACEDATA2	O	Synchronous CPU trace data bit 2. Select 4-wire synchronous trace interface in ARM core. Enable trace interface in ARM core. Select alternate output function with GPIO_PACFGH[3:0].
27	23	PA5	I/O	Digital I/O
		ADC5	Analog	ADC Input 5. Select analog function with GPIO_PACFGH[7:4].
		PTI_DATA	O	Data signal of Packet Trace Interface (PTI). Disable trace interface in ARM core. Select alternate output function with GPIO_PACFGH[7:4].
		nBOOTMODE	I	Embedded serial bootloader activation out of reset. Signal is active during and immediately after a reset on NRST. See *Section 6.2: Resets on page 37* for details.
		TRACEDATA3	O	Synchronous CPU trace data bit 3. Select 4-wire synchronous trace interface in ARM core. Enable trace interface in ARM core. Select alternate output function with GPIO_PACFGH[7:4]
28	24	VDD_PADS	Power	Pads supply (2.1-3.6 V)
29		PA6	I/O High current	Digital I/O
		TIM1_CH3	O	Timer 1 channel 3 output Enable timer output in TIM1_CCER Select alternate output function with GPIO_PACFGH[11:8]
			I	Timer 1 channel 3 input (Cannot be remapped.)

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
30	25	PB1	I/O	Digital I/O
		SC1MISO	O	SPI slave data out of Serial Controller 1 Either disable timer output in TIM2_CCER or disable remap with TIM2_OR[4] Select SPI with SC1_MODE Select slave with SC1_SPICR Select alternate output function with GPIO_PBCFGL[7:4]
		SC1MOSI	O	SPI master data out of Serial Controller 1 Either disable timer output in TIM2_CCER or disable remap with TIM2_OR[4] Select SPI with SC1_MODE Select master with SC1_SPICR Select alternate output function with GPIO_PBCFGL[7:4]
		SC1SDA	I/O	I^2C data of Serial Controller 1 Either disable timer output in TIM2_CCER, or disable remap with TIM2_OR[4] Select I^2C with SC1_MODE Select alternate open-drain output function with GPIO_PBCFGL[7:4]
		SC1TXD	O	UART transmit data of Serial Controller 1 Either disable timer output in TIM2_CCER or disable remap with TIM2_OR[4] Select UART with SC1_MODE Select alternate output function with GPIO_PBCFGL[7:4]
		TIM2_CH1 (see also Pin 21)	O	Timer 2 channel 1 output Enable remap with TIM2_OR[4] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PACFGL[7:4]
			I	Timer 2 channel 1 input. Disable remap with TIM2_OR[4].

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
31	26	PB2	I/O	Digital I/O
		SC1MISO	I	SPI master data in of Serial Controller 1 Select SPI with SC1_MODE Select master with SC1_SPICR
		SC1MOSI	I	SPI slave data in of Serial Controller 1 Select SPI with SC1_MODE Select slave with SC1_SPICR
		SC1SCL	I/O	I^2C clock of Serial Controller 1 Either disable timer output in TIM2_CCER, or disable remap with TIM2_OR[5] Select I^2C with SC1_MODE Select alternate open-drain output function with GPIO_PBCFGL[11:8]
		SC1RXD	I	UART receive data of Serial Controller 1 Select UART with SC1_MODE
		TIM2_CH2 (see also Pin 25)	O	Timer 2 channel 2 output Enable remap with TIM2_OR[5] Enable timer output in TIM2_CCER Select alternate output function with GPIO_PBCFGL[11:8]
			I	Timer 2 channel 2 input. Enable remap with TIM2_OR[5].
32	27	SWCLK	I/O	Serial Wire clock input/output with debugger Selected when in Serial Wire mode (see JTMS description, Pin 35)
		JTCK	I	JTAG clock input from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-down is enabled
33	28	PC2	I/O	Digital I/O Enable with GPIO_DBGCFG[5]
		JTDO	O	JTAG data out to debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35)
		SWO	O	Serial Wire Output asynchronous trace output to debugger Select asynchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[11:8] Enable Serial Wire mode (see JTMS description, Pin 35) Internal pull-up is enabled

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
34	29	PC3	I/O	Digital I/O Either Enable with GPIO_DBGCFG[5], or enable Serial Wire mode (see JTMS description)
		JTDI	I	JTAG data in from debugger Selected when in JTAG mode (default mode, see JTMS description, Pin 35) Internal pull-up is enabled
35	30	PC4	I/O	Digital I/O Enable with GPIO_DBGCFG[5]
		JTMS	I	JTAG mode select from debugger Selected when in JTAG mode (default mode) JTAG mode is enabled after power-up or by forcing NRST low Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
		SWDIO	I/O	Serial Wire bidirectional data to/from debugger Enable Serial Wire mode (see JTMS description) Select Serial Wire mode using the ARM-defined protocol through a debugger Internal pull-up is enabled
36		PB0	I/O	Digital I/O
		VREF	Analog O	ADC reference output. Enable analog function with GPIO_PBCFGL[3:0].
		VREF	Analog I	ADC reference input. Enable analog function with GPIO_PBCFGL[3:0]. Enable reference output with an ST system function.
		IRQA	I	External interrupt source A.
		TRACECLK (see also Pin 25)	O	Synchronous CPU trace clock. Enable trace interface in ARM core. Select alternate output function with GPIO_PBCFGL[3:0].
		TIM1CLK	I	Timer 1 external clock input.
		TIM2MSK	I	Timer 2 external clock mask input.
37		VDD_PADS	Power	Pads supply (2.1 to 3.6 V).

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
38	31	PC1	I/O	Digital I/O
		ADC3	Analog	ADC Input 3 Enable analog function with GPIO_PCCFGL[7:4]
		SWO (see also Pin 33)	O	Serial Wire Output asynchronous trace output to debugger Select asynchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[7:4]
		TRACEDATA0	O	Synchronous CPU trace data bit 0 Select 1-, 2- or 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[7:4]
39	32	VDD_MEM	Power	1.8 V supply (flash, RAM)
40	33	PC0	I/O High current	Digital I/O Either enable with GPIO_DBGCFG[5], or enable Serial Wire mode (see JTMS description, Pin 35) and disable TRACEDATA1
		JRST	I	JTAG reset input from debugger Selected when in JTAG mode (default mode, see JTMS description) and TRACEDATA1 is disabled Internal pull-up is enabled
		IRQD (1)	I	Default external interrupt source D
		TRACEDATA1	O	Synchronous CPU trace data bit 1 Select 2- or 4-wire synchronous trace interface in ARM core Enable trace interface in ARM core Select alternate output function with GPIO_PCCFGL[3:0]
41	34	PB7	I/O High current	Digital I/O
		ADC2	Analog	ADC Input 2 Enable analog function with GPIO_PBCFGH[15:12]
		IRQC (1)	I	Default external interrupt source C
		TIM1_CH2	O	Timer 1 channel 2 output Enable timer output in TIM1_CCER Select alternate output function with GPIO_PBCFGH[15:12]
			I	Timer 1 channel 2 input (Cannot be remapped)

Table 2. Pin descriptions (continued)

48-Pin Package Pin no.	40-Pin Package Pin no.	Signal	Direction	Description
42	35	PB6	I/O High current	Digital I/O
		ADC1	Analog	ADC Input 1 Enable analog function with GPIO_PBCFGH[11:8]
		IRQB	I	External interrupt source B
		TIM1_CH1	O	Timer 1 channel 1 output Enable timer output in TIM1_CCER Select alternate output function with GPIO_PBCFGH[11:8]
			I	Timer 1 channel 1 input (Cannot be remapped)
43		PB5	I/O	Digital I/O
		ADC0	Analog	ADC Input 0 Enable analog function with GPIO_PBCFGH[7:4]
		TIM2CLK	I	Timer 2 external clock input
		TIM1MSK	I	Timer 2 external clock mask input
44	36	VDD_CORE	Power	1.25 V digital core supply decoupling
45	37	VDD_PRE	Power	1.8 V prescaler supply
46		VDD_SYNTH	Power	1.8 V synthesizer supply
47	38	OSCB	I/O	24 MHz crystal oscillator or left open when using external clock input on OSCA
48	39	OSCA	I/O	24 MHz crystal oscillator or external clock input
49	41	GND	Ground	Ground supply pad in the bottom center of the package.

1. IRQC and IRQD external interrupts can be mapped to any digital I/O pin using the GPIO_IRQCSEL and GPIO_IRQDSEL registers.

4 Embedded memory

Figure 4. STM32W108xB memory mapping

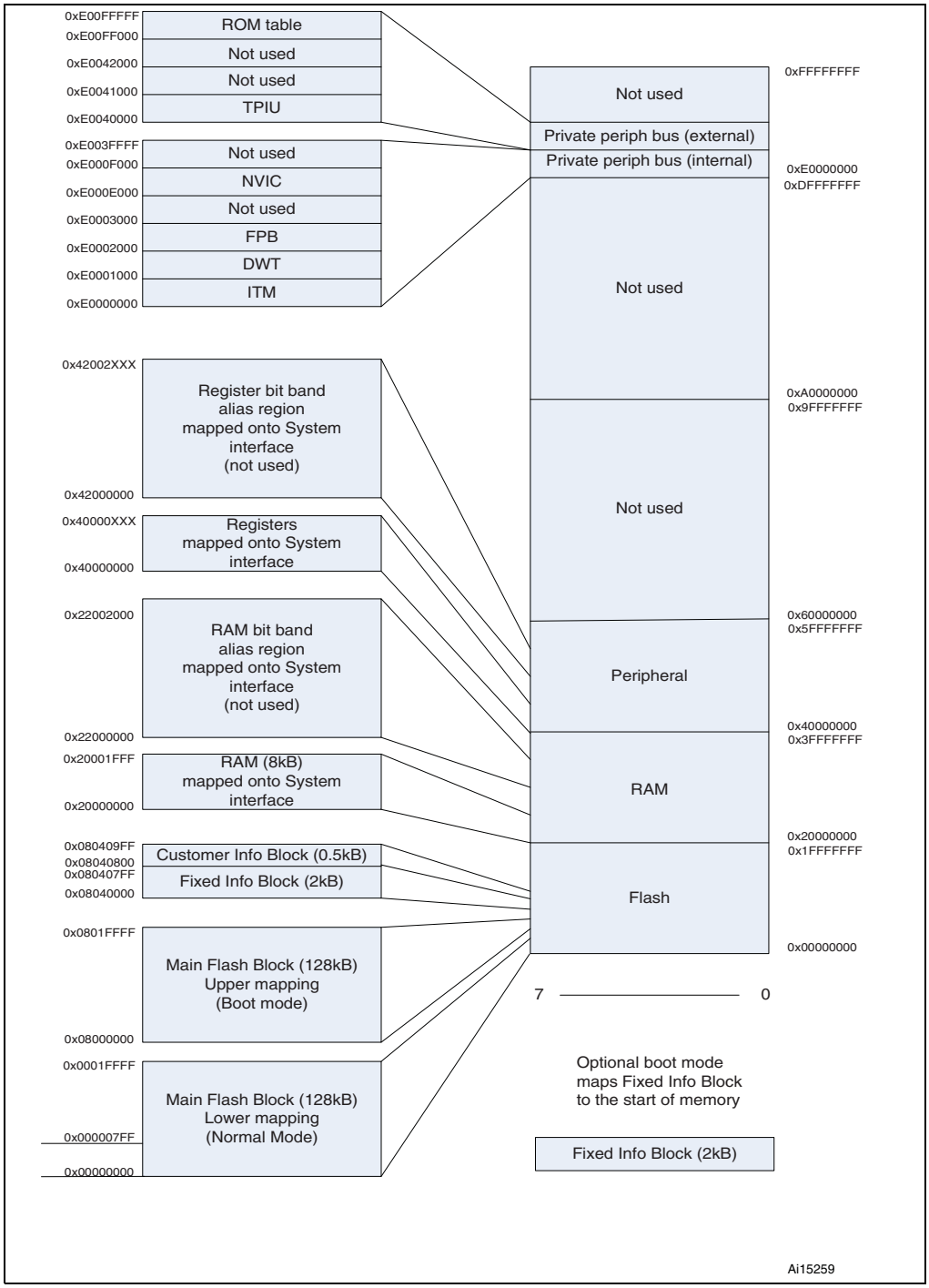
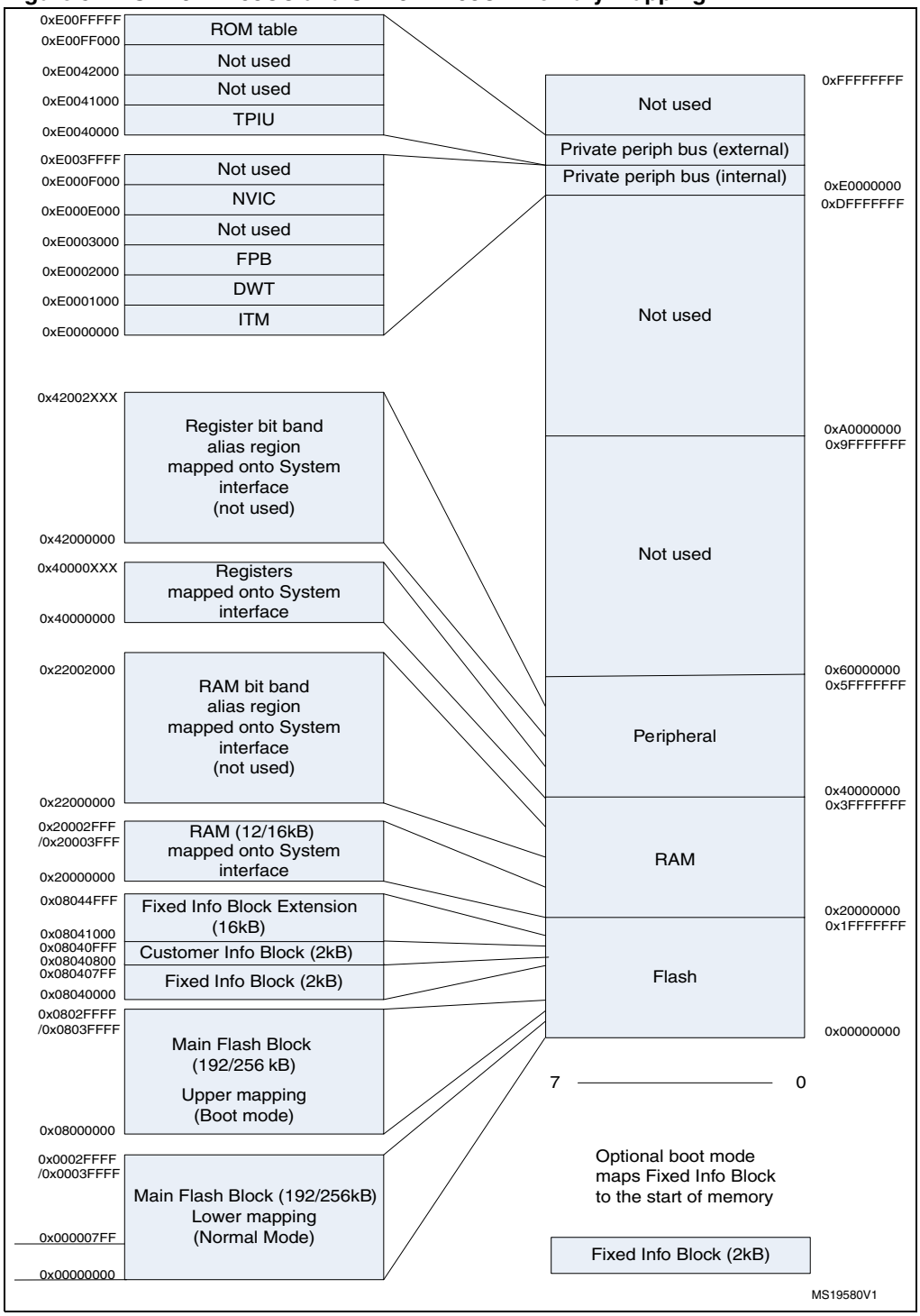


Figure 5. STM32W108CC and STM32W108CZ memory mapping



4.1 Flash memory

The STM32W108 provides Flash memory in four separate blocks as follows:

- Main Flash Block (MFB)
- Fixed Information Block (FIB)
- Fixed Information Block Extension (FIB-EXT)
- Customer Information Block (CIB)

The size of these blocks and associated page size is described in *Table 3*.

Table 3. Flash memory

	STM32W108xB		STM32W108CC		STM32W108CZ		Unit
	Size	Page size	Size	Page size	Size	Page size	
MFB	128	1	256	2	192	2	K Bytes
FIB	2	2	2	2	2	2	K Bytes
CIB	0.5	0.5	2	2	2	2	K Bytes
FIB-EXT	0	N/A	16	2	16	2	K Bytes
Total	130.5		276		212		K Bytes

The smallest erasable unit is one page and the smallest writable unit is an aligned 16-bit half-word. The flash is rated to have a guaranteed 1,000 write/erase cycles. The flash cell has been qualified for a data retention time of >100 years at room temperature.

Flash may be programmed either through the Serial Wire/JTAG interface or through bootloader software. Programming flash through Serial Wire/JTAG requires the assistance of RAM-based utility code. Programming through a bootloader requires specific software for over-the-air loading or serial link loading. A simplified, serial-link-only bootloader is also available preprogrammed into the FIB.

4.2 Random-access memory

The STM32W108 has 8/12/16 Kbytes of static RAM on-chip. The start of RAM is mapped to address 0x20000000. Although the ARM® Cortex-M3 allows bit band accesses to this address region, the standard MPU configuration does not permit use of the bit-band feature.

The RAM is physically connected to the AHB System bus and is therefore accessible to both the ARM® Cortex-M3 microprocessor and the debugger. The RAM can be accessed for both instruction and data fetches as bytes, half words, or words. The standard MPU configuration does not permit execution from the RAM, but for special purposes, such as programming the main flash block, the MPU may be disabled. To the bus, the RAM appears as 32-bit wide memory and in most situations has zero wait state read or write access. In the higher CPU clock mode the RAM requires two wait states. This is handled by hardware transparent to the user application with no configuration required.

4.2.1 Direct memory access (DMA) to RAM

Several of the peripherals are equipped with DMA controllers allowing them to transfer data into and out of RAM autonomously. This applies to the radio (802.15.4 MAC), general purpose ADC, and both serial controllers. In the case of the serial controllers, the DMA is full duplex so that a read and a write to RAM may be requested at the same time. Thus there are six DMA channels in total.

The STM32W108 integrates a DMA arbiter that ensures fair access to the microprocessor as well as the peripherals through a fixed priority scheme appropriate to the memory bandwidth requirements of each master. The priority scheme is as follows, with the top peripheral being the highest priority:

1. General Purpose ADC
2. Serial Controller 2 Receive
3. Serial Controller 2 Transmit
4. MAC
5. Serial Controller 1 Receive
6. Serial Controller 1 Transmit

4.2.2 RAM memory protection

The STM32W108 integrates two memory protection mechanisms. The first memory protection mechanism is through the ARM® Cortex-M3 Memory Protection Unit (MPU) described in the Memory Protection Unit section. The MPU may be used to protect any area of memory. MPU configuration is normally handled by software. The second memory protection mechanism is through a fine granularity RAM protection module. This allows segmentation of the RAM into blocks where any block can be marked as write protected. An attempt to write to a protected RAM block using a user mode write results in a bus error being signaled on the AHB System bus. A system mode write is allowed at any time and reads are allowed in either mode. The main purpose of this fine granularity RAM protection module is to notify the stack of erroneous writes to system areas of memory. RAM protection is configured using a group of registers that provide a bit map. Each bit in the map represents a 32-byte block of RAM for STM32W108xB and 64 bytes of RAM for STM32W108CC and STM32W108CZ. When the bit is set the block is write protected.

The fine granularity RAM memory protection mechanism is also available to the peripheral DMA controllers. A register bit is provided to enable the memory protection to include DMA writes to protected memory. If a DMA write is made to a protected location in RAM, a management interrupt is generated. At the same time the faulting address and the identification of the peripheral is captured for later debugging. Note that only peripherals capable of writing data to RAM, such as received packet data or a received serial port character, can generate this interrupt.

4.3 Memory protection unit

The STM32W108 includes the ARM® Cortex-M3 Memory Protection Unit, or MPU. The MPU controls access rights and characteristics of up to eight address regions, each of which may be divided into eight equal sub-regions. Refer to the ARM® Cortex-M3 Technical Reference Manual (DDI 0337A) for a detailed description of the MPU.

ST software configures the MPU in a standard configuration and application software should not modify it. The configuration is designed for optimal detection of illegal instruction or data

accesses. If an illegal access is attempted, the MPU captures information about the access type, the address being accessed, and the location of the offending software. This simplifies software debugging and increases the reliability of deployed devices. As a consequence of this MPU configuration, accessing RAM and register bit-band address alias regions is not permitted, and generates a bus fault if attempted.

5 Radio frequency module

The radio module consists of an analog front end and digital baseband as shown in *Figure 1: STM32W108 block diagram*.

5.1 Receive (Rx) path

The Rx path uses a low-IF, super-heterodyne receiver that rejects the image frequency using complex mixing and polyphase filtering. In the analog domain, the input RF signal from the antenna is first amplified and mixed down to a 4 MHz IF frequency. The mixers' output is filtered, combined, and amplified before being sampled by a 12 Msps ADC. The digitized signal is then demodulated in the digital baseband. The filtering within the Rx path improves the STM32W108's co-existence with other 2.4 GHz transceivers such as IEEE 802.15.4, IEEE 802.11g, and Bluetooth radios. The digital baseband also provides gain control of the Rx path, both to enable the reception of small and large wanted signals and to tolerate large interferers.

5.1.1 Rx baseband

The STM32W108 Rx digital baseband implements a coherent demodulator for optimal performance. The baseband demodulates the O-QPSK signal at the chip level and synchronizes with the IEEE 802.15.4-defined preamble. An automatic gain control (AGC) module adjusts the analog gain continuously every ¼ symbol until the preamble is detected. Once detected, the gain is fixed for the remainder of the packet. The baseband despreads the demodulated data into 4-bit symbols. These symbols are buffered and passed to the hardware-based MAC module for packet assembly and filtering.

In addition, the Rx baseband provides the calibration and control interface to the analog Rx modules, including the LNA, Rx baseband filter, and modulation modules. The ST RF software driver includes calibration algorithms that use this interface to reduce the effects of silicon process and temperature variation.

5.1.2 RSSI and CCA

The STM32W108 calculates the RSSI over every 8-symbol period as well as at the end of a received packet. The linear range of RSSI is specified to be at least 40 dB over temperature. At room temperature, the linear range is approximately 60 dB (-90 dBm to -30 dBm input signal).

The STM32W108 Rx baseband provides support for the IEEE 802.15.4-2003 RSSI CCA method, Clear channel reports busy medium if RSSI exceeds its threshold.

5.2 Transmit (Tx) path

The STM32W108 Tx path produces an O-QPSK-modulated signal using the analog front end and digital baseband. The area- and power-efficient Tx architecture uses a two-point modulation scheme to modulate the RF signal generated by the synthesizer. The modulated RF signal is fed to the integrated PA and then out of the STM32W108.

5.2.1 Tx baseband

The STM32W108 Tx baseband in the digital domain spreads the 4-bit symbol into its IEEE 802.15.4-2003-defined 32-chip sequence. It also provides the interface for software to calibrate the Tx module to reduce silicon process, temperature, and voltage variations.

5.2.2 TX_ACTIVE and nTX_ACTIVE signals

For applications requiring an external PA, two signals are provided called TX_ACTIVE and nTX_ACTIVE. These signals are the inverse of each other. They can be used for external PA power management and RF switching logic. In transmit mode the Tx baseband drives TX_ACTIVE high, as described in *Table 26: GPIO signal assignments on page 65*. In receive mode the TX_ACTIVE signal is low. TX_ACTIVE is the alternate function of PC5, and nTX_ACTIVE is the alternate function of PC6. See *Section 8: General-purpose input/outputs on page 58* for details of the alternate GPIO functions.

5.3 Calibration

The ST RF software driver calibrates the radio using dedicated hardware resources.

5.4 Integrated MAC module

The STM32W108 integrates most of the IEEE 802.15.4 MAC requirements in hardware. This allows the ARM® Cortex-M3 CPU to provide greater bandwidth to application and network operations. In addition, the hardware acts as a first-line filter for unwanted packets. The STM32W108 MAC uses a DMA interface to RAM to further reduce the overall ARM® Cortex-M3 CPU interaction when transmitting or receiving packets.

When a packet is ready for transmission, the software configures the Tx MAC DMA by indicating the packet buffer RAM location. The MAC waits for the backoff period, then switches the baseband to Tx mode and performs channel assessment. When the channel is clear the MAC reads data from the RAM buffer, calculates the CRC, and provides 4-bit symbols to the baseband. When the final byte has been read and sent to the baseband, the CRC remainder is read and transmitted.

The MAC is in Rx mode most of the time. In Rx mode various format and address filters keep unwanted packets from using excessive RAM buffers, and prevent the CPU from being unnecessarily interrupted. When the reception of a packet begins, the MAC reads 4-bit symbols from the baseband and calculates the CRC. It then assembles the received data for storage in a RAM buffer. Rx MAC DMA provides direct access to RAM. Once the packet has been received additional data, which provides statistical information on the packet to the software stack, is appended to the end of the packet in the RAM buffer space.

The primary features of the MAC are:

- CRC generation, appending, and checking
- Hardware timers and interrupts to achieve the MAC symbol timing
- Automatic preamble and SFD pre-pending on Tx packets
- Address recognition and packet filtering on Rx packets
- Automatic acknowledgement transmission
- Automatic transmission of packets from memory
- Automatic transmission after backoff time if channel is clear (CCA)
- Automatic acknowledgement checking
- Time stamping received and transmitted messages
- Attaching packet information to received packets (LQI, RSSI, gain, time stamp, and packet status)
- IEEE 802.15.4 timing and slotted/unslotted timing

5.5 Packet trace interface (PTI)

The STM32W108 integrates a true PHY-level PTI for effective network-level debugging. It monitors all the PHY Tx and Rx packets between the MAC and baseband modules without affecting their normal operation. It cannot be used to inject packets into the PHY/MAC interface. This 500 kbps asynchronous interface comprises the frame signal (PTI_EN, PA4) and the data signal (PTI_DATA, PA5).

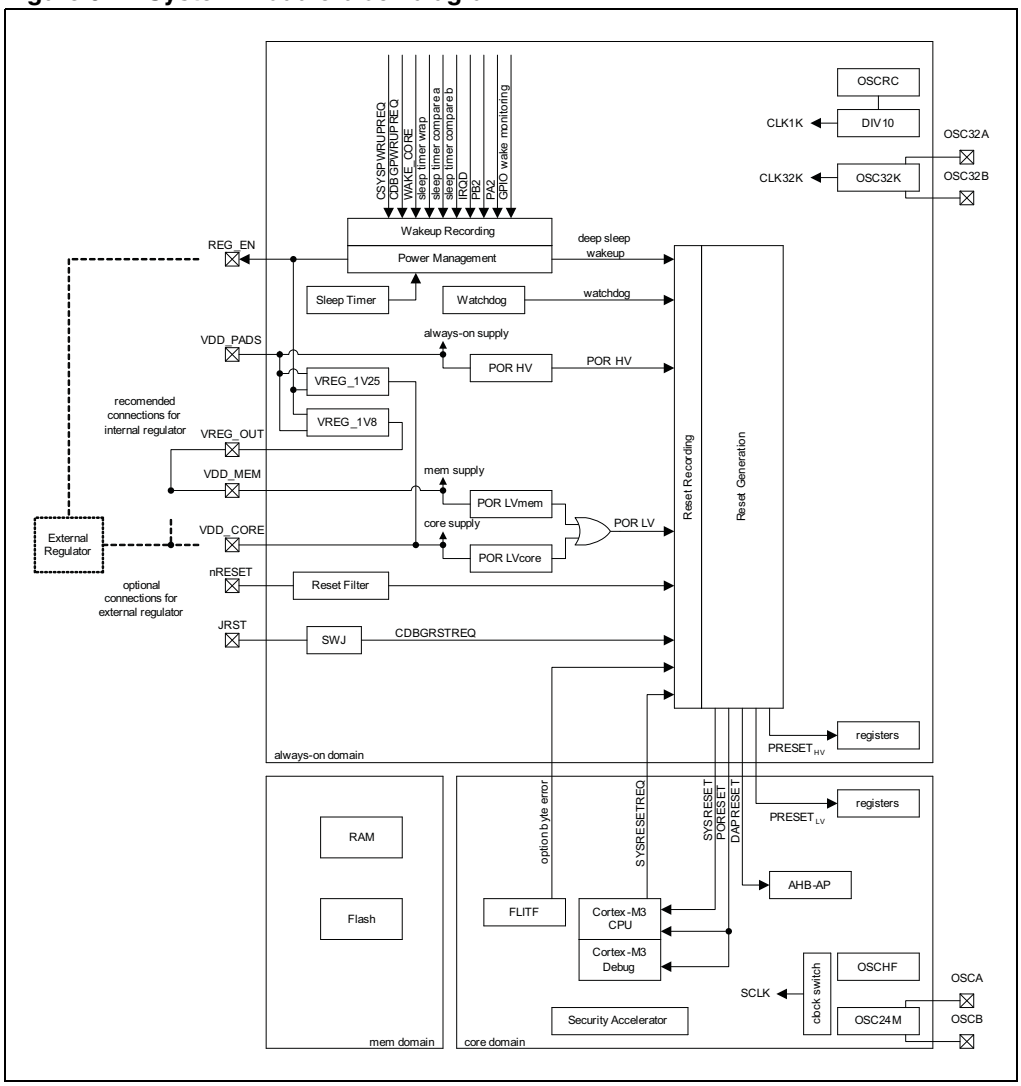
5.6 Random number generator

Thermal noise in the analog circuitry is digitized to provide entropy for a true random number generator (TRNG). The TRNG produces 16-bit uniformly distributed numbers. The Software can use the TRNG to seed a pseudo random number generator (PNRG). The TRNG is also used directly for cryptographic key generation.

6 System modules

System modules encompass power, resets, clocks, system timers, power management, and encryption. *Figure 6* shows these modules and how they interact.

Figure 6. System module block diagram



6.1 Power domains

The STM32W108 contains three power domains:

- An "always on domain" containing all logic and analog cells required to manage the STM32W108's power modes, including the GPIO controller and sleep timer. This domain must remain powered.
- A "core domain" containing the CPU, Nested Vectored Interrupt Controller (NVIC), and peripherals. To save power, this domain can be powered down using a mode called deep sleep.
- A "memory domain" containing the RAM and flash memories. This domain is managed by the power management controller. When in deep sleep, the RAM portion of this domain is powered from the always-on domain supply to retain the RAM contents while the regulators are disabled. During deep sleep the flash portion is completely powered down.

6.1.1 Internally regulated power

The preferred and recommended power configuration is to use the internal regulated power supplies to provide power to the core and memory domains. The internal regulators (VREG_1V25 and VREG_1V8) generate nominal 1.25 V and 1.8 V supplies. The 1.25 V supply is internally routed to the core domain and to an external pin. The 1.8 V supply is routed to an external pin where it can be externally routed back into the chip to supply the memory domain. The internal regulators are described in *Section 7: Integrated voltage regulator on page 56*.

When using the internal regulators, the always-on domain must be powered between 2.1 V and 3.6 V at all four VDD_PADS pins.

When using the internal regulators, the VREG_1V8 regulator output pin (VREG_OUT) must be connected to the VDD_MEM, VDD_PADSA, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE, and VDD_SYNTH pins.

When using the internal regulators, the VREG_1V25 regulator output and supply requires a connection between both VDD_CORE pins.

6.1.2 Externally regulated power

Optionally, the on-chip regulators may be left unused, and the core and memory domains may instead be powered from external supplies. For simplicity, the voltage for the core domain can be raised to nominal 1.8 V, requiring only one external regulator. Note that if the core domain is powered at a higher voltage (1.8 V instead of 1.25 V) then power consumption increases. A regulator enable signal, REG_EN, is provided for control of external regulators. This is an open-drain signal that requires an external pull-up resistor. If REG_EN is not required to control external regulators it can be disabled (see *Section 8.1.3: Forced functions on page 60*).

Using an external regulator requires the always-on domain to be powered between 1.8 V and 3.6 V at all four VDD_PADS pins.

When using an external regulator, the VREG_1V8 regulator output pin (VREG_OUT) must be left unconnected.

When using an external regulator, this external nominal 1.8 V supply has to be connected to both VDD_CORE pins and to the VDD_MEM, VDD_PADSA, VDD_VCO, VDD_RF, VDD_IF, VDD_PRE and VDD_SYNTH pins.

6.2 Resets

The STM32W108 resets are generated from a number of sources. Each of these reset sources feeds into central reset detection logic that causes various parts of the system to be reset depending on the state of the system and the nature of the reset event.

6.2.1 Reset sources

For power-on reset (POR HV and POR LV) thresholds, see *Section 14.3.2: Operating conditions at power-up on page 195*.

Watchdog reset

The STM32W108 contains a watchdog timer (see also the Watchdog Timer section) that is clocked by the internal 1 kHz timing reference. When the timer expires it generates the reset source WATCHDOG_RESET to the Reset Generation module.

Software reset

The ARM® Cortex-M3 CPU can initiate a reset under software control. This is indicated with the reset source SYSRESETREQ to the Reset Generation module.

Note: When using certain external debuggers, the chip may lock up require a pin reset or power cycle if the debugger asserts SYSRESETREQ. It is recommended not to write to the SCS_AIRCR register directly from application code. The ST software provides a reset function that should be used instead. This reset function ensures that the chip is in a safe clock mode prior to triggering the reset.

Option byte error

The flash memory controller contains a state machine that reads configuration information from the information blocks in the Flash at system start time. An error check is performed on the option bytes that are read from Flash and, if the check fails, an error is signaled that provides the reset source OPT_BYTE_ERROR to the Reset Generation module.

If an option byte error is detected, the system restarts and the read and check process is repeated. If the error is detected again the process is repeated but stops on the 3rd failure. The system is then placed into an emulated deep sleep where recovery is possible. In this state, Flash memory readout protection is forced active to prevent secure applications from being compromised.

Debug reset

The Serial Wire/JTAG Interface (SWJ) provides access to the SWJ Debug Port (SWJ-DP) registers. By setting the register bit CDBGRSTREQ in the SWJ-DP, the reset source CDBGRSTREQ is provided to the Reset Generation module.

JTAG reset

One of the STM32W108's pins can function as the JTAG reset, conforming to the requirements of the JTAG standard. This input acts independently of all other reset sources and, when asserted, does not reset any on-chip hardware except for the JTAG TAP. If the STM32W108 is in the Serial Wire mode or if the SWJ is disabled, this input has no effect.

Deep sleep reset

The Power Management module informs the Reset Generation module of entry into and exit from the deep sleep states. The deep sleep reset is applied in the following states: before entry into deep sleep, while removing power from the memory and core domain, while in deep sleep, while waking from deep sleep, and while reapplying power until reliable power levels have been detect by POR LV.

The Power Management module allows a special emulated deep sleep state that retains memory and core domain power while in deep sleep.

6.2.2 Reset recording

The STM32W108 records the last reset condition that generated a restart to the system. The reset conditions recorded are:

- POWER_HV — Always-on domain power supply failure
- POWER_LV — Core or memory domain power supply failure
- RSTB — NRST pin asserted
- W_DOG — Watchdog timer expired
- SW_RST — Software reset by SYSERSETREQ from ARM® Cortex-M3 CPU
- WAKE_UP_DSLEEP — Wake-up from deep sleep
- OPT_BYTE_FAIL — Error check failed when reading option bytes from Flash memory

The *Reset event source register (RESET_EVENT)* is used to read back the last reset event. All bits are mutually exclusive except the OPT_BYTE_FAIL bit which preserves the original reset event when set.

Note: While CPU Lockup is marked as a reset condition in software, CPU Lockup is not specifically a reset event. CPU Lockup is set to indicate that the CPU entered an unrecoverable exception. Execution stops but a reset is not applied. This is so that a debugger can interpret the cause of the error. We recommend that in a live application (i.e. no debugger attached) the watchdog be enabled by default so that the STM32W108 can be restarted.

6.2.3 Reset generation

The Reset Generation module responds to reset sources and generates the following reset signals:

- PORESET — Reset of the ARM® Cortex-M3 CPU and ARM® Cortex-M3 System Debug components (Flash Patch and Breakpoint, Data Watchpoint and Trace, Instrumentation Trace Macrocell, Nested Vectored Interrupt Controller). ARM defines PORESET as the region that is reset when power is applied.
- SYSRESET — Reset of the ARM® Cortex-M3 CPU without resetting the Core Debug and System Debug components, so that a live system can be reset without disturbing the debug configuration.
- DAPRESET — Reset to the SWJ's AHB Access Port (AHB-AP).

- PRESETHV — Peripheral reset for always-on power domain, for peripherals that are required to retain their configuration across a deep sleep cycle.
- PRESETLV — Peripheral reset for core power domain, for peripherals that are not required to retain their configuration across a deep sleep cycle.

Table 4 shows which reset sources generate certain resets.

Table 4. Generated resets

Reset source	Reset generation				
	PORESET	SYSRESET	DAPRESET	PRESETHV	PRESETLV
POR HV	X	X	X	X	X
POR LV (in deep sleep)	X	X	X		X
POR LV (not in deep sleep)	X	X	X	X	X
RSTB	X	X		X	X
Watchdog reset		X		X	X
Software reset		X		X	X
Option byte error	X	X			X
Normal deep sleep	X	X	X		X
Emulated deep sleep		X			X
Debug reset		X			

6.2.4 Reset register

Reset event source register (RESET_EVENT)

Address offset: 0x4000 002C
Reset value: 0x0000 0001

Table 5. Reset event source register (RESET_EVENT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CPU_LOCKUP	OPT_BYTE_FAIL	WAKE_UP_DSLEEP	SW_RST	W_DOG	RSTB_PIN	POWER_LV	POWER_HV
								r	r	r	r	r	r	r	r

Bit 7 CPU_LOCKUP: When set to ‘1’, the reset is due to core lockup.

Bit 6 OPT_BYTE_FAIL: When set to ‘1’, the reset is due to an Option byte load failure (may be set with other bits).

Bit 5 WAKE_UP_DSLEEP: When set to ‘1’, the reset is due to a wake-up from Deep Sleep.

Bit 4 SW_RST: When set to ‘1’, the reset is due to a software reset.

Bit 3 W_DOG: When set to ‘1’, the reset is due to watchdog expiration.

Bit 2 RSTB_PIN: When set to ‘1’, the reset is due to an external reset pin signal.

Bit 1 POWER_LV: When set to ‘1’, the reset is due to the application of a Core power supply (or previously failed).

Bit 0 POWER_HV: Always set to ‘1’, Normal power applied

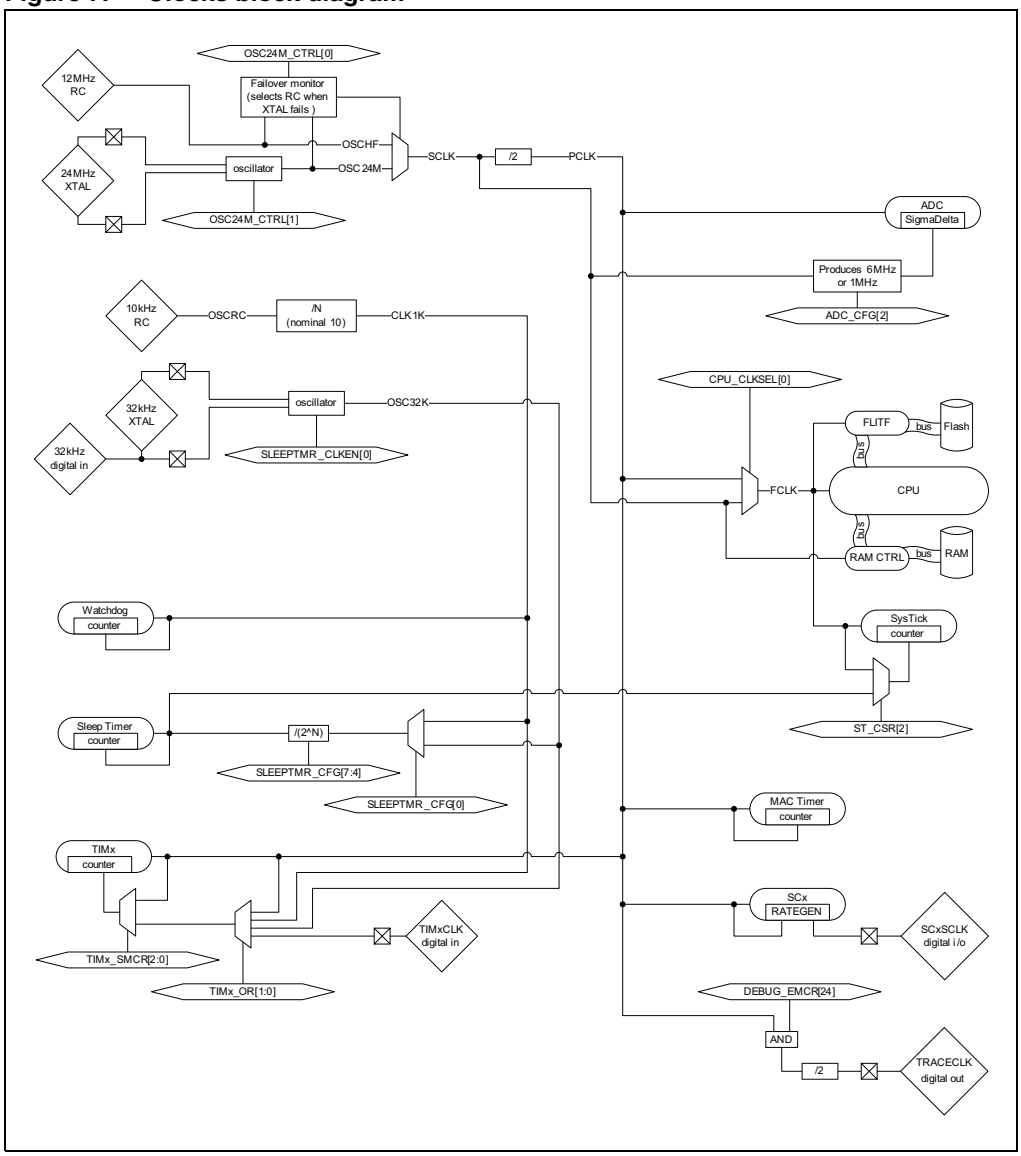
6.3 Clocks

The STM32W108 integrates four oscillators:

- High frequency RC oscillator
- 24 MHz crystal oscillator
- 10 kHz RC oscillator
- 32.768 kHz crystal oscillator

Figure 7 shows a block diagram of the clocks in the STM32W108. This simplified view shows all the clock sources and the general areas of the chip to which they are routed.

Figure 7. Clocks block diagram



6.3.1 High-frequency internal RC oscillator (OSCHF)

The high-frequency RC oscillator (OSCHF) is used as the default system clock source when power is applied to the core domain. The nominal frequency coming out of reset is 12 MHz.

Most peripherals, excluding the radio peripheral, are fully functional using the OSCHF clock source. Application software must be aware that peripherals are clocked at different speeds depending on whether OSCHF or OSC24M is being used. Since the frequency step of OSCHF is 0.5 MHz and the high-frequency crystal oscillator is used for calibration, the

calibrated accuracy of OSCHF is ±250 kHz ±40 ppm. The UART and ADC peripherals may not be usable due to the lower accuracy of the OSCHF frequency.

See also *Section 14.5.1: High frequency internal clock characteristics on page 202*.

6.3.2 High-frequency crystal oscillator (OSC24M)

The high-frequency crystal oscillator (OSC24M) requires an external 24 MHz crystal with an accuracy of ±40 ppm. Based upon the application's bill of materials and current consumption requirements, the external crystal may cover a range of ESR requirements.

The crystal oscillator has a software-programmable bias circuit to minimize current consumption. ST software configures the bias circuit for minimum current consumption.

All peripherals including the radio peripheral are fully functional using the OSC24M clock source. Application software must be aware that peripherals are clocked at different speeds depending on whether OSCHF or OSC24M is being used.

If the 24 MHz crystal fails, a hardware failover mechanism forces the system to switch back to the high-frequency RC oscillator as the main clock source, and a non-maskable interrupt (NMI) is signaled to the ARM® Cortex-M3 NVIC.

See also *Section 14.5.2: High frequency external clock characteristics on page 202*.

6.3.3 Low-frequency internal RC oscillator (OSCRC)

A low-frequency RC oscillator (OSCRC) is provided as an internal timing reference. The nominal frequency coming out of reset is 10 kHz, and ST software calibrates this clock to 10 kHz. From the tuned 10 kHz oscillator (OSCRC) ST software calibrates a fractional-N divider to produce a 1 kHz reference clock, CLK1K.

See also *Section 14.5.3: Low frequency internal clock characteristics on page 202*.

6.3.4 Low-frequency crystal oscillator (OSC32K)

A low-frequency 32.768 kHz crystal oscillator (OSC32K) is provided as an optional timing reference for on-chip timers. This oscillator is designed for use with an external watch crystal.

See also *Section 14.5.4: Low frequency external clock characteristics on page 203*.

6.3.5 Clock switching

The STM32W108 has two switching mechanisms for the main system clock, providing four clock modes.

The register bit OSC24M_SEL in the OSC24M_CTRL register switches between the high-frequency RC oscillator (OSCHF) and the high-frequency crystal oscillator (OSC24M) as the main system clock (SCLK). The peripheral clock (PCLK) is always half the frequency of SCLK.

The register bit CPU_CLK_SEL in the CPU_CLKSEL register switches between PCLK and SCLK to produce the ARM® Cortex-M3 CPU clock (FCLK). The default and preferred mode of operation is to run the CPU at the lower PCLK frequency, 12 MHz, but the higher SCLK frequency, 24 MHz, can be selected to give higher processing performance at the expense of an increase in power consumption.

In addition to these modes, further automatic control is invoked by hardware when flash programming is enabled. To ensure accuracy of the flash controller's timers, the FCLK frequency is forced to 12 MHz during flash programming and erase operations.

Table 6. System clock modes

OSC24M_SEL	CPU_CLK_SEL	SCLK	PCLK	f_{CLK}	
				Flash Program/ Erase Inactive	Flash Program/ Erase Active
0 (OSCHF)	0 (Normal CPU)	12 MHz	6 MHz	6 MHz	12 MHz
0 (OSCHF)	1 (Fast CPU)	12 MHz	6 MHz	12 MHz	12 MHz
1 (OSC24M)	0 (Normal CPU)	24 MHz	12 MHz	12 MHz	12 MHz
1 (OSC24M)	1 (Fast CPU)	24 MHz	12 MHz	24 MHz	12 MHz

6.3.6 Clock switching registers

XTAL or OSCHF main clock select register (OSC24M_CTRL)

Address offset: 0x4000 401C
Reset value: 0x0000 0000

Table 7. XTAL or OSCHF main clock select register (OSC24M_CTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OSC24M_EN	OSC24M_SEL
														rws	rws

Bit 1 OSC24M_EN: When set to ‘1’, 24 MHz crystal oscillator is main clock.

Bit 0 OSC24M_SEL: When set to ‘0’, OSCHF is selected. When set to ‘1’, XTAL is selected.

CPU clock source select register (CPU_CLK_SEL)

Address offset: 0x4000 4020
Reset value: 0x0000 0000

Table 8. CPU clock source select register (CPU_CLK_SEL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rserved															CPU_CLK_SEL
															rws

Bit 0 CPU_CLK_SEL: When set to ‘0’, 12-MHz CPU clock is selected. When set to ‘1’, 24-MHz CPU clock is selected. Note that the clock selection also determines if RAM controller is running at the same speed as the HCLK (CPU_CLK_SEL = ‘1’) or double speed of HCLK (CPU_CLK_SEL = ‘0’).

6.4 System timers

6.4.1 Watchdog timer

The STM32W108 integrates a watchdog timer which can be enabled to provide protection against software crashes and ARM® Cortex-M3 CPU lockup. By default, it is disabled at power up of the always-on power domain. The watchdog timer uses the calibrated 1 kHz clock (CLK1K) as its reference and provides a nominal 2.048 s timeout. A low water mark interrupt occurs at 1.792 s and triggers an NMI to the ARM® Cortex-M3 NVIC as an early warning. When enabled, periodically reset the watchdog timer by writing to the WDOG_RESTART register before it expires.

The watchdog timer can be paused when the debugger halts the ARM® Cortex-M3. To enable this functionality, set the bit DBG_PAUSE in the SLEEP_CONFIG register.

If the low-frequency internal RC oscillator (OSCRC) is turned off during deep sleep, CLK1K stops. As a consequence the watchdog timer stops counting and is effectively paused during deep sleep.

The watchdog enable/disable bits are protected from accidental change by requiring a two step process. To enable the watchdog timer the application must first write the enable code 0xEABE to the WDOG_CTRL register and then set the WDOG_EN register bit. To disable the timer the application must write the disable code 0xDEAD to the WDOG_CTRL register and then set the WDOG_DIS register bit.

6.4.2 Sleep timer

The STM32W108 integrates a 32-bit timer dedicated to system timing and waking from sleep at specific times. The sleep timer can use either the calibrated 1 kHz reference(CLK1K), or the 32 kHz crystal clock (CLK32K). The default clock source is the internal 1 kHz clock. The sleep timer clock source is chosen with the SLEEPTMR_CLKSEL register.

The sleep timer has a prescaler, a divider of the form 2^N, where N can be programmed from 1 to 2^15. This divider allows for very long periods of sleep to be timed. The timer provides two compare outputs and wrap detection, all of which can be used to generate an interrupt or a wake up event.

The sleep timer is paused when the debugger halts the ARM® Cortex-M3. No additional register bit must be set.

To save current during deep sleep, the low-frequency internal RC oscillator (OSCRC) can be turned off. If OSCRC is turned off during deep sleep and a low-frequency 32.768 kHz crystal oscillator is not being used, then the sleep timer will not operate during deep sleep and sleep timer wake events cannot be used to wakeup the STM32W108.

6.4.3 Event timer

The SysTick timer is an ARM® standard system timer in the NVIC. The SysTick timer can be clocked from either the FCLK (the clock going into the CPU) or the Sleep Timer clock. FCLK is either the SCLK or PCLK as selected by CPU_CLK_SEL (see *Section 6.3.5: Clock switching on page 42*).

6.4.4 Slow timers (Watchdog and Sleeptimer) control and status registers

These registers are powered from the always-on power domain.

All registers are only writable when in System mode

Watchdog general control register (WDOG_CFG)

Register bits for general top level chip functions and protection.

Watchdog bits can only be written after first writing the appropriate code to the WDOG_CTRL register.

Address: 0x4000 6000
Reset value: 0x0000 0002

Table 9. Watchdog general control register (WDOG_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														WDOG_DIS	WDOG_EN
														rw	rw

Bit 1 WDOG_DIS: Watchdog disable

Bit 0 WDOG_EN: Watchdog enable

Watchdog control register (WDOG_CTRL)

Requires magic number write to arm the watchdog enable or disable function.

Address: 0x4000 6004
Reset value: 0x0000 0000

Table 10. Watchdog control register (WDOG_CTRL)

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

Bits [15:0] WDOG_CTRL: Write 0xDEAD to disable or 0xEABE to enable.

Watchdog restart register (WDOG_RESTART)

Write any value to this register to kick-start the watchdog.

Address: 0x4000 6008
Reset value: 0x0000 0000

Sleep timer configuration register (SLEEPTMR_CFG)

This register sets the various options for the Sleep timer.

Address: 0x4000 600C
Reset value: 0x0000 0400

Table 11. Sleep timer configuration register (SLEEPTMR_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			SLEEPTMR_REVERSE	SLEEPTMR_ENABLE	SLEEPTMR_DBGPAUSE	Reserved		SLEEPTMR_CLKDIV				Reserved			SLEEPTMR_CLKSEL
r			rw	rw	rw	r		rw				r			rw

Bit 12 SLEEPTMR_REVERSE:
0: count forward; 1: count backwards.
Only changes when ENABLE bit is set to ‘0’.

Bit 11 SLEEPTMR_ENABLE:
0: disable sleep timer; 1: enable sleep timer.
To change other register bits (REVERSE, CLK_DIV, CLK_SEL), this bit must be set to ‘0’.
Enabling/Disabling latency can be up 2 to 3 clock-periods of selected clock.

Bit 10 SLEEPTMR_DBGPAUSE: Debug Pause
0: The timer continues working in Debug mode.
1: The timer is paused in Debug mode when the CPU is halted.

Bits [7:4] SLEEPTMR_CLKDIV: Sleep timer prescaler setting
Divides clock by 2^N where N = 0 to 15.
Can only be changed when the ENABLE bit is set to ‘0’.

Bit 0 SLEEPTMR_CLKSEL: Clock Select
0: Calibrated 1kHz RC clock (default); 1: 32kHz
Can only be changed when the ENABLE bit is set to ‘0’.

Sleep timer count high register (SLEEPTMR_CNTH)

Address: 0x4000 6010
Reset value: 0x0000 0000

Table 12. Sleep timer count high register (SLEEPTMR_CNTH)

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

Bits [15:0] SLEEPTMR_CNTH_FIELD:
Sleep timer counter high value [31:16].
Reading this register updates the SLEEP_COUNT_L for subsequent reads.

Sleep timer count low register (SLEEPTMR_CNTL)

Address: 0x4000 6014
Reset value: 0x0000 0000

Table 13. Sleep timer count low register (SLEEPTMR_CNTL)

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

Bits [15:0] SLEEPTMR_CNTL_FIELD:
Sleep timer counter low value [15:16].
This register is only valid following a read of the SLEEPTMR_CNTH register.

Sleep timer compare A high register (SLEEPTMR_CMPAH)

Address: 0x4000 6018
Reset value: 0x0000 FFFF

Table 14. Sleep timer compare A high register (SLEEPTMR_CMPAH)

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

Bits [15:0] SLEEPTMR_CMPAH_FIELD:

Sleep timer compare A high value [31:16].
Sleep timer compare value, writing updates COMP_A_H (directly) and COMP_A_L (from hold register).
Can only be changed when the ENABLE bit (bit 11 of SLEEP_CONFIG register) is set to '0'.
If changed when the ENABLE bit is set to '1', a spurious interrupt may be generated.
Therefore it is recommended to disable interrupts before changing this register.

Sleep timer compare A low register (SLEEPTMR_CMPAL)

Address: 0x4000 601C
Reset value: 0x0000 FFFF

Table 15. Sleep timer compare A low register (SLEEPTMR_CMPAL)

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

Bits [15:0] SLEEPTMR_CMPAL_FIELD:

Sleep timer compare A low value [15:0].
Writing to this register puts value in hold register until a write to the SLEEPTMR_CMPAH register.
Can only be changed when the ENABLE bit (bit 11 of SLEEP_CONFIG register) is set to '0'.
If changed when the ENABLE bit is set to '1', a spurious interrupt may be generated.
Therefore it is recommended to disable interrupts before changing this register.

Sleep timer compare B high register (SLEEPTMR_CMPBH)

Address: 0x4000 6020
Reset value: 0x0000 FFFF

Table 16. Sleep timer compare B high register (SLEEPTMR_CMPBH)

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

Bits [15:0] SLEEPTMR_CMPBH_FIELD:

Sleep timer compare B high value [31:16].
Sleep timer compare value, writing updates COMP_B_H (directly) and COMP_B_L (from hold register).
Can only be changed when the ENABLE bit (bit 11 of SLEEP_CONFIG register) is set to '0'.
If changed when the ENABLE bit is set to '1', a spurious interrupt may be generated.
Therefore it is recommended to disable interrupts before changing this register.

Sleep timer compare B low register (SLEEPTMR_CMPBL)

Address: 0x4000 6024
Reset value: 0x0000 FFFF

Table 17. Sleep timer compare B low register (SLEEPTMR_CMPBL)

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

Bits [15:0] SLEEPTMR_CMPBL_FIELD:

Sleep timer compare B low value [15:0].
Writing to this register puts value in hold register until a write to the SLEEPTMR_CMPBH register.
Can only be changed when the ENABLE bit (bit 11 of SLEEP_CONFIG register) is set to '0'.
If changed when the ENABLE bit is set to '1', a spurious interrupt may be generated.
Therefore it is recommended to disable interrupts before changing this register.

Sleep timer interrupt source register (INT_SLEEPTMRFLAG)

Address: 0x4000 A014
Reset value: 0x0000 0000

Table 18. Sleep timer interrupt source register (INT_SLEEPTMRFLAG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													INT_SLEEPTMRCMPB	INT_SLEEPTMRCMPA	INT_SLEEPTMRWRAP
r													rw	rw	rw

Bit 2 INT_ SLEEPTMR CMPB: Sleep timer compare B
Note: Bits are cleared when set to ‘1’.

Bit 1 INT_SLEEPTMRCMPA: Sleep timer compare A
Note: Bits are cleared when set to ‘1’.

Bit 0 INT_SLEEPTMRWRAP: Sleep timer overflow
Note: Bits are cleared when set to ‘1’.

Sleep timer interrupt mask register (INT_SLEEPTMRCFG)

Address: 0x4000 A054
Reset value: 0x0000 0000

Table 19. Sleep timer interrupt mask register (INT_SLEEPTMRCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r													rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													INT_ SLEEP TMR CMPB	INT_ SLEEP TMR CMPA	INT_ SLEEP TMR WRAP
r													rw	rw	rw

Bit 2 INT_ SLEEPTMR CMPB: Sleep timer compare B

Bit 1 INT_SLEEPTMRCMPA: Sleep timer compare A

Bit 0 INT_SLEEPTMRWRAP: Sleep timer overflow

Sleep timer clock source enables (SLEEPTMR_CLKEN)

This timer controls the low power clock gated modes.

Clearing CLKRC_EN before executing WFE with SLEEPDEEP bit set in the NVIC System control register causes DEEP_SLEEP2 to be entered. Setting this bit causes DEEP_SLEEP1 to be entered.

Address: 0x4000 0008
Reset value: 0x0000 0002

Table 20. Sleep timer clock source enables (SLEEPTMR_CLKEN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														SLEEP TMR_ CLK10K EN	SLEEP TMR_ CLK32K EN
r														rw	rw

Bit 1 SLEEPTMR_CLK10KEN: Enables 10kHz internal RC during deep
Note: Bits are cleared when set to ‘1’.

Bit 0 SLEEPTMR_CLK32KEN: Enables 32kHz external XTAL
Note: Bits are cleared when set to ‘1’.

6.5 Power management

The STM32W108's power management system is designed to achieve the lowest deep sleep current consumption possible while still providing flexible wakeup sources, timer activity, and debugger operation. The STM32W108 has four main sleep modes:

- Idle Sleep: Puts the CPU into an idle state where execution is suspended until any interrupt occurs. All power domains remain fully powered and nothing is reset.
- Deep Sleep 1: The primary deep sleep state. In this state, the core power domain is fully powered down and the sleep timer is active
- Deep Sleep 2: The same as Deep Sleep 1 except that the sleep timer is inactive to save power. In this mode the sleep timer cannot wakeup the STM32W108.
- Deep Sleep 0 (also known as Emulated Deep Sleep): The chip emulates a true deep sleep without powering down the core domain. Instead, the core domain remains powered and all peripherals except the system debug components (ITM, DWT, FPB, NVIC) are held in reset. The purpose of this sleep state is to allow STM32W108 software to perform a deep sleep cycle while maintaining debug configuration such as breakpoints.

6.5.1 Wake sources

When in deep sleep the STM32W108 can be returned to the running state in a number of ways, and the wake sources are split depending on deep sleep 1 or deep sleep 2.

The following wake sources are available in both deep sleep 1 and 2.

- Wake on GPIO activity: Wake due to change of state on any GPIO.
- Wake on serial controller 1: Wake due to a change of state on GPIO Pin PB2.
- Wake on serial controller 2: Wake due to a change of state on GPIO Pin PA2.
- Wake on IRQD: Wake due to a change of state on IRQD. Since IRQD can be configured to point to any GPIO, this wake source is another means of waking on any GPIO activity.
- Wake on setting of CDBGPWRUPREQ: Wake due to setting the CDBGPWRUPREQ bit in the debug port in the SWJ.
- Wake on setting of CSYSPWRUPREQ: Wake due to setting the CSYSPWRUPREQ bit in the debug port in the SWJ.

The following sources are only available in deep sleep 1 since the sleep timer is not active in deep sleep 2.

- Wake on sleep timer compare A.
- Wake on sleep timer compare B.
- Wake on sleep timer wrap.

The following source is only available in deep sleep 0 since the SWJ is required to write memory to set this wake source and the SWJ only has access to some registers in deep sleep 0.

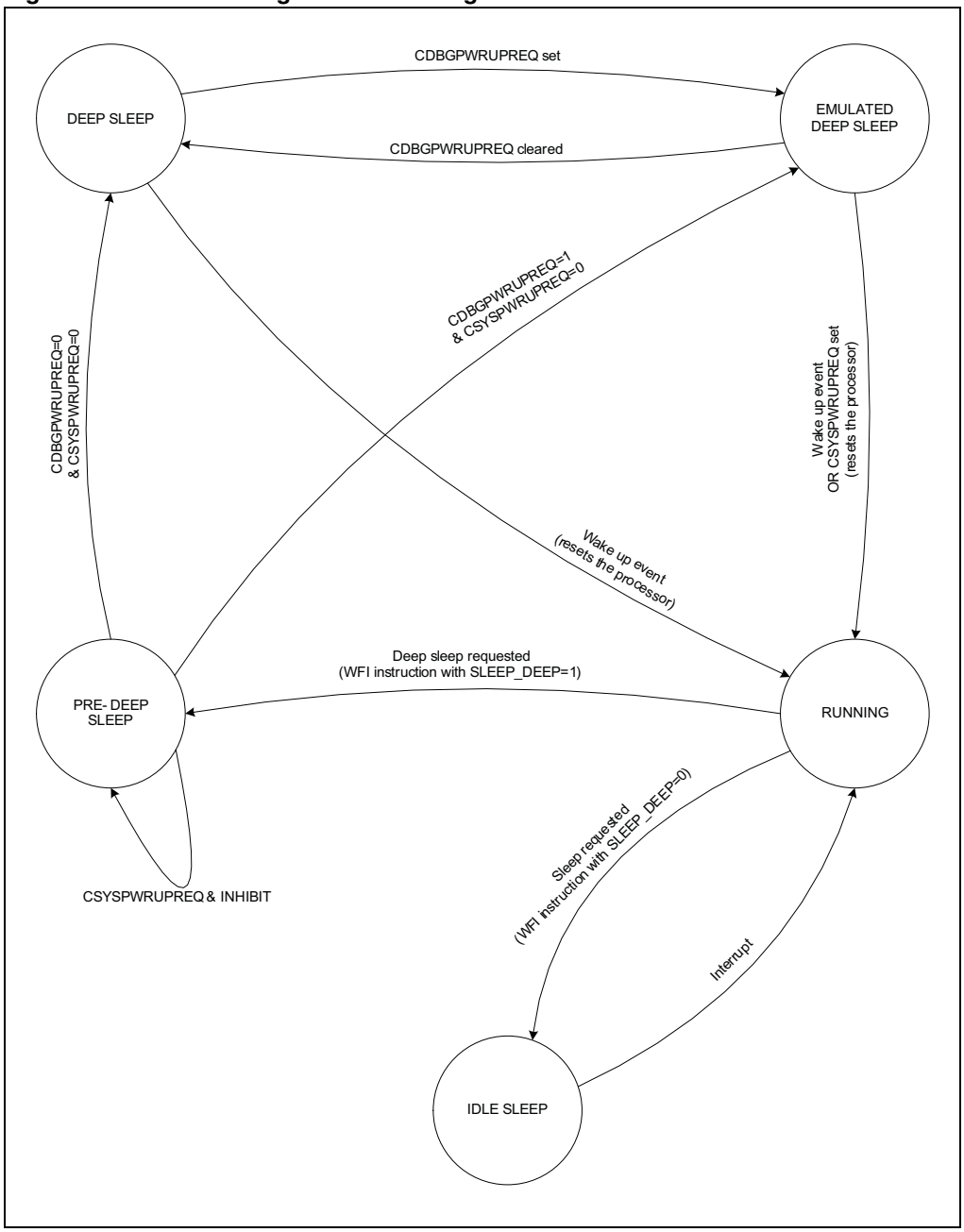
- Wake on write to the WAKE_CORE register bit.

The Wakeup Recording module monitors all possible wakeup sources. More than one wakeup source may be recorded because events are continually being recorded (not just in deep-sleep), since another event may happen between the first wake event and when the STM32W108 wakes up.

6.5.2 Basic sleep modes

The power management state diagram in *Figure 8* shows the basic operation of the power management controller.

Figure 8. Power management state diagram



In normal operation an application may request one of two low power modes through program execution:

- Idle Sleep is achieved by executing a WFI instruction whilst the SLEEPDEEP bit in the Cortex System Control register (SCS_SCR) is clear. This puts the CPU into an idle state where execution is suspended until an interrupt occurs. This is indicated by the state at the bottom of the diagram. Power is maintained to the core logic of the STM32W108 during the Idle Sleeping state.
- Deep sleep is achieved by executing a WFI instruction with the SLEEPDEEP bit in SCS_SCR set. This triggers the state transitions around the main loop of the diagram, resulting in powering down the STM32W108's core logic, and leaving only the always-on domain powered. Wake up is triggered when one of the pre-determined events occurs.

If a deep sleep is requested the STM32W108 first enters a pre-deep sleep state. This state prevents any section of the chip from being powered off or reset until the SWJ goes idle (by clearing CSYSPWRUPREQ). This pre-deep sleep state ensures debug operations are not interrupted.

In the deep sleep state the STM32W108 waits for a wake up event which will return it to the running state. In powering up the core logic the ARM® Cortex-M3 is put through a reset cycle and ST software restores the stack and application state to the point where deep sleep was invoked.

6.5.3 Further options for deep sleep

By default, the low-frequency internal RC oscillator (OSCRC) is running during deep sleep (known as deep sleep 1).

To conserve power, OSCRC can be turned off during deep sleep. This mode is known as deep sleep 2. Since the OSCRC is disabled, the sleep timer and watchdog timer do not function and cannot wake the chip unless the low-frequency 32.768 kHz crystal oscillator is used. Non-timer based wake sources continue to function. Once a wake event occurs, the OSCRC restarts and becomes enabled.

6.5.4 Use of debugger with sleep modes

The debugger communicates with the STM32W108 using the SWJ.

When the debugger is connected, the CDBGPWRUPREQ bit in the debug port in the SWJ is set, the STM32W108 will only enter deep sleep 0 (the Emulated Deep Sleep state). The CDBGPWRUPREQ bit indicates that a debug tool is connected to the chip and therefore there may be debug state in the system debug components. To maintain the state in the system debug components only deep sleep 0 may be used, since deep sleep 0 will not cause a power cycle or reset of the core domain. The CSYSPWRUPREQ bit in the debug port in the SWJ indicates that a debugger wants to access memory actively in the STM32W108. Therefore, whenever the CSYSPWRUPREQ bit is set while the STM32W108 is awake, the STM32W108 cannot enter deep sleep until this bit is cleared. This ensures the STM32W108 does not disrupt debug communication into memory.

Clearing both CSYSPWRUPREQ and CDBGPWRUPREQ allows the STM32W108 to achieve a true deep sleep state (deep sleep 1 or 2). Both of these signals also operate as wake sources, so that when a debugger connects to the STM32W108 and begins accessing the chip, the STM32W108 automatically comes out of deep sleep. When the debugger

initiates access while the STM32W108 is in deep sleep, the SWJ intelligently holds off the debugger for a brief period of time until the STM32W108 is properly powered and ready.

For more information regarding the SWJ and the interaction of debuggers with deep sleep, contact ST support for Application Notes and ARM® CoreSight documentation.

6.6 Security accelerator

The STM32W108 contains a hardware AES encryption engine accessible from the ARM® Cortex-M3. NIST-based CCM, CCM*, CBC-MAC, and CTR modes are implemented in hardware. These modes are described in the IEEE 802.15.4-2003 specification, with the exception of CCM*, which is described in the ZigBee Security Services Specification 1.0.

7 Integrated voltage regulator

The STM32W108 integrates two low dropout regulators to provide 1.8 V and 1.25 V power supplies. The 1V8 regulator supplies the analog and memories, and the 1V25 regulator supplies the digital core. In deep sleep the voltage regulators are disabled.

When enabled, the 1V8 regulator steps down the pads supply voltage (VDD_PADS) from a nominal 3.0 V to 1.8 V. The regulator output pin (VREG_OUT) must be decoupled externally with a suitable capacitor. VREG_OUT should be connected to the 1.8 V supply pins VDDA, VDD_RF, VDD_VCO, VDD_SYNTH, VDD_IF, and VDD_MEM. The 1V8 regulator can supply a maximum of 50 mA.

When enabled, the 1V25 regulator steps down VDD_PADS to 1.25 V. The regulator output pin (VDD_CORE, (Pin 17) must be decoupled externally with a suitable capacitor. It should connect to the other VDD_CORE pin (Pin 44). The 1V25 regulator can supply a maximum of 10 mA.

The regulators are controlled by the digital portion of the chip as described in *Section 6: System modules*.

Table 21. 1.8 V integrated voltage regulator specifications

Parameter	Min.	Typ.	Max.	Units	Comments
Supply range for regulator	2.1		3.6	V	VDD_PADS
1V8 regulator output	-5%	1.8	+5%	V	Regulator output after initialization
1V8 regulator output after reset	-5%	1.75	+5%		Regulator output after reset
1V25 regulator output	-5%	1.25	+5%	V	Regulator output after initialization
1V25 regulator output after reset	-5%	1.45	+5%		Regulator output after reset
1V8 regulator capacitor		2.2		µF	Low ESR tantalum capacitor ESR greater than 2 Ω ESR less than 10 Ω De-coupling less than100 nF ceramic
1V25 regulator capacitor		1.0		µF	Ceramic capacitor (0603)
1V8 regulator output current	0		50	mA	Regulator output current
1V25 regulator output current	0		10	mA	Regulator output current
No load current		600		µA	No load current (bandgap and regulators)
1V8 regulator current limit		200		mA	Short circuit current limit
1V25 regulator current limit		25		mA	Short circuit current limit

Table 21. 1.8 V integrated voltage regulator specifications (continued)

Parameter	Min.	Typ.	Max.	Units	Comments
1V8 regulator start-up time		50		µs	0 V to POR threshold 2.2 µF capacitor
1V25 regulator start-up time		50		µs	0 V to POR threshold 1.0 µF capacitor

An external 1.8 V regulator may replace both internal regulators. The STM32W108 can control external regulators during deep sleep using open-drain GPIO PA7, as described in *Section 8: General-purpose input/outputs*. The STM32W108 drives PA7 low during deep sleep to disable the external regulator and an external pull-up is required to release this signal to indicate that supply voltage should be provided. Current consumption increases approximately 2 mA when using an external regulator. When using an external regulator the internal regulators should be disabled through software.

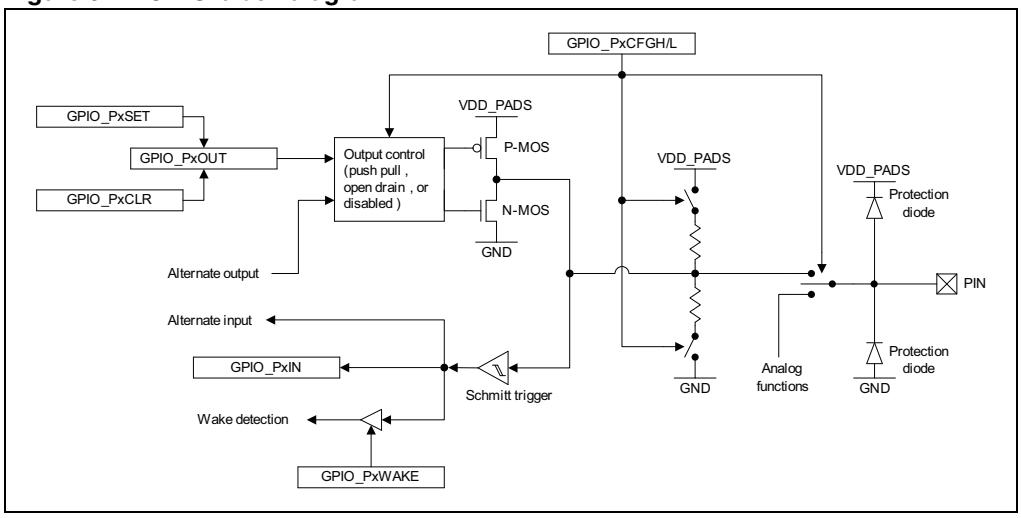
8 General-purpose input/outputs

The STM32W108 has 24 multi-purpose GPIO pins that may be individually configured as:

- General purpose output
- General purpose open-drain output
- Alternate output controlled by a peripheral device
- Alternate open-drain output controlled by a peripheral device
- Analog
- General purpose input
- General purpose input with pull-up or pull-down resistor

The basic structure of a single GPIO is illustrated in *Figure 9*.

Figure 9. GPIO block diagram



A Schmitt trigger converts the GPIO pin voltage to a digital input value. The digital input signal is then always routed to the GPIO_PxIN register; to the alternate inputs of associated peripheral devices; to wake detection logic if wake detection is enabled; and, for certain pins, to interrupt generation logic. Configuring a pin in analog mode disconnects the digital input from the pin and applies a high logic level to the input of the Schmitt trigger.

Only one device at a time can control a GPIO output. The output is controlled in normal output mode by the GPIO_PxOUT register and in alternate output mode by a peripheral device. When in input mode or analog mode, digital output is disabled.

8.1 Functional description

8.1.1 GPIO ports

The 24 GPIO pins are grouped into three ports: PA, PB, and PC. Individual GPIOs within a port are numbered 0 to 7 according to their bit positions within the GPIO registers.

Note: Because GPIO port registers' functions are identical, the notation Px is used here to refer to PA, PB, or PC. For example, GPIO_PxIN refers to the registers GPIO_PAIN, GPIO_PBIN, and GPIO_PCIN.

Each of the three GPIO ports has the following registers whose low-order eight bits correspond to the port's eight GPIO pins:

- GPIO_PxIN (input data register) returns the pin level (unless in analog mode).
- GPIO_PxOUT (output data register) controls the output level in normal output mode.
- GPIO_PxCLR (clear output data register) clears bits in GPIO_PxOUT.
- GPIO_PxSET (set output data register) sets bits in GPIO_PxOUT.
- GPIO_PxWAKE (wake monitor register) specifies the pins that can wake the STM32W108.

In addition to these registers, each port has a pair of configuration registers, GPIO_PxCFGH and GPIO_PxCFGL. These registers specify the basic operating mode for the port's pins. GPIO_PxCFGL configures the pins Px[3:0] and GPIO_PxCFGH configures the pins Px[7:4]. For brevity, the notation GPIO_PxCFGH/L refers to the pair of configuration registers.

Five GPIO pins (PA6, PA7, PB6, PB7 and PC0) can sink and source higher current than standard GPIO outputs. Refer to *Table 151: Digital I/O characteristics on page 208* for more information.

8.1.2 Configuration

Each pin has a 4-bit configuration value in the GPIO_PxCFGH/L register. The various GPIO modes and their 4 bit configuration values are shown in *Table 22*.

Table 22. GPIO configuration modes

GPIO mode	GPIO_PxCFGH/L	Description
Analog	0x0	Analog input or output. When in analog mode, the digital input (GPIO_PxIN) always reads 1.
Input (floating)	0x4	Digital input without an internal pull up or pull down. Output is disabled.
Input (pull-up or pull-down)	0x8	Digital input with an internal pull up or pull down. A set bit in GPIO_PxOUT selects pull up and a cleared bit selects pull down. Output is disabled.
Output (push-pull)	0x1	Push-pull output. GPIO_PxOUT controls the output.
Output (open-drain)	0x5	Open-drain output. GPIO_PxOUT controls the output. If a pull up is required, it must be external.
Alternate Output (push-pull)	0x9	Push-pull output. An onboard peripheral controls the output.

Table 22. GPIO configuration modes (continued)

GPIO mode	GPIO_PxCFGH/L	Description
Alternate Output (open-drain)	0xD	Open-drain output. An onboard peripheral controls the output. If a pull up is required, it must be external.
Alternate Output (push-pull) SPI SCLK Mode	0xB	Push-pull output mode only for SPI master mode SCLK pins.

If a GPIO has two peripherals that can be the source of alternate output mode data, then other registers in addition to GPIO_PxCFGH/L determine which peripheral controls the output.

Several GPIOs share an alternate output with Timer 2 and the Serial Controllers. Bits in Timer 2's TIM2_OR register control routing Timer 2 outputs to different GPIOs. Bits in Timer 2's TIM2_CCER register enable Timer 2 outputs. When Timer 2 outputs are enabled they override Serial Controller outputs. *Table 23* indicates the GPIO mapping for Timer 2 outputs depending on the bits in the register TIM2_OR. Refer to *Section 10: General-purpose timers on page 112* for complete information on timer configuration.

Table 23. Timer 2 output configuration controls

Timer 2 output	Option register bit	GPIO mapping selected by TIM2_OR bit	
		0	1
TIM2_CH1	TIM2_OR[4]	PA0	PB1
TIM2_CH2	TIM2_OR[5]	PA3	PB2
TIM2_CH3	TIM2_OR[6]	PA1	PB3
TIM2_CH4	TIM2_OR[7]	PA2	PB4

For outputs assigned to the serial controllers, the serial interface mode registers (SCx_MODE) determine how the GPIO pins are used.

The alternate outputs of PA4 and PA5 can either provide packet trace data (PTI_EN and PTI_DATA), or synchronous CPU trace data (TRACEDATA2 and TRACEDATA3).

If a GPIO does not have an associated peripheral in alternate output mode, its output is set to 0.

8.1.3 Forced functions

For some GPIOs the GPIO_PxCFGH/L configuration may be overridden. *Table 24* shows the GPIOs that can have different functions forced on them regardless of the GPIO_PxCFGH/L registers.

Note: The DEBUG_DIS bit in the GPIO_DBGCFG register can disable the Serial Wire/JTAG debugger interface. When this bit is set, all debugger-related pins (PC0, PC2, PC3, PC4) behave as standard GPIO.

Table 24. GPIO forced functions

GPIO	Override condition	Forced function	Forced signal
PA7	GPIO_EXTREGEN bit set in the GPIO_DBGCFG register	Open-drain output	REG_EN
PC0	Debugger interface is active in JTAG mode	Input with pull up	JRST
PC2	Debugger interface is active in JTAG mode	Push-pull output	JTDO
PC3	Debugger interface is active in JTAG mode	Input with pull up	JDTI
PC4	Debugger interface is active in JTAG mode	Input with pull up	JTMS
PC4	Debugger interface is active in Serial Wire mode	Bidirectional (push-pull output or floating input) controlled by debugger interface	SWDIO

8.1.4 Reset

A full chip reset is one due to power on (low or high voltage), the NRST pin, the watchdog, or the SYSRESETREQ bit. A full chip reset affects the GPIO configuration as follows:

- The GPIO_PxCFGH/L configurations of all pins are configured as floating inputs.
- The GPIO_EXTREGEN bit is set in the GPIO_DBGCFG register, which overrides the normal configuration for PA7.
- The GPIO_DEBUGDIS bit in the GPIO_DBGCFG register is cleared, allowing Serial Wire/JTAG access to override the normal configuration of PC0, PC2, PC3, and PC4.

8.1.5 nBOOTMODE

nBOOTMODE is a special alternate function of PA5 that is active only during a pin reset (NRST) or a power-on-reset of the always-powered domain (POR_HV). If nBOOTMODE is asserted (pulled or driven low) when coming out of reset, the processor starts executing an embedded serial boot loader instead of its normal program.

While in reset and during the subsequent power-on-reset startup delay (512 high-frequency RC oscillator periods), PA5 is automatically configured as an input with a pull-up resistor. At the end of this time, the STM32W108 samples nBOOTMODE: a high level selects normal startup, and a low level selects the boot loader. After nBOOTMODE has been sampled, PA5 is configured as a floating input. The GPIO_BOOTMODE bit in the GPIO_DBGSTAT register captures the state of nBOOTMODE so that software may act on this signal if required.

Note: To avoid inadvertently asserting nBOOTMODE, PA5's capacitive load should not exceed 252 pF.

8.1.6 GPIO modes

Analog mode

Analog mode enables analog functions, and disconnects a pin from the digital input and output logic. Only the following GPIO pins have analog functions:

- PA4, PA5, PB5, PB6, PB7, and PC1 can be analog inputs to the ADC.
- PB0 can be an external analog voltage reference input to the ADC, or it can output the internal analog voltage reference from the ADC.
- PC6 and PC7 can connect to an optional 32.768 kHz crystal.

Note: When an external timing source is required, a 32.768 kHz crystal is commonly connected to PC6 and PC7. Alternatively, when PC7 is configured as a digital input, PC7 can accept a digital external clock input.

When configured in analog mode:

- The output drivers are disabled.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to a high logic level.
- Reading GPIO_PxIN returns a constant 1.

Input mode

Input mode is used both for general purpose input and for on-chip peripheral inputs. Input floating mode disables the internal pull-up and pull-down resistors, leaving the pin in a high-impedance state. Input pull-up or pull-down mode enables either an internal pull-up or pull-down resistor based on the GPIO_PxOUT register. Setting a bit to 0 in GPIO_PxOUT enables the pull-down and setting a bit to 1 enables the pull up.

When configured in input mode:

- The output drivers are disabled.
- An internal pull-up or pull-down resistor may be activated depending on GPIO_PxCFGH/L and GPIO_PxOUT.
- The Schmitt trigger input is connected to the pin.
- Reading GPIO_PxIN returns the input at the pin.
- The input is also available to on-chip peripherals.

Output mode

Output mode provides a general purpose output under direct software control. Regardless of whether an output is configured as push-pull or open-drain, the GPIO's bit in the GPIO_PxOUT register controls the output. The GPIO_PxSET and GPIO_PxCLR registers can atomically set and clear bits within GPIO_PxOUT register. These set and clear registers simplify software using the output port because they eliminate the need to disable interrupts to perform an atomic read-modify-write operation of GPIO_PxOUT.

When configured in output mode:

- The output drivers are enabled and are controlled by the value written to GPIO_PxOUT:
- In open-drain mode: 0 activates the N-MOS current sink; 1 tri-states the pin.
- In push-pull mode: 0 activates the N-MOS current sink; 1 activates the P-MOS current source.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to the pin.
- Reading GPIO_PxIN returns the input at the pin.
- Reading GPIO_PxOUT returns the last value written to the register.

Note: Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

Alternate output mode

In this mode, the output is controlled by an on-chip peripheral instead of GPIO_PxOUT and may be configured as either push-pull or open-drain. Most peripherals require a particular output type - I^2C requires an open-drain driver, for example - but since using a peripheral does not by itself configure a pin, the GPIO_PxCFGH/L registers must be configured properly for a peripheral's particular needs. As described in *Section 8.1.2: Configuration on page 59*, when more than one peripheral can be the source of output data, registers in addition to GPIO_PxCFGH/L determine which to use.

When configured in alternate output mode:

- The output drivers are enabled and are controlled by the output of an on-chip peripheral:
- In open-drain mode: 0 activates the N-MOS current sink; 1 tri-states the pin.
- In push-pull mode: 0 activates the N-MOS current sink; 1 activates the P-MOS current source.
- The internal pull-up and pull-down resistors are disabled.
- The Schmitt trigger input is connected to the pin.
- Reading GPIO_PxIN returns the input to the pin.

Note: Depending on configuration and usage, GPIO_PxOUT and GPIO_PxIN may not have the same value.

Alternate output SPI SCLK mode

SPI master mode SCLK outputs, PB3 (SC1SCLK) or PA2 (SC2SCLK), use a special output push-pull mode reserved for those signals. Otherwise this mode is identical to alternate output mode.

8.1.7 Wake monitoring

The GPIO_PxWAKE registers specify which GPIOs are monitored to wake the processor. If a GPIO's wake enable bit is set in GPIO_PxWAKE, then a change in the logic value of that GPIO causes the STM32W108 to wake from deep sleep. The logic values of all GPIOs are captured by hardware upon entering sleep. If any GPIO's logic value changes while in sleep and that GPIO's GPIO_PxWAKE bit is set, then the STM32W108 will wake from deep sleep. (There is no mechanism for selecting a specific rising-edge, falling-edge, or level on a GPIO: any change in logic value triggers a wake event.) Hardware records the fact that GPIO

activity caused a wake event, but not which specific GPIO was responsible. Instead, software should read the state of the GPIOs on waking to determine the cause of the event.

The register GPIO_WAKEFILT contains bits to enable digital filtering of the external wakeup event sources: the GPIO pins, SC1 activity, SC2 activity, and IRQD. The digital filter operates by taking samples based on the (nominal) 10 kHz RC oscillator. If three samples in a row all have the same logic value, and this sampled logic value is different from the logic value seen upon entering sleep, the filter outputs a wakeup event.

In order to use GPIO pins to wake the STM32W108 from deep sleep, the GPIO_WAKE bit in the WAKE_SEL register must be set. Waking up from GPIO activity does not work with pins configured for analog mode since the digital logic input is always set to 1 when in analog mode. Refer to *Section 6: System modules on page 35* for information on the STM32W108's power management and sleep modes.

8.2 External interrupts

The STM32W108 can use up to four external interrupt sources (IRQA, IRQB, IRQC, and IRQD), each with its own top level NVIC interrupt vector. Since these external interrupt sources connect to the standard GPIO input path, an external interrupt pin may simultaneously be used by a peripheral device or even configured as an output. Analog mode is the only GPIO configuration that is not compatible with using a pin as an external interrupt.

External interrupts have individual triggering and filtering options selected using the registers GPIO_INTCFGA, GPIO_INTCFGB, GPIO_INTCFGC, and GPIO_INTCFGD. The bit field GPIO_INTMOD of the GPIO_INTCFGx register enables IRQx's second level interrupt and selects the triggering mode: 0 is disabled; 1 for rising edge; 2 for falling edge; 3 for both edges; 4 for active high level; 5 for active low level. The minimum width needed to latch an unfiltered external interrupt in both level- and edge-triggered mode is 80 ns. With the digital filter enabled (the GPIO_INTFILT bit in the GPIO_INTCFGx register is set), the minimum width needed is 450 ns.

The register INT_GPIOFLAG is the second-level interrupt flag register that indicates pending external interrupts. Writing 1 to a bit in the INT_GPIOFLAG register clears the flag while writing 0 has no effect. If the interrupt is level-triggered, the flag bit is set again immediately after being cleared if its input is still in the active state.

Two of the four external interrupts, IRQA and IRQB, have fixed pin assignments. The other two external interrupts, IRQC and IRQD, can use any GPIO pin. The GPIO_IRQCSEL and GPIO_IRQDSEL registers specify the GPIO pins assigned to IRQC and IRQD, respectively. *Table 25* shows how the GPIO_IRQCSEL and GPIO_IRQDSEL register values select the GPIO pin used for the external interrupt.

Table 25. IRQC/D GPIO selection

GPIO_IRQxSEL	GPIO	GPIO_IRQxSEL	GPIO	GPIO_IRQxSEL	GPIO
0	PA0	8	PB0	16	PC0
1	PA1	9	PB1	17	PC1
2	PA2	10	PB2	18	PC2
3	PA3	11	PB3	19	PC3

Table 25. IRQC/D GPIO selection (continued)

GPIO_IRQxSEL	GPIO	GPIO_IRQxSEL	GPIO	GPIO_IRQxSEL	GPIO
4	PA4	12	PB4	20	PC4
5	PA5	13	PB5	21	PC5
6	PA6	14	PB6	22	PC6
7	PA7	15	PB7	23	PC7

In some cases, it may be useful to assign IRQC or IRQD to an input also in use by a peripheral, for example to generate an interrupt from the slave select signal (nSSEL) in an SPI slave mode interface.

Refer to *Section 12: Interrupts on page 177* for further information regarding the STM32W108 interrupt system.

8.3 Debug control and status

Two GPIO registers are largely concerned with debugger functions. GPIO_DBGCFG can disable debugger operation, but has other miscellaneous control bits as well. GPIO_DBGSTAT, a read-only register, returns status related to debugger activity (GPIO_FORCEDBG and GPIO_SWEN), as well a flag (GPIO_BOOTMODE) indicating whether nBOOTMODE was asserted at the last power-on or NRST-based reset.

8.4 GPIO alternate functions

Table 26 lists the GPIO alternate functions.

Table 26. GPIO signal assignments

GPIO	Analog	Alternate function	Input	Output current drive
PA0		TIM2_CH1(1), SC2MOSI	TIM2_CH1(1), SC2MOSI	Standard
PA1		TIM2_CH3(1), SC2MISO, SC2SDA	TIM2_CH3(1), SC2MISO, SC2SDA	Standard
PA2		TIM2_CH4(1), SC2SCLK, SC2SCL	TIM2_CH4(1), SC2SCLK	Standard
PA3		TIM2_CH2(1), TRACECLK	TIM2_CH2(1), SC2nSSEL	Standard
PA4	ADC4	PTI_EN, TRACEDATA2		Standard
PA5	ADC5	PTI_DATA, TRACEDATA3	nBOOTMODE(2)	Standard
PA6		TIM1_CH3	TIM1_CH3	High
PA7		TIM1_CH4, REG_EN (3)	TIM1_CH4	High

Table 26. GPIO signal assignments (continued)

GPIO	Analog	Alternate function	Input	Output current drive
PB0	VREF	TRACECLK	TIM1CLK, TIM2MSK, IRQA	Standard
PB1		TIM2_CH1(4), SC1TXD, SC1MOSI, SC1MISO, SC1SDA	TIM2_CH1(4), SC1SDA	Standard
PB2		TIM2_CH2(4), SC1SCLK	TIM2_CH2(4), SC1MISO, SC1MOSI, SC1SCL, SC1RXD	Standard
PB3		TIM2_CH3(4), SC1SCLK	TIM2_CH3(4), SC1SCLK, UART_CTS	Standard
PB4		TIM2_CH4(4), UART_RTS	TIM2_CH4(4), SC1nSSEL	Standard
PB5	ADC0		TIM2CLK, TIM1MSK	Standard
PB6	ADC1	TIM1_CH1	TIM1_CH1, IRQB	High
PB7	ADC2	TIM1_CH2	TIM1_CH2	High
PC0		TRACEDATA1	JRST(5)	High
PC1	ADC3	TRACEDATA0, SWO		Standard
PC2		JTDO(6), SWO		Standard
PC3			JTDI(5)	Standard
PC4		SWDIO(7)	SWDIO(7), JTMS(5)	Standard
PC5		TX_ACTIVE		Standard
PC6	OSC32B	nTX_ACTIVE		Standard
PC7	OSC32A		OSC32_EXT	Standard

1. Default signal assignment (not remapped).
2. Overrides during reset as an input with pull up.
3. Overrides after reset as an open-drain output.
4. Alternate signal assignment (remapped).
5. Overrides in JTAG mode as an input with pull up.
6. Overrides in JTAG mode as a push-pull output.
7. Overrides in Serial Wire mode as either a push-pull output, or a floating input, controlled by the debugger.

8.5 General-purpose input / output (GPIO) registers

8.5.1 Port x configuration register (Low) (GPIO_PxCFGL)

Address offset: 0xB000 (GPIO_PACFGL), 0xB400 (GPIO_PBCFGL) and 0xB800 (GPIO_PCCFGL)
Reset value: 0x0000 4444

Table 27. Port x configuration register (Low) (GPIO_PxCFGL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px3_CFG				Px2_CFG				Px1_CFG				Px0_CFG			
rw				rw				rw				rw			

Bits [15:12] Px3_CFG: GPIO configuration control.
0x0: Analog, input or output (GPIO_PxIN always reads 1).
0x1: Output, push-pull (GPIO_PxOUT controls the output).
0x4: Input, floating.
0x5: Output, open-drain (GPIO_PxOUT controls the output).
0x8: Input, pulled up or down (selected by GPIO_PxOUT: 0 = pull-down, 1 = pull-up).
0x9: Alternate output, push-pull (peripheral controls the output).
0xB: Alternate output SPI SCLK, push-pull (only for SPI master mode SCLK).
0xD: Alternate output, open-drain (peripheral controls the output).

Bits [11:8] Px2_CFG: GPIO configuration control: see Px3_CFG above.

Bits [7:4] Px1_CFG: GPIO configuration control: see Px3_CFG above.

Bits [3:0] Px0_CFG: GPIO configuration control: see Px3_CFG above.

8.5.2 Port x configuration register (High) (GPIO_PxCFGH)

Address offset: 0xB004 (GPIO_PACFGH), 0xB404 (GPIO_PBCFGH) and 0xB804 (GPIO_PCCFGH)
Reset value: 0x0000 4444

Table 28. Port x configuration register (High) (GPIO_PxCFGH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Px7_CFG				Px6_CFG				Px5_CFG				Px4_CFG			
rw				rw				rw				rw			

Bits [15:12] Px7_CFG: GPIO configuration control.
0x0: Analog, input or output (GPIO_PxIN always reads 1).
0x1: Output, push-pull (GPIO_PxOUT controls the output).
0x4: Input, floating.
0x5: Output, open-drain (GPIO_PxOUT controls the output).
0x8: Input, pulled up or down (selected by GPIO_PxOUT: 0 = pull-down, 1 = pull-up).
0x9: Alternate output, push-pull (peripheral controls the output).
0xB: Alternate output SPI SCLK, push-pull (only for SPI master mode SCLK).
0xD: Alternate output, open-drain (peripheral controls the output).

Bits [11:8] Px6_CFG: GPIO configuration control: see Px7_CFG above.

Bits [7:4] Px5_CFG: GPIO configuration control: see Px7_CFG above.

Bits [3:0] Px4_CFG: GPIO configuration control: see Px7_CFG above.

8.5.3 Port x input data register (GPIO_PxIN)

Address offset: 0xB008 (GPIO_PAIN), 0xB408 (GPIO_PBIN) and 0xB808 (GPIO_PCIN)
Reset value: 0x0000 0000

Table 29. Port x input data register (GPIO_PxIN)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Px7: Input level at pin Px7.

Bit 6 Px6: Input level at pin Px6.

Bit 5 Px5: Input level at pin Px5.

Bit 4 Px4: Input level at pin Px4.

Bit 3 Px3: Input level at pin Px3.

Bit 2 Px2: Input level at pin Px2.

Bit 1 Px1: Input level at pin Px1.

Bit 0 Px0: Input level at pin Px0.

8.5.4 Port x output data register (GPIO_PxOUT)

Address offset: 0xB00C (GPIO_PAOUT), 0xB40C (GPIO_PBOUT) and 0xB80C (GPIO_PCOUT)
Reset value: 0x0000 0000

Table 30. Port x output data register (GPIO_PxOUT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Px7: Output data for Px7.

Bit 6 Px6: Output data for Px6.

Bit 5 Px5: Output data for Px5.

Bit 4 Px4: Output data for Px4.

Bit 3 Px3: Output data for Px3.

Bit 2 Px2: Output data for Px2.

Bit 1 Px1: Output data for Px1.

Bit 0 Px0: Output data for Px0.

8.5.5 Port x output clear register (GPIO_PxCLR)

Address offset: 0xB014 (GPIO_PACLR), 0xB414 (GPIO_PBCLR) and 0xB814 (GPIO_PCCLR)
Reset value: 0x0000 0000

Table 31. Port x output clear register (GPIO_PxCLR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
								w	w	w	w	w	w	w	w

Bit 7 Px7: Write 1 to clear the output data bit for Px7 (writing 0 has no effect).

Bit 6 Px6: Write 1 to clear the output data bit for Px6 (writing 0 has no effect).

Bit 5 Px5: Write 1 to clear the output data bit for Px5 (writing 0 has no effect).

Bit 4 Px4: Write 1 to clear the output data bit for Px4 (writing 0 has no effect).

Bit 3 Px3: Write 1 to clear the output data bit for Px3 (writing 0 has no effect).

Bit 2 Px2: Write 1 to clear the output data bit for Px2 (writing 0 has no effect).

Bit 1 Px1: Write 1 to clear the output data bit for Px1 (writing 0 has no effect).

Bit 0 Px0: Write 1 to clear the output data bit for Px0 (writing 0 has no effect).

8.5.6 Port x output set register (GPIO_PxSET)

Address offset: 0xB010 (GPIO_PASET), 0xB410 (GPIO_PBSET) and 0xB810 (GPIO_PCSET)

Reset value: 0x0000 0000

Table 32. Port x output set register (GPIO_PxSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits [15:8] Reserved: these bits must be set to 0.

Bit 7 Px7: Write 1 to set the output data bit for Px7 (writing 0 has no effect).

Bit 6 Px6: Write 1 to set the output data bit for Px6 (writing 0 has no effect).

Bit 5 Px5: Write 1 to set the output data bit for Px5 (writing 0 has no effect).

Bit 4 Px4: Write 1 to set the output data bit for Px4 (writing 0 has no effect).

Bit 3 Px3: Write 1 to set the output data bit for Px3 (writing 0 has no effect).

Bit 2 Px2: Write 1 to set the output data bit for Px2 (writing 0 has no effect).

Bit 1 Px1: Write 1 to set the output data bit for Px1 (writing 0 has no effect).

Bit 0 Px0: Write 1 to set the output data bit for Px0 (writing 0 has no effect).

8.5.7 Port x wakeup monitor register (GPIO_PxWAKE)

Address offset: 0xBC08 (GPIO_PAWAKE), 0xBC0C (GPIO_PBWAKE) and 0xBC10 (GPIO_PCWAKE)

Reset value: 0x0000 0000

Table 33. Port x wakeup monitor register (GPIO_PxWAKE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								Px7	Px6	Px5	Px4	Px3	Px2	Px1	Px0
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 7 Px7: Write 1 to enable wakeup monitoring of Px7.

Bit 6 Px6: Write 1 to enable wakeup monitoring of Px6.

Bit 5 Px5: Write 1 to enable wakeup monitoring of Px5.

Bit 4 Px4: Write 1 to enable wakeup monitoring of Px4.

Bit 3 Px3: Write 1 to enable wakeup monitoring of Px3.

Bit 2 Px2: Write 1 to enable wakeup monitoring of Px2.

Bit 1 Px1: Write 1 to enable wakeup monitoring of Px1.

Bit 0 Px0: Write 1 to enable wakeup monitoring of Px0.

8.5.8 GPIO wakeup filtering register (GPIO_WAKEFILT)

Address offset: 0xBC0C
Reset value: 0x0000 0000

Table 34. GPIO wakeup filtering register (GPIO_WAKEFILT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												IRQD_WAKE_FILTER	SC2_WAKE_FILTER	SC1_WAKE_FILTER	GPIO_WAKE_FILTER
												rw	rw	rw	rw

Bit 3 IRQD_WAKE_FILTER: Enable filter on GPIO wakeup source IRQD.

Bit 2 SC2_WAKE_FILTER: Enable filter on GPIO wakeup source SC2 (PA2).

Bit 1 SC1_WAKE_FILTER: Enable filter on GPIO wakeup source SC1 (PB2).

Bit 0 GPIO_WAKE_FILTER: Enable filter on GPIO wakeup sources enabled by the GPIO_PnWAKE registers.

8.5.9 Interrupt x select register (GPIO_IRQxSEL)

Address offset: 0xBC14 (GPIO_IRQCSEL) and 0xBC18 (GPIO_IRQDSEL)
Reset value: 0x0000 000F (GPIO_IRQCSEL) and 0x0000 0010 (GPIO_IRQDSEL)

Table 35. Interrupt x select register (GPIO_IRQxSEL)

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

Bits [4:0] SEL_GPIO: Pin assigned to IRQx.

0x00: PA0.
0x01: PA1.
0x02: PA2.
0x03: PA3.
0x04: PA4.
0x05: PA5.
0x06: PA6.
0x07: PA7.
0x08: PB0.
0x09: PB1.
0x0A: PB2.
0x0B: PB3.
0x0C: PB4.
0x0D: PB5.
0x0E: PB6.
0x0F: PB7.
0x10: PC0.
0x11: PC1.
0x12: PC2.
0x13: PC3.
0x14: PC4.
0x15: PC5.
0x16: PC6.
0x17: PC7.
0x18 - 0x1F: Reserved.

8.5.10 GPIO interrupt x configuration register (GPIO_INTCFGx)

Address offset: 0xA860 (GPIO_INTCFGA), 0xA864 (GPIO_INTCFGB), 0xA868 (GPIO_INTCFGC) and 0xA86C (GPIO_INTCFGD)
Reset value: 0x0000 0000

Table 36. GPIO interrupt x configuration register (GPIO_INTCFGx)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							GPIO_INTFILT	GPIO_INTMOD			Reserved				
							rw	rw							

Bit [8] GPIO_INTFILT: Set this bit to enable digital filtering on IRQx.

Bits [7:5] GPIO_INTMOD: IRQx triggering mode.
0x0: Disabled.0x4: Active high level triggered.
0x1: Rising edge triggered.0x5: Active low level triggered.
0x2: Falling edge triggered.0x6, 0x7: Reserved.
0x3: Rising and falling edge triggered.

8.5.11 GPIO interrupt flag register (INT_GPIOFLAG)

Address offset: 0xA814
Reset value: 0x0000 0000

Table 37. GPIO interrupt flag register (INT_GPIOFLAG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												INT_IRQDFLAG	INT_IRQCFLAG	INT_IRQBFLAG	INT_IRQAFLAG
												rw	rw	rw	rw

Bit 3 INT_IRQDFLAG: IRQD interrupt pending.

Bit 2 INT_IRQCFLAG: IRQC interrupt pending.

Bit 1 INT_IRQBFLAG: IRQB interrupt pending.

Bit 0 INT_IRQAFLAG: IRQA interrupt pending.

8.5.12 GPIO debug configuration register (GPIO_DBGCFG)

Address offset: 0xBC00
Reset value: 0x0000 0010

Table 38. GPIO debug configuration register (GPIO_DBGCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										GPIO_DEBUGDIS	GPIO_EXTREGEN	Re-served			
										rw	rw				

Bit 5 GPIO_DEBUGDIS: Disable debug interface override of normal GPIO configuration.
0: Permit debug interface to be active.
1: Disable debug interface (if it is not already active).

Bit 4 GPIO_EXTREGEN: : Disable REG_EN override of PA7's normal GPIO configuration.
0: Enable override.
1: Disable override.

Bit 3 Reserved: this bit can change during normal operation. When writing to GPIO_DBGCFG, the value of this bit must be preserved.

8.5.13 GPIO debug status register (GPIO_DBGSTAT)

Address offset: 0xBC04
Reset value: 0x0000 0000

Table 39. GPIO debug status register (GPIO_DBGSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
r															
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved												GPIO_BOOTMODE	Reserved	GPIO_FORCEDBG	GPIO_SWEN
												r		r	r

Bit 3 GPIO_BOOTMODE: The state of the nBOOTMODE signal sampled at the end of reset.
0: nBOOTMODE was not asserted (it read high).
1: nBOOTMODE was asserted (it read low).

Bit 1 GPIO_FORCEDBG: Status of debugger interface.
0: Debugger interface not forced active.
1: Debugger interface forced active by debugger cable.

Bit 0 GPIO_SWEN: Status of Serial Wire interface.
0: Not enabled by SWJ-DP.
1: Enabled by SWJ-DP.

9 Serial interfaces

9.1 Functional description

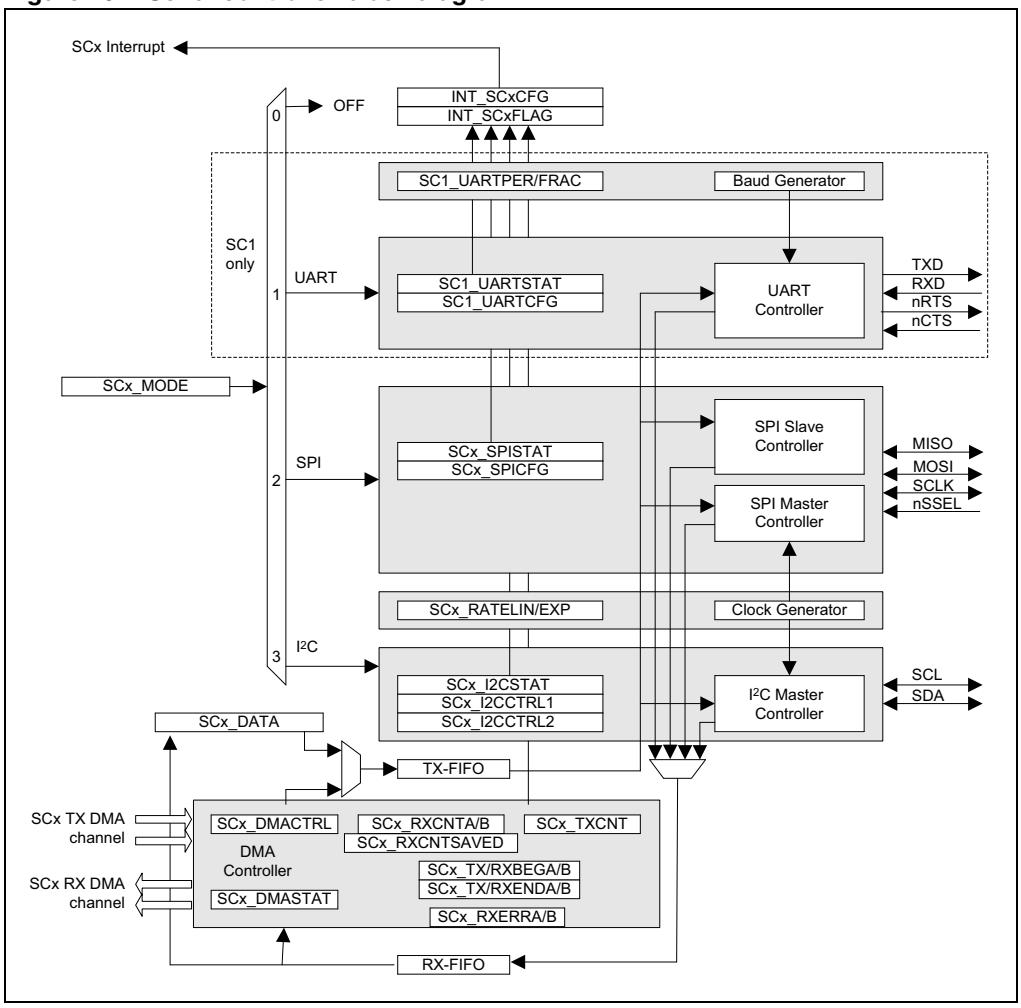
The STM32W108 has two serial controllers, SC1 and SC2, which provide several options for full-duplex synchronous and asynchronous serial communications.

- SPI (Serial Peripheral Interface), master or slave
- I^2C (Inter-Integrated Circuit), master only
- UART (Universal Asynchronous Receiver/Transmitter), SC1 only
- Receive and transmit FIFOs and DMA channels, SPI and UART modes

Receive and transmit FIFOs allow faster data speeds using byte-at-a-time interrupts. For the highest SPI and UART speeds, dedicated receive and transmit DMA channels reduce CPU loading and extend the allowable time to service a serial controller interrupt. Polled operation is also possible using direct access to the serial data registers. *Figure 10* shows the components of the serial controllers.

Note: The notation SCx means that either SC1 or SC2 may be substituted to form the name of a specific register or field within a register.

Figure 10. Serial controller block diagram



9.2 Configuration

Before using a serial controller, it should be configured and initialized as follows:

1. Set up the parameters specific to the operating mode (master/slave for SPI, baud rate for UART, etc.).
2. Configure the GPIO pins used by the serial controller as shown in *Table 40* and *Table 41*. *Section 8.1.2: Configuration on page 59* shows how to configure GPIO pins."If using DMA, set up the DMA and buffers. This is described fully in *Section 9.13: DMA channel registers on page 103*.
3. If using interrupts, select edge- or level-triggered interrupts with the SCx_INTMODE register, enable the desired second-level interrupt sources in the INT_SCxCFG register, and finally enable the top-level SCx interrupt in the NVIC.
4. Write the serial interface operating mode - SPI, I^2C, or UART - to the SCx_MODE register.

Table 40. SC1 GPIO usage and configuration

Interface	PB1	PB2	PB3	PB4
SPI - Master	SC1MOSI alternate output (push-pull)	SC1MISO input	SC1SCLK alternate output (push-pull); special SCLK mode	(not used)
SPI - Slave	SC1MISO alternate output (push-pull)	SC1MOSI input	SC1SCLK input	SC1nSSEL input
I^2C - Master	SC1SDA alternate output (open-drain)	SC1SCL alternate output (open-drain)	(not used)	(not used)
UART	TXD alternate output (push-pull)	RXD input	nCTS input (1)	nRTS alternate output (push-pull) (1)

1. used if RTS/CTS hardware flow control is enabled.

Table 41. SC2 GPIO usage and configuration

Interface	PA0	PA1	PA2	PA3
SPI - Master	SC2MOSI Alternate Output (push-pull)	SC2MISO Input	SC2SCLK Alternate Output (push-pull), special SCLK mode	(not used)
SPI - Slave	SC2MOSI Alternate Output (push-pull)	SC2MISO Input	SC2SCLK Input	SC2nSSEL Input
I^2C - Master	(not used)	SC2SDA Alternate Output (open-drain)	SC2SCL Alternate Output (open-drain)	(not used)

9.3 SPI master mode

The SPI master controller has the following features:

- Full duplex operation
- Programmable clock frequency (6 MHz max. for STM32W108xB and 12 MHz max for STM32W108CC and STM32W108CZ)
- Programmable clock polarity and phase
- Selectable data shift direction (either LSB or MSB first)
- Receive and transmit FIFOs
- Receive and transmit DMA channels

The SPI master controller uses the three signals:

- MOSI (Master Out, Slave In) - outputs serial data from the master
- MISO (Master In, Slave Out) - inputs serial data from a slave
- SCLK (Serial Clock) - outputs the serial clock used by MOSI and MISO

The GPIO pins used for these signals are shown in *Table 42*. Additional outputs may be needed to drive the nSSEL signals on slave devices.

Table 42. SPI master GPIO usage

Parameter	MOSI	MISO	SCLK
Direction	Output	Input	Output
GPIO configuration	Alternate Output (push-pull)	Input	Alternate Output (push-pull) Special SCLK mode
SC1 pin	PB1	PB2	PB3
SC2 pin	PA0	PA1	PA2

9.3.1 Setup and configuration

Both serial controllers, SC1 and SC2, support SPI master mode. SPI master mode is enabled by the following register settings:

- The serial controller mode register (SCx_MODE) is '2'.
- The SC_SPIMST bit in the SPI configuration register (SCx_SPICFG) is '1'.
- The SC_TWIACK bit in the I^2C control register (SCx_TWICTRL2) is '1'.

The SPI serial clock (SCLK) is produced by a programmable clock generator. The serial clock is produced by dividing down 12 MHz according to this equation:

$$Rate = \frac{12MHz}{(LIN+1)x2^{EXP}}$$

EXP is the value written to the SCx_RATEEXP register and LIN is the value written to the SCx_RATELIN register. The SPI master mode clock may not exceed 6 Mbps, so EXP and LIN cannot both be zero.

The SPI master controller supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see *SPI master mode formats on page 78*). The bits SC_SPIPOL, SC_SPIPHA, and SC_SPIORD are defined within the SCx_SPICFG register.

Table 43. SPI master mode formats

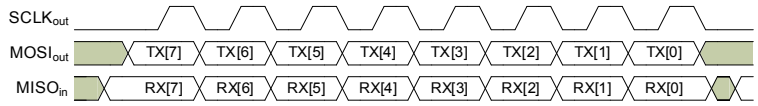
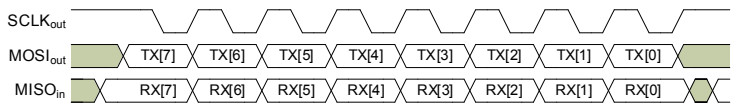
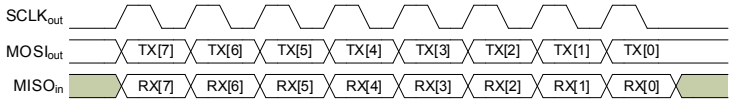
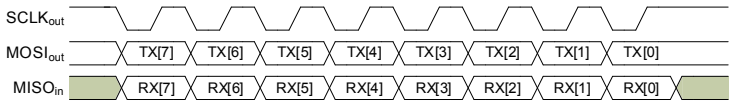
SCx_SPICFG				Frame formats
SC_SPIxxx(1)				
MST	ORD	PHA	POL	
1	0	0	0	
1	0	0	1	

Table 43. SPI master mode formats (continued)

SCx_SPICFG				Frame formats
SC_SPIxxx(1)				
MST	ORD	PHA	POL	
1	0	1	0	
1	0	1	1	
1	1	-	-	Same as above except data is sent LSB first instead of MSB first.

1. The notation xxx means that the corresponding column header below is inserted to form the field name.

9.3.2 Operation

Characters transmitted and received by the SPI master controller are buffered in transmit and receive FIFOs that are both 4 entries deep. When software writes a character to the SCx_DATA register, the character is pushed onto the transmit FIFO. Similarly, when software reads from the SCx_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, they also write to and read from the transmit and receive FIFOs.

When the transmit FIFO and the serializer are both empty, writing a character to the transmit FIFO clears the SC_SPITXIDLE bit in the SCx_SPISTAT register. This indicates that some characters have not yet been transmitted. If characters are written to the transmit FIFO until it is full, the SC_SPITXFREE bit in the SCx_SPISTAT register is cleared. Shifting out a character to the MOSI pin sets the SC_SPITXFREE bit in the SCx_SPISTAT register. When the transmit FIFO empties and the last character has been shifted out, the SC_SPITXIDLE bit in the SCx_SPISTAT register is set.

Characters received are stored in the receive FIFO. Receiving characters sets the SC_SPIRXVAL bit in the SCx_SPISTAT register, indicating that characters can be read from the receive FIFO. Characters received while the receive FIFO is full are dropped, and the SC_SPIRXOVF bit in the SCx_SPISTAT register is set. The receive FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the receive FIFO is drained. Once the DMA marks a receive error, two conditions will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SCx_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

To receive a character, you must transmit a character. If a long stream of receive characters is expected, a long sequence of dummy transmit characters must be generated. To avoid software or transmit DMA initiating these transfers and consuming unnecessary bandwidth, the SPI serializer can be instructed to retransmit the last transmitted character or to transmit a busy token (0xFF), which is determined by the SC_SPIRPT bit in the SCx_SPICFG register. This functionality can only be enabled or disabled when the transmit FIFO is empty and the transmit serializer is idle, indicated by a cleared SC_SPITXIDLE bit in the SCx_SPISTAT register.

Every time an automatic character transmission starts, a transmit underrun is detected as there is no data in transmit FIFO, and the INT_SCTXUND bit in the INT_SC2FLAG register is set. After automatic character transmission is disabled, no more new characters are received. The receive FIFO holds characters just received.

Note: The Receive DMA complete event does not always mean the receive FIFO is empty.

The DMA Channels section describes how to configure and use the serial receive and transmit DMA channels.

9.3.3 Interrupts

SPI master controller second level interrupts are generated by the following events:

- Transmit FIFO empty and last character shifted out (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPIRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Received and lost character while receive FIFO was full (receive overrun error)
- Transmitted character while transmit FIFO was empty (transmit underrun error)

To enable CPU interrupts, set the desired interrupt bits in the second level INT_SCxCFG register, and enable the top level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

9.4 SPI slave mode

Both SC1 and SC2 SPI controllers include a SPI slave controller with these features:

- Full duplex operation
- Up to 5 Mbps data transfer rate
- Programmable clock polarity and clock phase
- Selectable data shift direction (either LSB or MSB first)
- Slave select input

The SPI slave controller uses four signals:

- MOSI (Master Out, Slave In) - inputs serial data from the master
- MISO (Master In, Slave Out) - outputs serial data to the master
- SCLK (Serial Clock) - clocks data transfers on MOSI and MISO
- nSSEL (Slave Select) - enables serial communication with the slave

The GPIO pins that can be assigned to these signals are shown in *Table 44*.

Table 44. SPI slave GPIO usage

Parameter	MOSI	MISO	SCLK	nSSEL
Direction	Input	Output	Input	Input
GPIO configuration	Input	Alternate Output (push-pull)	Input	Input
SC1 pin	PB2	PB1	PB3	PB4
SC2 pin	PA0	PA1	PA2	PA3

9.4.1 Setup and configuration

Both serial controllers, SC1 and SC2, support SPI slave mode. SPI slave mode is enabled by the following register settings:

- The serial controller mode register, SCx_MODE, is '2'.
- The SC_SPIMST bit in the SPI configuration register, SCx_SPICFG, is '0'.

The SPI slave controller receives its clock from an external SPI master device and supports rates up to 5 Mbps.

The SPI slave controller supports various frame formats depending upon the clock polarity (SC_SPIPOL), clock phase (SC_SPIPHA), and direction of data (SC_SPIORD) (see Table 8 6). The SC_SPIPOL, SC_SPIPHA, and SC_SPIORD bits are defined within the SCx_SPICFG registers.

Table 45. SPI slave mode formats

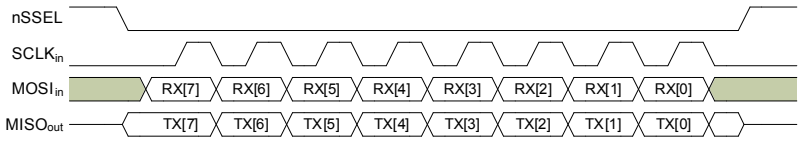
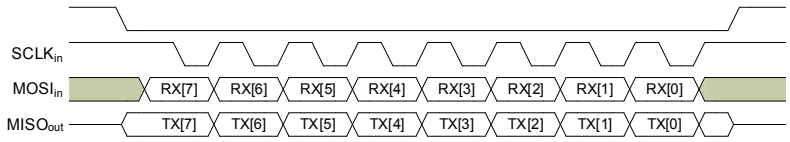
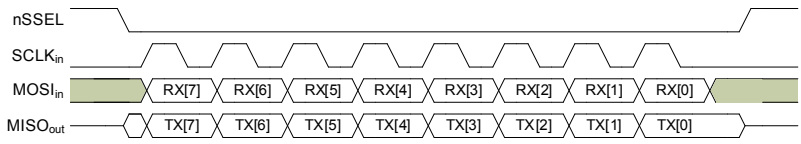
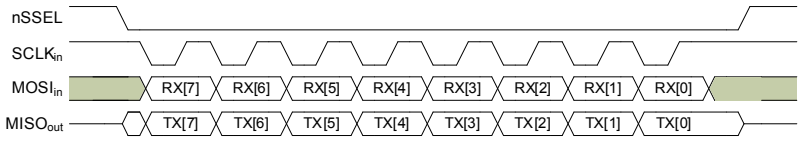
SCx_SPICFG				Frame format
SC_SPIxxx(1)				
MST	ORD	PHA	POL	
0	0	0	0	
0	0	0	1	
0	0	1	0	

Table 45. SPI slave mode formats (continued)

SCx_SPICFG				Frame format
SC_SPIxxx[1]				
MST	ORD	PHA	POL	
0	0	1	1	
0	1	-	-	Same as above except LSB first instead of MSB first.

1. The notation xxx means that the corresponding column header below is inserted to form the field name.

9.4.2 Operation

When the slave select (nSSEL) signal is asserted by the master, SPI transmit data is driven to the output pin MISO, and SPI data is received from the input pin MOSI. The nSSEL pin has to be asserted to enable the transmit serializer to drive data to the output signal MISO. A falling edge on nSSEL resets the SPI slave shift registers.

Characters transmitted and received by the SPI slave controller are buffered in the transmit and receive FIFOs that are both 4 entries deep. When software writes a character to the SCx_DATA register, it is pushed onto the transmit FIFO. Similarly, when software reads from the SCx_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, the DMA channels also write to and read from the transmit and receive FIFOs.

Characters received are stored in the receive FIFO. Receiving characters sets the SC_SPIRXVAL bit in the SCx_SPISTAT register, to indicate that characters can be read from the receive FIFO. Characters received while the receive FIFO is full are dropped, and the SC_SPIRXOVF bit in the SCx_SPISTAT register is set. The receive FIFO hardware generates the INT_SCRXOVF interrupt, but the DMA register will not indicate the error condition until the receive FIFO is drained. Once the DMA marks a receive error, two conditions will clear the error indication: setting the appropriate SC_TX/RXDMARST bit in the SCx_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

Receiving a character causes the serial transmission of a character pulled from the transmit FIFO. When the transmit FIFO is empty, a transmit underrun is detected (no data in transmit FIFO) and the INT_SCTXUND bit in the INT_SCxFLAG register is set. Because no character is available for serialization, the SPI serializer retransmits the last transmitted character or a busy token (0xFF), determined by the SC_SPIRPT bit in the SCx_SPICFG register.

When the transmit FIFO and the serializer are both empty, writing a character to the transmit FIFO clears the SC_SPITXIDLE bit in the SCx_SPISTAT register. This indicates that not all characters have been transmitted. If characters are written to the transmit FIFO until it is full, the SC_SPITXFREE bit in the SCx_SPISTAT register is cleared. Shifting out a transmit character to the MISO pin causes the SC_SPITXFREE bit in the SCx_SPISTAT register to get set. When the transmit FIFO empties and the last character has been shifted out, the SC_SPITXIDLE bit in the SCx_SPISTAT register is set.

The SPI slave controller must guarantee that there is time to move new transmit data from the transmit FIFO into the hardware serializer. To provide sufficient time, the SPI slave controller inserts a byte of padding at the start of every new string of transmit data. After slave select asserts and the SC_SPIRXVAL bit in the SCx_SPISTAT register gets set at least once, the following operation holds true until slave select deasserts. Whenever the transmit FIFO is empty and data is placed into the transmit FIFO, either manually or through DMA, the SPI hardware inserts a byte of padding onto the front of the transmission as if this byte was placed there by software. The value of the byte of padding that is inserted is selected by the SC_SPIRPT bit in the SCx_SPICFG register.

9.4.3 DMA

The DMA Channels section describes how to configure and use the serial receive and transmit DMA channels.

When using the receive DMA channel and nSSEL transitions to the high (deasserted) state, the active buffer's receive DMA count register (SCx_RXCNTA/B) is saved in the SCx_RXCNTSAVED register. SCx_RXCNTSAVED is only written the first time nSSEL goes high after a buffer has been loaded. Subsequent rising edges set a status bit but are otherwise ignored. The 3-bit field SC_RXSSEL in the SCx_DMASTAT register records what, if anything, was saved to the SCx_RXCNTSAVED register, and whether or not another rising edge occurred on nSSEL.

9.4.4 Interrupts

SPI slave controller second level interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXIDLE)
- Transmit FIFO changed from full to not full (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPITXFREE)
- Receive FIFO changed from empty to not empty (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC_SPIRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Received and lost character while receive FIFO was full (receive overrun error)
- Transmitted character while transmit FIFO was empty (transmit underrun error)

To enable CPU interrupts, set desired interrupt bits in the second level INT_SCxCFG register, and also enable the top level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

9.5 Inter-integrated circuit interfaces (I^2C)

Both STM32W108 serial controllers SC1 and SC2 include an Inter-integrated circuit interface (I^2C) master controller with the following features:

- Uses only two bidirectional GPIO pins
- Programmable clock frequency (up to 400 kHz)
- Supports both 7-bit and 10-bit addressing
- Compatible with Philips' I^2C-bus slave devices

The I^{2}C master controller uses just two signals:

- SDA (Serial Data) - bidirectional serial data
- SCL (Serial Clock) - bidirectional serial clock

Table 46 lists the GPIO pins used by the SC1 and SC2 I^{2}C master controllers. Because the pins are configured as open-drain outputs, they require external pull-up resistors.

Table 46. I^{2}C Master GPIO Usage

Parameter	SDA	SCL
Direction	Input / Output	Input / Output
GPIO configuration	Alternate Output (open drain)	Alternate Output (open drain)
SC1 pin	PB1	PB2
SC2 pin	PA1	PA2

9.5.1 Setup and configuration

The I^{2}C controller is enabled by writing 3 to the SCx_MODE register. The I^{2}C controller operates only in master mode and supports both Standard (100 kbps) and Fast (400 kbps) I^{2}C modes. Address arbitration is not implemented, so multiple master applications are not supported.

The I^{2}C master controller's serial clock (SCL) is produced by a programmable clock generator. SCL is produced by dividing down 12 MHz according to this equation:

$$Rate = \frac{12MHz}{(LIN + 1)x2^{EXP}}$$

EXP is the value written to the SCx_RATEEXP register and LIN is the value written to the SCx_RATELIN register. *I2C clock rate programming on page 84* shows the rate settings for Standard-Mode I^{2}C (100 kbps) and Fast-Mode I^{2}C (400 kbps) operation.

Table 47. I^{2}C clock rate programming

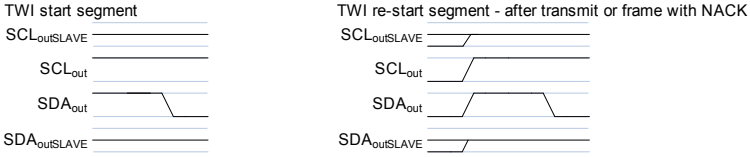
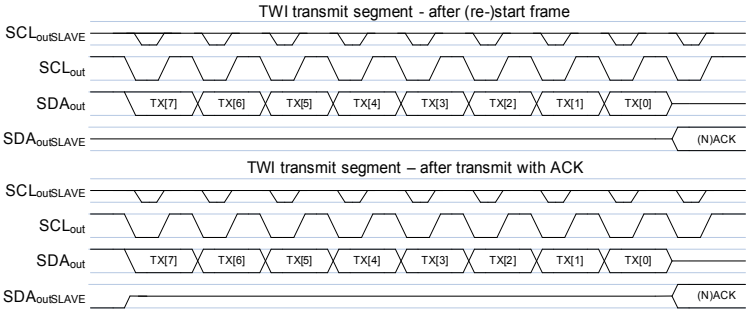
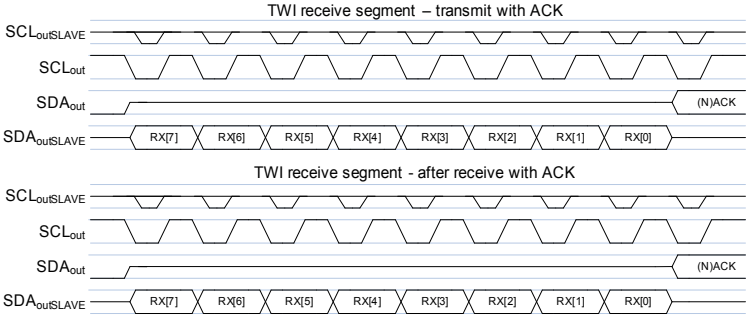
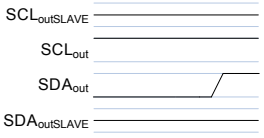
Clock rate	SCx_RATELIN	SCx_RATEEXP
100 kbps	14	3
375 kbps	15	1
400 kbps	14	1

Note: At 400 kbps, the Philips I^{2}C Bus specification requires the minimum low period of SCL to be 1.3 µs, but on the STM32W108 it is 1.25 µs. If a slave device requires strict compliance with SCL timing, the clock rate must be lowered to 375 kbps.

9.5.2 Constructing frames

The I^{2}C master controller supports generating various frame segments by means of the SC_TWISTART, SC_TWISTOP, SC_TWISEND, and SC_TWIRECV bits in the SCx_TWICTRL1 registers. *Figure 48* summarizes these frames.

Table 48. I²C master frame segments

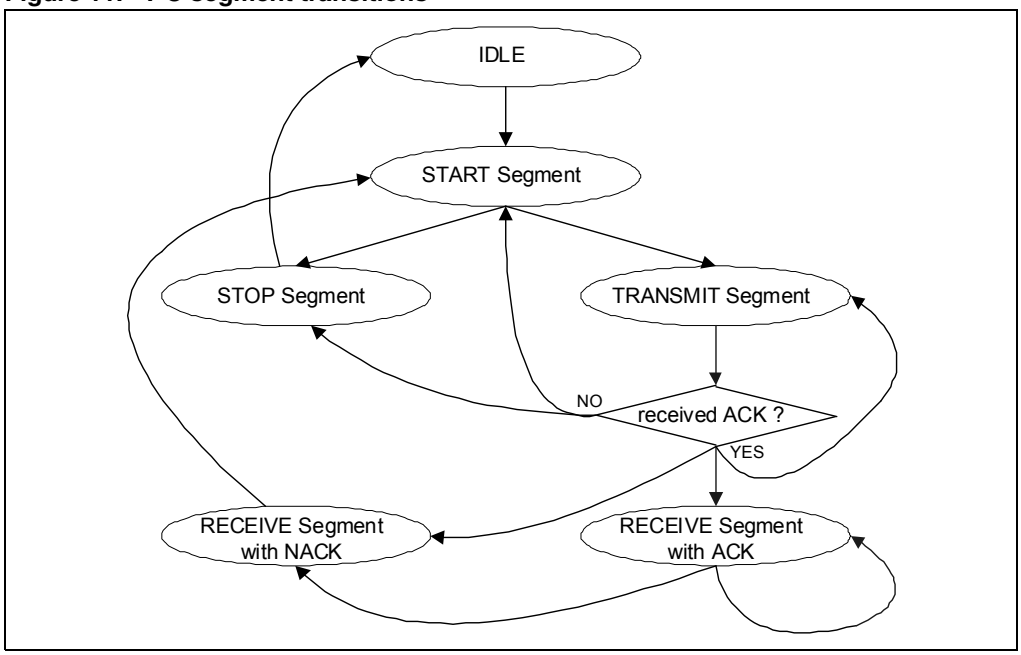
SCx_TWICTRL1				Frame segments
SC_TWIxxxx(1)				
START	SEND	RECV	STOP	
1	0	0	0	TWI start segment / TWI re-start segment - after transmit or frame with NACK 
0	1	0	0	TWI transmit segment - after (re-)start frame TWI transmit segment – after transmit with ACK 
0	0	1	0	TWI receive segment – transmit with ACK TWI receive segment - after receive with ACK 
0	0	0	1	TWI stop segment - after frame with NACK or stop 
0	0	0	0	No pending frame segment
1 - - 1	1 1 - -	- 1 1 -	- - 1 1	Illegal

1. The notation xxx means that the corresponding column header below is inserted to form the field name.

Full I²C frames have to be constructed by software from individual I²C segments. All necessary segment transitions are shown in *Figure 11*. ACK or NACK generation of an I²C

receive frame segment is determined with the SC_TWIACK bit in the SCx_TWICTRL2 register.

Figure 11. I²C segment transitions



Generation of a 7-bit address is accomplished with one transmit segment. The upper 7 bits of the transmitted character contain the 7-bit address. The remaining lower bit contains the command type ("read" or "write").

Generation of a 10-bit address is accomplished with two transmit segments. The upper 5 bits of the first transmit character must be set to 0x1E. The next 2 bits are for the 2 most significant bits of the 10-bit address. The remaining lower bit contains the command type ("read" or "write"). The second transmit segment is for the remaining 8 bits of the 10-bit address.

Transmitted and received characters are accessed through the SCx_DATA register.

To initiate (re)start and stop segments, set the SC_TWISTART or SC_TWISTOP bit in the SCx_TWICTRL1 register, then wait until the bit is clear. Alternatively, the SC_TWICMDFIN bit in the SCx_TWISTAT can be used for waiting.

To initiate a transmit segment, write the data to the SCx_DATA data register, then set the SC_TWISEND bit in the SCx_TWICTRL1 register, and finally wait until the bit is clear. Alternatively the SC_TWITXFIN bit in the SCx_TWISTAT register can be used for waiting.

To initiate a receive segment, set the SC_TWIRECV bit in the SCx_TWICTRL1 register, wait until it is clear, and then read from the SCx_DATA register. Alternatively, the SC_TWIRXFIN bit in the SCx_TWISTAT register can be used for waiting. Now the SC_TWIRXNAK bit in the SCx_TWISTAT register indicates if a NACK or ACK was received from an I²C slave device.

9.5.3 Interrupts

I^2C master controller interrupts are generated on the following events:

- Bus command (SC_TWISTART/SC_TWISTOP) completed (0 to 1 transition of SC_TWICMDFIN)
- Character transmitted and slave device responded with NACK
- Character transmitted (0 to 1 transition of SC_TWITXFIN)
- Character received (0 to 1 transition of SC_TWIRXFIN)
- Received and lost character while receive FIFO was full (receive overrun error)
- Transmitted character while transmit FIFO was empty (transmit underrun error)

To enable CPU interrupts, set the desired interrupt bits in the second level INT_SCxCFG register, and enable the top level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

9.6 Universal asynchronous receiver / transmitter (UART)

The SC1 UART is enabled by writing 1 to SC1_MODE. The SC2 serial controller does not include UART functions.

The UART supports the following features:

- Flexible baud rate clock (300 bps to 921.6 bps)
- Data bits (7 or 8)
- Parity bits (none, odd, or even)
- Stop bits (1 or 2)
- False start bit and noise filtering
- Receive and transmit FIFOs
- Optional RTS/CTS flow control
- Receive and transmit DMA channels

The UART uses two signals to transmit and receive serial data:

- TXD (Transmitted Data) - serial data received by the STM32W108
- RXD (Received Data) - serial data sent by the STM32W108

If RTS/CTS flow control is enabled, these two signals are also used:

- nRTS (Request To Send) - indicates the STM32W108 is able to receive data RXD
- nCTS (Clear To Send) - inhibits sending data from the STM32W108 if not asserted

The GPIO pins assigned to these signals are shown in *Table 49*.

Table 49. UART GPIO usage

Parameter	TXD	RXD	nCTS(1)	nRTS(1)
Direction	Output	Input	Input	Output
GPIO configuration	Alternate Output (push-pull)	Input	Input	Alternate Output (push-pull)
SC1 pin	PB1	PB2	PB3	PB4

1. Only used if RTS/CTS hardware flow control is enabled.

9.6.1 Setup and configuration

The UART baud rate clock is produced by a programmable baud generator starting from the 24 Hz clock:

$$baud = \frac{24MHz}{2N + F}$$

The integer portion of the divisor, N, is written to the SC1_UARTPER register and the fractional part, F, to the SC1_UARTFRAC register. *Table 50* shows the values used to generate some common baud rates and their associated clock frequency error. The UART requires an internal clock that is at least eight times the baud rate clock, so the minimum allowable setting for SC1_UARTPER is '8'.

Table 50. UART baud rate divisors for common baud rates

Baud rate (bits/sec)	SC1_UARTPER	SC1_UARTFRAC	Baud rate error (%)
300	40000	0	0
2400	5000	0	0
4800	2500	0	0
9600	1250	0	0
19200	625	0	0
38400	312	1	0
57600	208	1	- 0.08
115200	104	0	+ 0.16
230400	52	0	+ 0.16
460800	26	0	+ 0.16
921600	13	0	+ 0.16

Note: The UART may receive corrupt bytes if the interbyte gap is long or there is a baud rate mismatch between receive and transmit. The UART may detect a parity and/or framing error on the corrupt byte, but there will not necessarily be any error detected. As a result, the device should be operated in systems where the other side of the communication link also uses a crystal as its timing reference, and baud rates should be selected to minimize the baud rate mismatch to the crystal tolerance. UART protocols should contain some form of error checking (e.g. CRC) at the packet level to detect, and retry in the event of errors.

The UART character frame format is determined by three bits in the SC1_UARTCFG register:

- SC1_UART2STP selects the number of stop bits in transmitted characters. (Only one stop bit is ever required in received characters.) If this bit is clear, characters are transmitted with one stop bit; if set, characters are transmitted with two stop bits.
- SC1_UARTPAR controls whether or not received and transmitted characters include a parity bit. If SC1_UARTPAR is clear, characters do not contain a parity bit, otherwise, characters do contain a parity bit.
- SC1_UARTODD specifies whether transmitted and received parity bits contain odd or even parity. If this bit is clear, the parity bit is even, and if set, the parity bit is odd. Even parity is the exclusive-or of all of the data bits, and odd parity is the inverse of the even parity value. SC1_UARTODD has no effect if SC1_UARTPAR is clear.

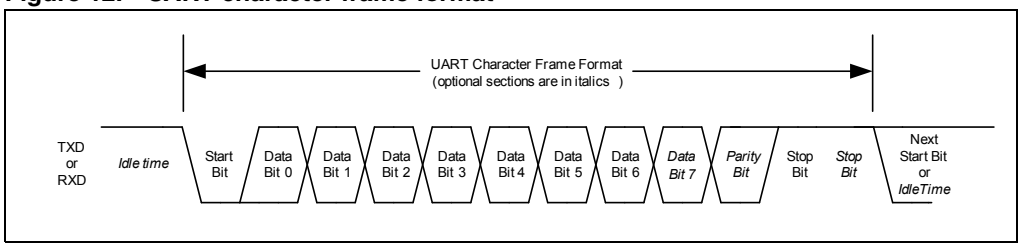
A UART character frame contains, in sequence:

- The start bit
- The least significant data bit
- The remaining data bits
- If parity is enabled, the parity bit
- The stop bit, or bits, if 2 stop bits are selected.

Figure 12 shows the UART character frame format, with optional bits indicated. Depending on the options chosen for the character frame, the length of a character frame ranges from 9 to 12 bit times.

Note that asynchronous serial data may have arbitrarily long idle periods between characters. When idle, serial data (TXD or RXD) is held in the high state. Serial data transitions to the low state in the start bit at the beginning of a character frame.

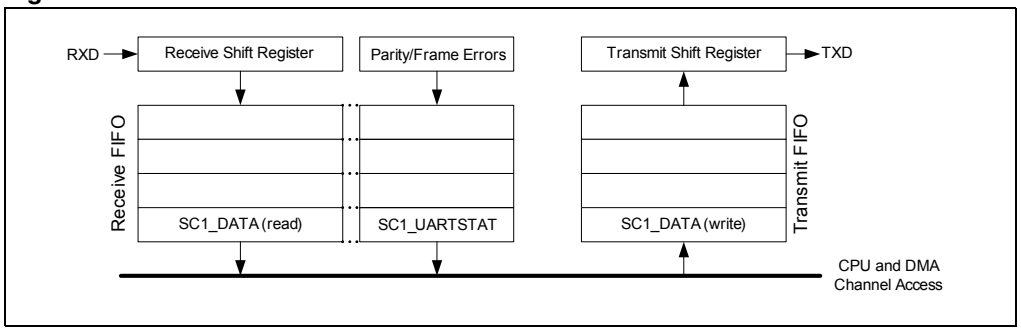
Figure 12. UART character frame format



9.6.2 FIFOs

Characters transmitted and received by the UART are buffered in the transmit and receive FIFOs that are both 4 entries deep (see *Figure 13*). When software writes a character to the SC1_DATA register, it is pushed onto the transmit FIFO. Similarly, when software reads from the SC1_DATA register, the character returned is pulled from the receive FIFO. If the transmit and receive DMA channels are used, the DMA channels also write to and read from the transmit and receive FIFOs.

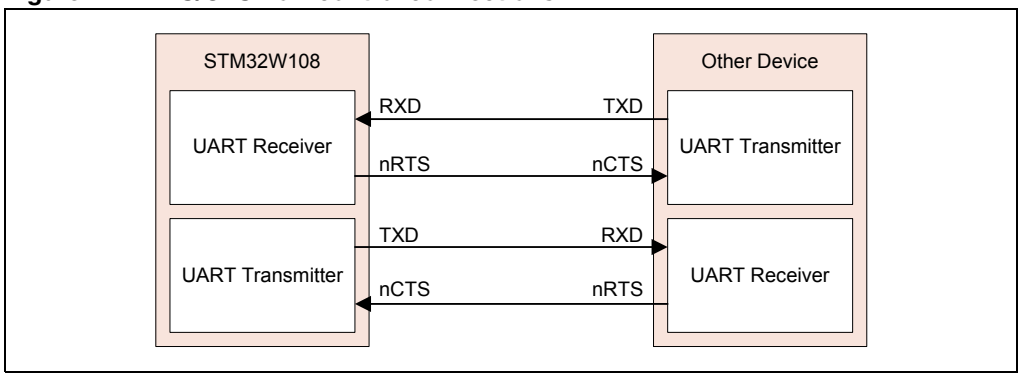
Figure 13. UART FIFOs



9.6.3 RTS/CTS flow control

RTS/CTS flow control, also called hardware flow control, uses two signals (nRTS and nCTS) in addition to received and transmitted data (see *Figure 14*). Flow control is used by a data receiver to prevent buffer overflow, by signaling an external device when it is and is not allowed to transmit.

Figure 14. RTS/CTS flow control connections



The UART RTS/CTS flow control options are selected by the SC1_UARTFLOW and SC1_UARTAUTO bits in the SC1_UARTCFG register (see *Table 51*). Whenever the SC1_UARTFLOW bit is set, the UART will not start transmitting a character unless nCTS is low (asserted). If nCTS transitions to the high state (deasserts) while a character is being transmitted, transmission of that character continues until it is complete.

If the SC1_UARTAUTO bit is set, nRTS is controlled automatically by hardware: nRTS is put into the low state (asserted) when the receive FIFO has room for at least two characters, otherwise is it in the high state (unasserted). If SC1_UARTAUTO is clear, software controls the nRTS output by setting or clearing the SC1_UARTRTS bit int the SC1_UARTCFG register. Software control of nRTS is useful if the external serial device cannot stop transmitting characters promptly when nRTS is set to the high state (deasserted).

Table 51. UART RTS/CTS flow control configurations

SC1_UARTCFG SC1_UARTxxx(1)			Pins used	Operating mode
FLOW	AUTO	RTS		
0	-	-	TXD, RXD	No RTS/CTS flow control
1	0	0/1	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with software control of nRTS: nRTS controlled by SC1_UARTRTS bit in SC1_UARTCFG register
1	1	-	TXD, RXD, nCTS, nRTS	Flow control using RTS/CTS with hardware control of nRTS: nRTS is asserted if room for at least 2 characters in receive FIFO

1. The notation xxx means that the corresponding column header below is inserted to form the field name.

9.6.4 DMA

The DMA Channels section describes how to configure and use the serial receive and transmit DMA channels.

The receive DMA channel has special provisions to record UART receive errors. When the DMA channel transfers a character from the receive FIFO to a buffer in memory, it checks the stored parity and frame error status flags. When an error is flagged, the SC1_RXERRA/B register is updated, marking the offset to the first received character with a parity or frame error. Similarly if a receive overrun error occurs, the SC1_RXERRA/B registers mark the error offset. The receive FIFO hardware generates the INT_SCRXOVF interrupt and DMA status register indicates the error immediately, but in this case the error

offset is 4 characters ahead of the actual overflow at the input to the receive FIFO. Two conditions will clear the error indication: setting the appropriate SC_RXDMARST bit in the SC1_DMACTRL register, or loading the appropriate DMA buffer after it has unloaded.

9.6.5 Interrupts

UART interrupts are generated on the following events:

- Transmit FIFO empty and last character shifted out (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC1_UARTTXIDLE)
- Transmit FIFO changed from full to not full (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC1_UARTTXFREE)
- Receive FIFO changed from empty to not empty (depending on SCx_INTMODE, either the 0 to 1 transition or the high level of SC1_UARTRXVAL)
- Transmit DMA buffer A/B complete (1 to 0 transition of SC_TXACTA/B)
- Receive DMA buffer A/B complete (1 to 0 transition of SC_RXACTA/B)
- Character received with parity error
- Character received with frame error
- Character received and lost when receive FIFO was full (receive overrun error)

To enable CPU interrupts, set the desired interrupt bits in the second level INT_SCxCFG register, and enable the top level SCx interrupt in the NVIC by writing the INT_SCx bit in the INT_CFGSET register.

9.7 Direct memory access (DMA) channels

The STM32W108 serial DMA channels enable efficient, high-speed operation of the SPI and UART controllers by reducing the load on the CPU as well as decreasing the frequency of interrupts that it must service. The transmit and receive DMA channels can transfer data between the transmit and receive FIFOs and the DMA buffers in main memory as quickly as it can be transmitted or received. Once software defines, configures, and activates the DMA, it only needs to handle an interrupt when a transmit buffer has been emptied or a receive buffer has been filled. The DMA channels each support two memory buffers, labeled A and B, and can alternate ("ping-pong") between them automatically to allow continuous communication without critical interrupt timing.

Note: DMA memory buffer terminology:

- *load - make a buffer available for the DMA channel to use*
- *pending - a buffer loaded but not yet active*
- *active - the buffer that will be used for the next DMA transfer*
- *unload - DMA channel action when it has finished with a buffer*
- *idle - a buffer that has not been loaded, or has been unloaded*

To use a DMA channel, software should follow these steps:

- Reset the DMA channel by setting the SC_TXDMARST (or SC_RXDMARST) bit in the SCx_DMACTRL register.
- Set up the DMA buffers. The two DMA buffers, A and B, are defined by writing the start address to SCx_TXBEGA/B (or SCx_RXBEGA/B) and the (inclusive) end address to SCx_TXENDA/B (or SCx_RXENDA/B). Note that DMA buffers must be in RAM.
- Configure and initialize SCx for the desired operating mode.
- Enable second level interrupts triggered when DMA buffers unload by setting the INT_SCTXULDA/B (or INT_SCRXULDA/B) bits in the INT_SCxFLAG register.

- Enable top level NVIC interrupts by setting the INT_SCx bit in the INT_CFGSET register.
- Start the DMA by loading the DMA buffers by setting the SC_TXLODA/B (or SC_RXLODA/B) bits in the SCx_DMACTRL register.

A DMA buffer's end address, SCx_TXENDA/B (or SCx_RXENDA/B), can be written while the buffer is loaded or active. This is useful for receiving messages that contain an initial byte count, since it allows software to set the buffer end address at the last byte of the message.

As the DMA channel transfers data between the transmit or receive FIFO and a memory buffer, the DMA count register contains the byte offset from the start of the buffer to the address of the next byte that will be written or read. A transmit DMA channel has a single DMA count register (SCx_TXCNT) that applies to whichever transmit buffer is active, but a receive DMA channel has two DMA count registers (SCx_RXCNTA/B), one for each receive buffer. The DMA count register contents are preserved until the corresponding buffer, or either buffer in the case of the transmit DMA count, is loaded, or until the DMA is reset.

The receive DMA count register may be written while the corresponding buffer is loaded. If the buffer is not loaded, writing the DMA count register also loads the buffer while preserving the count value written. This feature can simplify handling UART receive errors.

The DMA channel stops using a buffer and unloads it when the following is true:

(DMA buffer start address + DMA buffer count) > DMA buffer end address

Typically a transmit buffer is unloaded after all its data has been sent, and a receive buffer is unloaded after it is filled with data, but writing to the buffer end address or buffer count registers can also cause a buffer to unload early.

Serial controller DMA channels include additional features specific to the SPI and UART operation and are described in those sections.

9.8 Serial controller registers

9.8.1 Serial mode register (SCx_MODE)

Address offset: 0xC854 (SC1_MODE) and 0xC054 (SC2_MODE)
Reset value: 0x0000 0000

Table 52. Serial mode register (SCx_MODE)

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

Bits [1:0] SC_MODE: Serial controller mode.
0: Disabled.
1: UART mode (valid only for SC1).
2: SPI mode.
3: I^2C mode.

9.8.2 Serial controller interrupt flag register (INT_SCxFLAG)

Address offset: 0xA808 (INT_SC1FLAG) and 0xA80C (INT_SC2FLAG)
Reset value: 0x0000 0000

Table 53. Serial controller interrupt flag register (INT_SCxFLAG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	INT_SC1PARERR	INT_SC1FRMERR	INT_SCTXULDB	INT_SCTXULDA	INT_SCRXULDB	INT_SCRXULDA	INT_SCNAK	INT_SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_SCTXUND	INT_SCRXOVF	INT_SCTXIDLE	INT_SCTXFREE	INT_SCRXVAL
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 14 INT_SC1PARERR: Parity error received (UART) interrupt pending.

Bit 13 INT_SC1FRMERR: Frame error received (UART) interrupt pending.

Bit 12 INT_SCTXULDB: DMA transmit buffer B unloaded interrupt pending.

Bit 11 INT_SCTXULDA: DMA transmit buffer A unloaded interrupt pending.

Bit 10 INT_SCRXULDB: DMA receive buffer B unloaded interrupt pending.

Bit 9 INT_SCRXULDA: DMA receive buffer A unloaded interrupt pending.

Bit 8 INT_SCNAK: NACK received (I^2C) interrupt pending.

Bit 7 INT_SCCMDFIN: START/STOP command complete (I^2C) interrupt pending.

Bit 6 INT_SCTXFIN: Transmit operation complete (I^2C) interrupt pending.

Bit 5 INT_SCRXFIN: Receive operation complete (I^2C) interrupt pending.

Bit 4 INT_SCTXUND: Transmit buffer underrun interrupt pending.

Bit 3 INT_SCRXOVF: Receive buffer overrun interrupt pending.

Bit 2 INT_SCTXIDLE: Transmitter idle interrupt pending.

Bit 1 INT_SCTXFREE: Transmit buffer free interrupt pending.

Bit 0 INT_SCRXVAL: Receive buffer has data interrupt pending.

9.8.3 Serial controller interrupt configuration register (INT_SCxCFG)

Address offset: 0xA848 (INT_SC1CFG) and 0xA84C (INT_SC2CFG)
Reset value: 0x0000 0000

Table 54. Serial controller interrupt configuration register (INT_SCxCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	INT_SC1PARERR	INT_SC1FRMERR	INT_SCTXULDB	INT_SCTXULDA	INT_SCRXULDB	INT_SCRXULDA	INT_SCNAK	INT_SCCMDFIN	INT_SCTXFIN	INT_SCRXFIN	INT_SCTXUND	INT_SCRXOVF	INT_SCTXIDLE	INT_SCTXFREE	INT_SCRXVAL
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 14 INT_SC1PARERR: Parity error received (UART) interrupt enable.

Bit 13 INT_SC1FRMERR: Frame error received (UART) interrupt enable.

Bit 12 INT_SCTXULDB: DMA transmit buffer B unloaded interrupt enable.

Bit 11 INT_SCTXULDA: DMA transmit buffer A unloaded interrupt enable.

Bit 10 INT_SCRXULDB: DMA receive buffer B unloaded interrupt enable.

Bit 9 INT_SCRXULDA: DMA receive buffer A unloaded interrupt enable.

Bit 8 INT_SCNAK: NACK received (I^2C) interrupt enable.

Bit 7 INT_SCCMDFIN: START/STOP command complete (I^2C) interrupt enable.

Bit 6 INT_SCTXFIN: Transmit operation complete (I^2C) interrupt enable.

Bit 5 INT_SCRXFIN: Receive operation complete (I^2C) interrupt enable.

Bit 4 INT_SCTXUND: Transmit buffer underrun interrupt enable.

Bit 3 INT_SCRXOVF: Receive buffer overrun interrupt enable.

Bit 2 INT_SCTXIDLE: Transmitter idle interrupt enable.

Bit 1 INT_SCTXFREE: Transmit buffer free interrupt enable.

Bit 0 INT_SCRXVAL: Receive buffer has data interrupt enable.

9.8.4 Serial controller interrupt mode register (SCx_INTMODE)

Address offset: 0xA854 (SC1_INTMODE) and 0xA858 (SC2_INTMODE)
Reset value: 0x0000 0000

Table 55. Serial controller interrupt mode register (SCx_INTMODE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved													SC_TXIDLELEVEL	SC_TXFREELEVEL	SC_RXVALLEVEL
													rw	rw	rw

Bit 2 SC_TXIDLELEVEL: Transmitter idle interrupt mode
0: Edge triggered 1: Level triggered.

Bit 1 SC_TXFREELEVEL: Transmit buffer free interrupt mode
0: Edge triggered 1: Level triggered.

Bit 0 SC_RXVALLEVEL: Receive buffer has data interrupt mode
0: Edge triggered 1: Level triggered.

9.9 SPI master mode registers

9.9.1 Serial data register (SCx_DATA)

Address offset: 0xC83C (SC1_DATA) and 0xC03C (SC2_DATA)
Reset value: 0x0000 0000

Table 56. Serial data register (SCx_DATA)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								SC_DATA							
								rw							

Bits [7:0] SC_DATA: Transmit and receive data register. Writing to this register adds a byte to the transmit FIFO. Reading from this register takes the next byte from the receive FIFO and clears the overrun error bit if it was set.

In UART mode (SC1 only), reading from this register loads the UART status register with the parity and frame error status of the next byte in the FIFO, and clears these bits if the FIFO is now empty.

9.9.2 SPI configuration register (SCx_SPICFG)

Address offset: 0xC858 (SC1_SPICFG) and 0xC058 (SC2_SPICFG)
Reset value: 0x0000 0000

Table 57. SPI configuration register (SCx_SPICFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SC_SPIRXDRV	SC_SPIMST	SC_SPIRPT	SC_SPIORD	SC_SPIPHA	SC_SPIPOL
										rw	rw	rw	rw	rw	rw

Bit 5 SC_SPIRXDRV: Receiver-driven mode selection bit (SPI master mode only). Clear this bit to initiate transactions when transmit data is available. Set this bit to initiate transactions when the receive buffer (FIFO or DMA) has space.

Bit 4 SC_SPIMST: Set this bit to put the SPI in master mode, clear this bit to put the SPI in slave mode.

Bit 3 SC_SPIRPT: This bit controls behavior on a transmit buffer underrun condition in slave mode. Clear this bit to send the BUSY token (0xFF) and set this bit to repeat the last byte. Changes to this bit take effect when the transmit FIFO is empty and the transmit serializer is idle.

Bit 2 SC_SPIORD: This bit specifies the bit order in which SPI data is transmitted and received.
0: Most significant bit first. 1: Least significant bit first.

Bit 1 SC_SPIPHA: Clock phase configuration: clear this bit to sample on the leading (first edge) and set this bit to sample on the second edge.

Bit 0 SC_SPIPOL: Clock polarity configuration: clear this bit for a rising leading edge and set this bit for a falling leading edge.

9.9.3 SPI status register (SCx_SPISTAT)

Address offset: 0xC840 (SC1_SPISTAT) and 0xC040 (SC2_SPISTAT)
Reset value: 0x0000 0000

Table 58. SPI status register (SCx_SPISTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SC_SPI TXIDLE	SC_SPI TXFREE	SC_SPI RXVAL	SC_SPI RXOVF
												r	r	r	r

Bit 3 SC_SPITXIDLE: This bit is set when both the transmit FIFO and the transmit serializer are empty.

Bit 2 SC_SPITXFREE: This bit is set when the transmit FIFO has space to accept at least one byte.

Bit 1 SC_SPIRXVAL: This bit is set when the receive FIFO contains at least one byte.

Bit 0 SC_SPIRXOVF: This bit is set if a byte is received when the receive FIFO is full. This bit is cleared by reading the data register.

9.9.4 Serial clock linear prescaler register (SCx_RATELIN)

Address offset: 0xC860 (SC1_RATELIN) and 0xC060 (SC2_RATELIN)
Reset value: 0x0000 0000

Table 59. Serial clock linear prescaler register (SCx_RATELIN)

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

Bits [3:0] SC_RATELIN: The linear component (LIN) of the clock rate in the equation:
Rate = 12 MHz / ((LIN + 1) * (2^EXP))

9.9.5 Serial clock exponential prescaler register (SCx_RATEEXP)

Address offset: 0xC864 (SC1_RATEEXP) and 0xC064 (SC2_RATEEXP)
Reset value: 0x0000 0000

Table 60. Serial clock exponential prescaler register (SCx_RATEEXP)

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

Bits [3:0] SC_RATEEXP: The exponential component (EXP) of the clock rate in the equation:
Rate = 12 MHz / ((LIN + 1) * (2^EXP))

9.10 SPI slave mode registers

Refer to *Section 9.9: SPI master mode registers on page 95* for a description of the SCx_DATA, SCx_SPICFG, and SCx_SPISTAT registers.

9.11 Inter-integrated circuit (I^2C) interface registers

9.11.1 I^2C status register (SCx_TWISTAT)

Address offset: 0xC844 (SC1_TWISTAT) and 0xC044 (SC2_TWISTAT)
Reset value: 0x0000 0000

Table 61. I^2C status register (SCx_TWISTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SC_TWICMDFIN	SC_TWIRXFIN	SC_TWITXFIN	SC_TWIRXNAK
												r	r	r	r

Bit 3 SC_TWICMDFIN: This bit is set when a START or STOP command completes. It clears on the next I2C bus activity.

Bit 2 SC_TWIRXFIN: This bit is set when a byte is received. It clears on the next I^2C bus activity.

Bit 1 SC_TWITXFIN: This bit is set when a byte is transmitted. It clears on the next I^2C bus activity.

Bit 0 SC_TWIRXNAK: This bit is set when a NACK is received from the slave. It clears on the next I^2C bus activity.

9.11.2 I²C control 1 register (SCx_TWICTRL1)

Address offset: 0xC84C (SC1_TWICTRL1) and 0xC04C (SC2_TWICTRL1)
Reset value: 0x0000 0000

Table 62. I²C control 1 register (SCx_TWICTRL1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												SC_TWISTOP	SC_TWISTART	SC_TWISEND	SC_TWIRECV
												rw	rw	rw	rw

Bit 3 SC_TWISTOP: Setting this bit sends the STOP command. It clears when the command completes.

Bit 2 SC_TWISTART: Setting this bit sends the START or repeated START command. It clears when the command completes.

Bit 1 SC_TWISEND: Setting this bit transmits a byte. It clears when the command completes.

Bit 0 SC_TWIRECV: Setting this bit receives a byte. It clears when the command completes.

9.11.3 I²C control 2 register (SCx_TWICTRL2)

Address offset: 0xC850 (SC1_TWICTRL2) and 0xC050 (SC2_TWICTRL2)
Reset value: 0x0000 0000

Table 63. I²C control 2 register (SCx_TWICTRL2)

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

Bit 0 SC_TWIACK: Setting this bit signals ACK after a received byte. Clearing this bit signals NACK after a received byte.

9.12 Universal asynchronous receiver / transmitter (UART) registers

Refer to the SPI Master mode section for a description of the SCx_DATA register.

9.12.1 UART status register (SC1_UARTSTAT)

Address offset: 0xC848
Reset value: 0x0000 0040

Table 64. UART status register (SC1_UARTSTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SC_UARTTXIDLE	SC_UARTPARERR	SC_UARTFRMERR	SC_UARTRXOVF	SC_UARTTXFREE	SC_UARTRXVAL	SC_UARTCTS
									r	r	r	r	r	r	r

Bit 6 SC_UARTTXIDLE: This bit is set when both the transmit FIFO and the transmit serializer are empty.

Bit 5 SC_UARTPARERR: This bit is set when the byte in the data register was received with a parity error. This bit is updated when the data register is read, and is cleared if the receive FIFO is empty.

Bit 4 SC_UARTFRMERR: This bit is set when the byte in the data register was received with a frame error. This bit is updated when the data register is read, and is cleared if the receive FIFO is empty.

Bit 3 SC_UARTRXOVF: This bit is set when the receive FIFO has been overrun. This occurs if a byte is received when the receive FIFO is full. This bit is cleared by reading the data register.

Bit 2 SC_UARTTXFREE: This bit is set when the transmit FIFO has space for at least one byte.

Bit 1 SC_UARTRXVAL: This bit is set when the receive FIFO contains at least one byte.

Bit 0 SC_UARTCTS: This bit is set when both the transmit FIFO and the transmit serializer are empty.

9.12.2 UART configuration register (SC1_UARTCFG)

Address offset: 0xC85C
Reset value: 0x0000 0000

Table 65. UART configuration register (SC1_UARTCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									SC_UARTAUTO	SC_UARTFLOW	SC_UARTODD	SC_UARTPAR	SC_UART2STP	SC_UART8BIT	SC_UARTRTS
									rw	rw	rw	rw	rw	rw	rw

Bit 6 SC_UARTAUTO: Set this bit to enable automatic nRTS control by hardware (SC_UARTFLOW must also be set). When automatic control is enabled, nRTS will be deasserted when the receive FIFO has space for only one more byte (inhibits transmission from the other device) and will be asserted if it has space for more than one byte (enables transmission from the other device). The SC_UARTRTS bit in this register has no effect if this bit is set.

Bit 5 SC_UARTFLOW: Set this bit to enable using nRTS/nCTS flow control signals. Clear this bit to disable the signals. When this bit is clear, the UART transmitter will not be inhibited by nCTS.

Bit 4 SC_UARTODD: If parity is enabled, specifies the kind of parity.
0: Even parity. 1: Odd parity.

Bit 3 SC_UARTPAR: Specifies whether to use parity bits.
0: Don't use parity. 1: Use parity.

Bit 2 SC_UART2STP: Number of stop bits transmitted.
0: 1 stop bit. 1: 2 stop bits.

Bit 1 SC_UART8BIT: Number of data bits.
0: 7 data bits. 1: 8 data bits.

Bit 0 SC_UARTRTS: nRTS is an output to control the flow of serial data sent to the STM32W108 from another device. This bit directly controls the output at the nRTS pin (SC_UARTFLOW must be set and SC_UARTAUTO must be cleared). When this bit is set, nRTS is asserted (pin is low, 'XON', RS232 positive voltage); the other device's transmission is enabled. When this bit is cleared, nRTS is deasserted (pin is high, 'XOFF', RS232 negative voltage), the other device's transmission is inhibited.

9.12.3 UART baud rate period register (SC1_UARTPER)

Address offset: 0xC868
Reset value: 0x0000 0000

Table 66. UART baud rate period register (SC1_UARTPER)

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

Bits [15:0] SC_UARTPER: The integer part of baud rate period (N) in the equation:
Rate = 24 MHz / ((2 * N) + F)

9.12.4 UART baud rate fractional period register (SC1_UARTFRAC)

Address offset: 0xC86C
Reset value: 0x0000 0000

Table 67. UART baud rate fractional period register (SC1_UARTFRAC)

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

Bits [0] SC_UARTFRAC: The fractional part of the baud rate period (F) in the equation:
Rate = 24 MHz / ((2 * N) + F)

9.13 DMA channel registers

9.13.1 Serial DMA control register (SCx_DMACTRL)

Address offset: 0xC830 (SC1_DMACTRL) and 0xC030 (SC2_DMACTRL)
Reset value: 0x0000 0000

Table 68. Serial DMA control register (SCx_DMACTRL)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										SC_TXDMARST	SC_RXDMARST	SC_TXLODB	SC_TXLODA	SC_RXLODB	SC_RXLODA
										w	w	rw	rw	rw	rw

Bit 5 SC_TXDMARST: Setting this bit resets the transmit DMA. The bit clears automatically.

Bit 4 SC_RXDMARST: Setting this bit resets the receive DMA. The bit clears automatically.

Bit 3 SC_TXLODB: Setting this bit loads DMA transmit buffer B addresses and allows the DMA controller to start processing transmit buffer B. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect.

Reading this bit returns DMA buffer status:

0: DMA processing is complete or idle.

1: DMA processing is active or pending.

Bit 2 SC_TXLODA: Setting this bit loads DMA transmit buffer A addresses and allows the DMA controller to start processing transmit buffer A. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect.

Reading this bit returns DMA buffer status:

0: DMA processing is complete or idle.

1: DMA processing is active or pending.

Bit 1 SC_RXLODB: Setting this bit loads DMA receive buffer B addresses and allows the DMA controller to start processing receive buffer B. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect.

Reading this bit returns DMA buffer status:

0: DMA processing is complete or idle.

1: DMA processing is active or pending.

Bit 0 SC_RXLODA: Setting this bit loads DMA receive buffer A addresses and allows the DMA controller to start processing receive buffer A. If both buffer A and B are loaded simultaneously, buffer A will be used first. This bit is cleared when DMA completes. Writing a zero to this bit has no effect.

Reading this bit returns DMA buffer status:

0: DMA processing is complete or idle.

1: DMA processing is active or pending.

9.13.2 Serial DMA status register (SCx_DMASTAT)

Address offset: 0xC82C (SC1_DMASTAT) and 0xC02C (SC2_DMASTAT)
Reset value: 0x0000 0000

Table 69. Serial DMA status register (SCx_DMASTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			SC_RXSSEL			SC_RX FRMB	SC_RX FRMA	SC_RX PARB	SC_RX PARA	SC_RX OVFB	SC_R XOVF A	SC_TX ACTB	SC_TX ACTA	SC_RX ACTB	SC_RX ACTA
			r			r	r	r	r	r	r	r	r	r	r

Bits [12:10] SC_RXSSEL: Status of the receive count saved in SCx_RXCNTSAVED (SPI slave mode) when nSSEL deasserts. Cleared when a receive buffer is loaded and when the receive DMA is reset.

0: No count was saved because nSSEL did not deassert.
2: Buffer A's count was saved, nSSEL deasserted once.
3: Buffer B's count was saved, nSSEL deasserted once.
6: Buffer A's count was saved, nSSEL deasserted more than once.
7: Buffer B's count was saved, nSSEL deasserted more than once.
1, 4, 5: Reserved.

Bit 9 SC_RXFRMB: This bit is set when DMA receive buffer B reads a byte with a frame error from the receive FIFO. It is cleared the next time buffer B is loaded or when the receive DMA is reset. (SC1 in UART mode only)

Bit 8 SC_RXFRMA: This bit is set when DMA receive buffer A reads a byte with a frame error from the receive FIFO. It is cleared the next time buffer A is loaded or when the receive DMA is reset. (SC1 in UART mode only)

Bit 7 This bit is set when DMA receive buffer B reads a byte with a parity error from the receive FIFO. It is cleared the next time buffer B is loaded or when the receive DMA is reset. (SC1 in UART mode only)

Bit 6 This bit is set when DMA receive buffer A reads a byte with a parity error from the receive FIFO. It is cleared the next time buffer A is loaded or when the receive DMA is reset. (SC1 in UART mode only)

Bit 5 This bit is set when DMA receive buffer B was passed an overrun error from the receive FIFO. Neither receive buffer was capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer B was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. Cleared the next time buffer B is loaded and when the receive DMA is reset.

Bit 4 This bit is set when DMA receive buffer A was passed an overrun error from the receive FIFO. Neither receive buffer was capable of accepting any more bytes (unloaded), and the FIFO filled up. Buffer A was the next buffer to load, and when it drained the FIFO the overrun error was passed up to the DMA and flagged with this bit. Cleared the next time buffer A is loaded and when the receive DMA is reset.

Bit 3 This bit is set when DMA transmit buffer B is active.

Bit 2 This bit is set when DMA transmit buffer A is active.

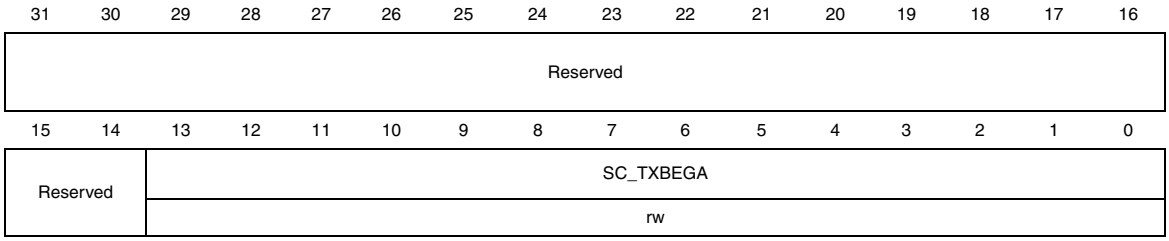
Bit 1 This bit is set when DMA receive buffer B is active.

Bit 0 This bit is set when DMA receive buffer A is active.

9.13.3 Transmit DMA begin address register A (SCx_TXBEGA)

Address offset: 0xC810 (SC1_TXBEGA) and 0xC010 (SC2_TXBEGA)
Reset value: 0x2000 0000

Table 70. Transmit DMA begin address register A (SCx_TXBEGA)

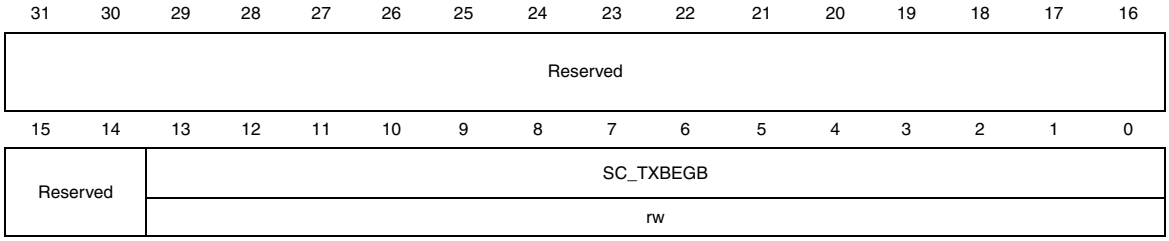


Bits [13:0] SC_TXBEGA: DMA transmit buffer A start address.

9.13.4 Transmit DMA begin address register B (SCx_TXBEGB)

Address offset: 0xC818 (SC1_TXBEGB) and 0xC018 (SC2_TXBEGB)
Reset value: 0x2000 0000

Table 71. Transmit DMA begin address register B (SCx_TXBEGB)



Bits [13:0] SC_TXBEGA: DMA transmit buffer B start address.

9.13.5 Transmit DMA end address register A (SCx_TXENDA)

Address offset: 0xC814 (SC1_TXENDA) and 0xC014 (SC2_TXENDA)
Reset value: 0x2000 0000

Table 72. Transmit DMA end address register A (SCx_TXENDA)

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

Bits [13:0] SC_TXENDA: Address of the last byte that will be read from the DMA transmit buffer A.

9.13.6 Transmit DMA end address register B (SCx_TXENDB)

Address offset: 0xC81C (SC1_TXENDB) and 0xC01C (SC2_TXENDB)
Reset value: 0x2000 0000

Table 73. Transmit DMA end address register B (SCx_TXENDB)

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

Bits [13:0] SC_TXENDB: Address of the last byte that will be read from the DMA transmit buffer B.

9.13.7 Transmit DMA count register (SCx_TXCNT)

Address offset: 0xC828 (SC1_TXCNT) and 0xC028 (SC2_TXCNT)
Reset value: 0x0000 0000

Table 74. Transmit DMA count register (SCx_TXCNT)

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

Bits [13:0] SC_TXCNT: The offset from the start of the active DMA transmit buffer from which the next byte will be read. This register is set to zero when the buffer is loaded and when the DMA is reset.

9.13.8 Receive DMA begin address register A (SCx_RXBEGA)

Address offset: 0xC800 (SC1_RXBEGA) and 0xC000 (SC2_RXBEGA)
Reset value: 0x2000 0000

Table 75. Receive DMA begin address register A (SCx_RXBEGA)

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

Bits [13:0] SC_RXBEGA: DMA receive buffer A start address.

9.13.9 Receive DMA begin address register B (SCx_RXBEGB)

Address offset: 0xC808 (SC1_RXBEGB) and 0xC008 (SC2_RXBEGB)
Reset value: 0x2000 0000

Table 76. Receive DMA begin address register B (SCx_RXBEGB)

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

Bits [13:0] SC_RXBEGB: DMA receive buffer B start address.

9.13.10 Receive DMA end address register A (SCx_RXENDA)

Address offset: 0xC804 (SC1_RXENDA) and 0xC004 (SC2_RXENDA)
Reset value: 0x0000 0000

Table 77. Receive DMA end address register A (SCx_RXENDA)

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

Bits [13:0] SC_RXENDA: Address of the last byte that will be written in the DMA receive buffer A.

9.13.11 Receive DMA end address register B (SCx_RXENDB)

Address offset: 0xC80C (SC1_RXENDB) and 0xC00C (SC2_RXENDB)
Reset value: 0x2000 0000

Table 78. Receive DMA end address register B (SCx_RXENDB)

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

Bits [13:0] SC_RXENDB: Address of the last byte that will be written in the DMA receive buffer B.

9.13.12 Receive DMA count register A (SCx_RXCNTA)

Address offset: 0xC820 (SC1_RXCNTA) and 0xC020 (SC2_RXCNTA)
Reset value: 0x0000 0000

Table 79. Receive DMA count register A (SCx_RXCNTA)

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

Bits [13:0] SC_RXCNTA: The offset from the start of DMA receive buffer A at which the next byte will be written. This register is set to zero when the buffer is loaded and when the DMA is reset. If this register is written when the buffer is not loaded, the buffer is loaded.

9.13.13 Receive DMA count register B (SCx_RXCNTB)

Address offset: 0xC824 (SC1_RXCNTB) and 0xC024 (SC2_RXCNTB)
Reset value: 0x0000 0000

Table 80. Receive DMA count register B (SCx_RXCNTB)

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

Bits [13:0] SC_RXCNTB: The offset from the start of DMA receive buffer B at which the next byte will be written. This register is set to zero when the buffer is loaded and when the DMA is reset. If this register is written when the buffer is not loaded, the buffer is loaded.

9.13.14 Saved receive DMA count register (SCx_RXCNTSAVED)

Address offset: 0xC870 (SC1_RXCNTSAVED) and 0xC070 (SC2_RXCNTSAVED)
Reset value: 0x0000 0000

Table 81. Saved receive DMA count register (SCx_RXCNTSAVED)

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

Bits [13:0] SC_RXCNTSAVED: Receive DMA count saved in SPI slave mode when nSSEL deasserts. The count is only saved the first time nSSEL deasserts.

9.13.15 DMA first receive error register A (SCx_RXERRA)

Address offset: 0xC834 (SC1_RXERRA) and 0xC034 (SC2_RXERRA)
Reset value: 0x0000 0000

Table 82. DMA first receive error register A (SCx_RXERRA)

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

Bits [13:0] SC_RXERRA: The offset from the start of DMA receive buffer A of the first byte received with a parity, frame, or overflow error. Note that an overflow error occurs at the input to the receive FIFO, so this offset is 4 bytes before the overflow position. If there is no error, it reads zero. This register will not be updated by subsequent errors until the buffer unloads and is reloaded, or the receive DMA is reset.

9.13.16 DMA first receive error register B (SCx_RXERRB)

Address offset: 0xC838 (SC1_RXERRB) and 0xC038 (SC2_RXERRB)
Reset value: 0x0000 0000

Table 83. DMA first receive error register B (SCx_RXERRB)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		SC_RXERRB													
		r													

Bits [13:0] SC_RXERRB: The offset from the start of DMA receive buffer B of the first byte received with a parity, frame, or overflow error. Note that an overflow error occurs at the input to the receive FIFO, so this offset is 4 bytes before the overflow position. If there is no error, it reads zero. This register will not be updated by subsequent errors until the buffer unloads and is reloaded, or the receive DMA is reset.

10 General-purpose timers

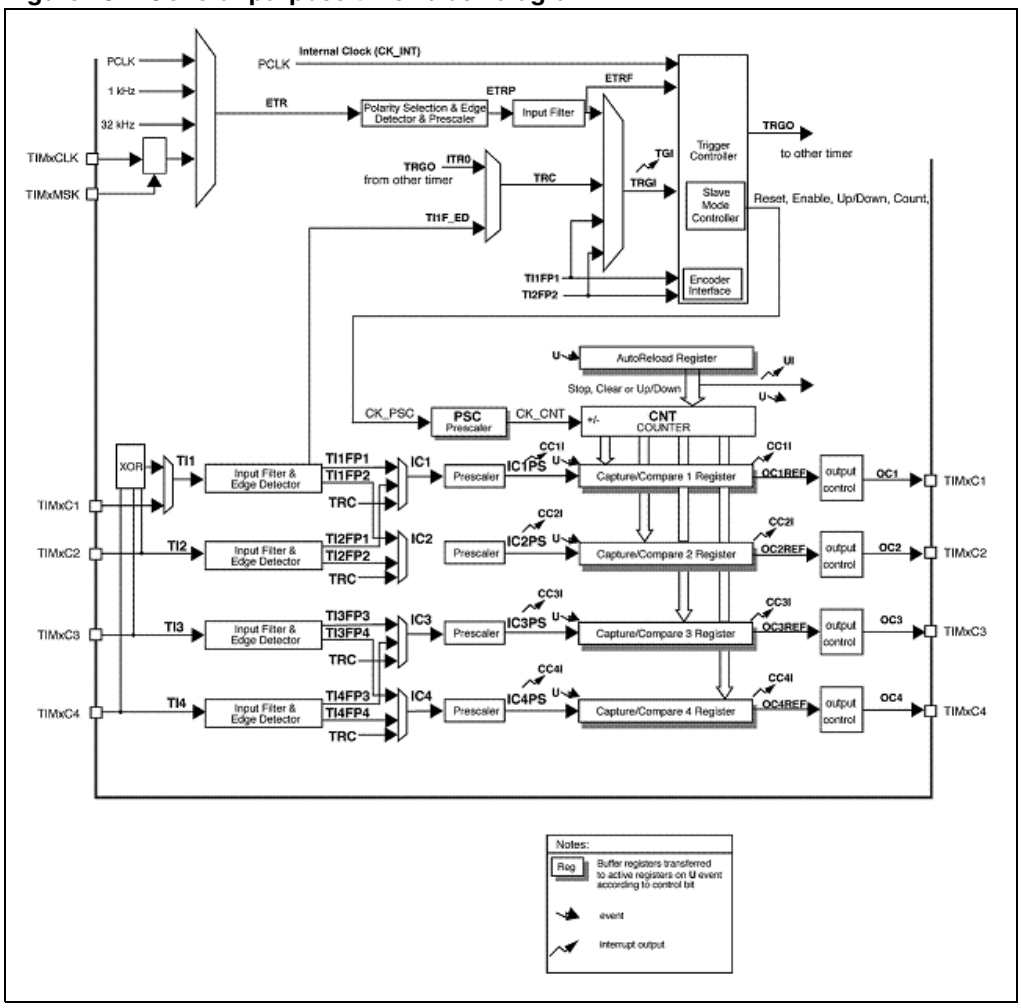
Each of the STM32W108's two general-purpose timers consists of a 16-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM). Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler. The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 10.1.14: Timer synchronization on page 139*.

The two general-purpose timers, TIM1 and TIM2, have the following features:

- 16-bit up, down, or up/down auto-reload counter.
- Programmable prescaler to divide the counter clock by any power of two from 1 through 32768.
- 4 independent channels for:
  - Input capture
  - Output compare
- PWM generation (edge- and center-aligned mode)
- One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect the timers.
- Flexible clock source selection:
  - Peripheral clock (PCLK at 6 or 12 MHz)
  - 32 kHz external clock (if available)
  - 1 kHz clock
- GPIO input
- Interrupt generation on the following events:
  - Update: counter overflow/underflow, counter initialization (software or internal/external trigger)
  - Trigger event (counter start, stop, initialization or count by internal/external trigger)
- Input capture
- Output compare
- Supports incremental (quadrature) encoders and Hall sensors for positioning applications.
- Trigger input for external clock or cycle-by-cycle current management.

Note: Because the two timers are identical, the notation TIMx refers to either TIM1 or TIM2. For example, TIMx_PSC refers to both TIM1_PSC and TIM2_PSC. Similarly, "y" refers to any of the four channels of a given timer, so for example, OCy refers to OC1, OC2, OC3, and OC4.

Figure 15. General-purpose timer block diagram



Note: The internal signals shown in Figure 15 are described in Section 10.1.15: Timer signal descriptions on page 143 and are used throughout the text to describe how the timer components are interconnected.

10.1 Functional description

The timers can optionally use GPIOs in the PA and PB ports for external inputs or outputs. As with all STM32W108 digital inputs, a GPIO used as a timer input can be shared with other uses of the same pin. Available timer inputs include an external timer clock, a clock mask, and four input channels. Any GPIO used as a timer output must be configured as an alternate output and is controlled only by the timer.

Many of the GPIOs that can be assigned as timer outputs can also be used by another on-chip peripheral such as a serial controller. Use as a timer output takes precedence over another peripheral function, as long as the channel is configured as an output in the TIMx_CCMR1 register and is enabled in the TIMx_CCER register.

The GPIOs that can be used by Timer 1 are fixed, but the GPIOs that can be used as Timer 2 channels can be mapped to either of two pins, as shown in *Table 84*. The Timer 2 Option Register (TIM2_OR) has four single bit fields (TIM_REMAPCy) that control whether a Timer 2 channel is mapped to its default GPIO in port PA, or remapped to a GPIO in PB.

Table 84 specifies the pins that may be assigned to Timer 1 and Timer 2 functions.

Table 84. Timer GPIO use

Signal (direction)	TIMxC1 (in or out)	TIMxC2 (in or out)	TIMxC3 (in or out)	TIMxC4 (in or out)	TIMxCLK (in)	TIMxMSK (in)
Timer 1	PB6	PB7	PA6	PA7	PB0	PB5
Timer 2 (TIM_REMAPCy = 0)	PA0	PA3	PA1	PA2	PB5	PB0
Timer 2 (TIM_REMAPCy = 1)	PB1	PB2	PB3	PB4	PB5	PB0

The TIMxCLK and TIMxMSK inputs can be used only in the external clock modes: refer to the External Clock Source Mode 1 and External Clock Source Mode 2 sections for details concerning their use.

10.1.1 Time-base unit

The main block of the general purpose timer is a 16-bit counter with its related auto-reload register. The counter can count up, down, or alternate up and down. The counter clock can be divided by a prescaler.

The counter, the auto-reload register, and the prescaler register can be written to or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-reload register (TIMx_ARR)

Some timer registers cannot be directly accessed by software, which instead reads and writes a "buffer register". The internal registers actually used for timer operations are called "shadow registers".

The auto-reload register is buffered. Writing to or reading from the auto-reload register accesses the buffer register. The contents of the buffer register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload buffer enable bit (TIM_ARBE) in the TIMx_CR1 register. The update event is generated when both the counter reaches the overflow (or underflow when down-counting) and when the TIM_UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. Update event generation is described in detail for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (TIM_CEN) in the TIMx_CR1 register is set. Refer also to the slave mode controller description in the Timers and External Trigger Synchronization section to get more details on counter enabling.

Note that the actual counter enable signal CNT_EN is set one clock cycle after TIM_CEN.

Note: When the STM32W108 enters debug mode and the ARM® Cortex-M3 core is halted, the counters continue to run normally.

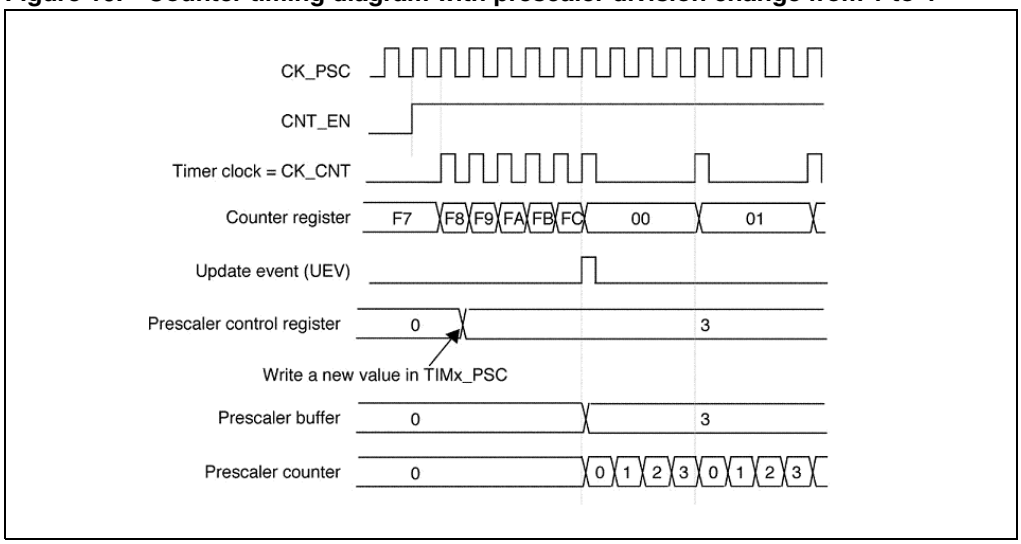
Prescaler

The prescaler can divide the counter clock frequency by power of two from 1 through 32768. It is based on a 16-bit counter controlled through the 4-bit TIM_PSCEXP bit field in the TIMx_PSC register. The factor by which the counter is divided is two raised to the power TIM_PSCEXP (2TIM_PSCEXP).

It can be changed on the fly as this control register is buffered. The new prescaler ratio is used starting at the next update event.

Figure 16 gives an example of the counter behavior when the prescaler ratio is changed on the fly.

Figure 16. Counter timing diagram with prescaler division change from 1 to 4



10.1.2 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow, by setting the TIM_UG bit in the TIMx_EGR register, or by using the slave mode controller.

Software can disable the update event by setting the TIM_UDIS bit in the TIMx_CR1 register, to avoid updating the shadow registers while writing new values in the buffer registers. No update event will occur until the TIM_UDIS bit is written to 0. Both the counter and the prescalar counter restart from 0, but the prescale rate does not change. In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates an update event but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_USR is 1) and the following registers are updated:

- The buffer of the prescaler is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload shadow register is updated with the buffer value (TIMx_ARR).

Figure 17, *Figure 18*, *Figure 19*, and *Figure 20* show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 17. Counter timing diagram, internal clock divided by 1

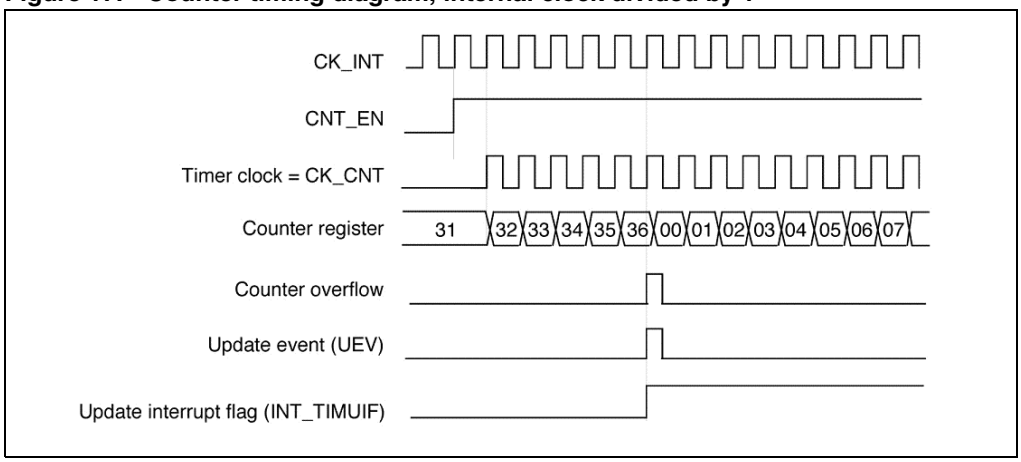


Figure 18. Counter timing diagram, internal clock divided by 4

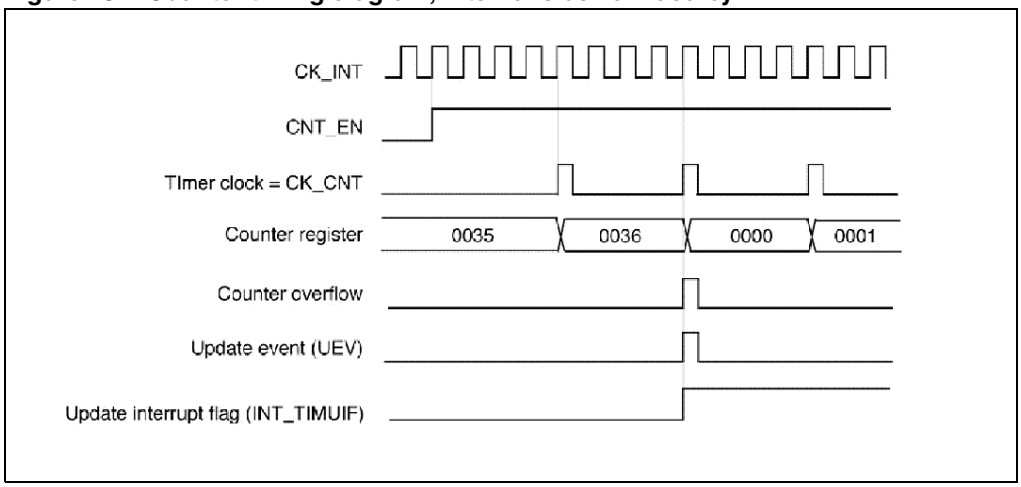


Figure 19. Counter timing diagram, update event when TIM_ARBE = 0 (TIMx_ARR not buffered)

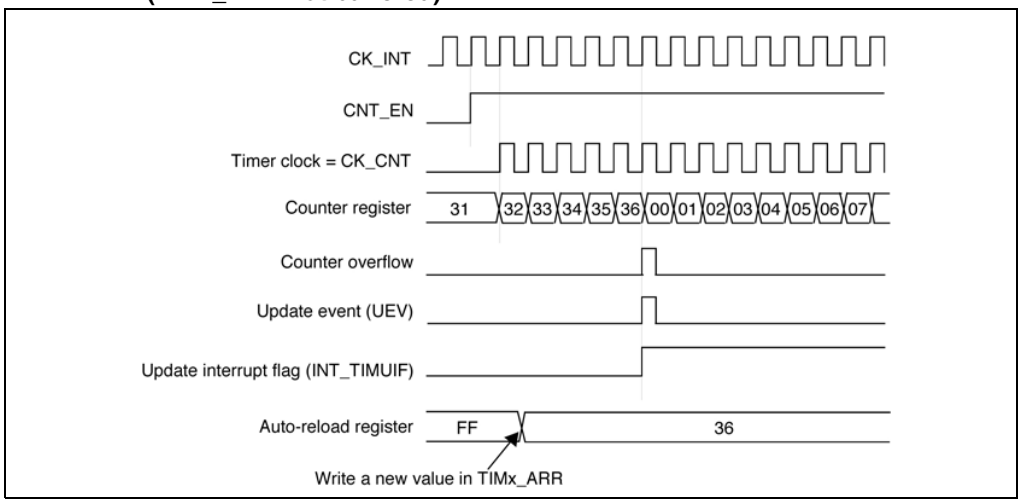
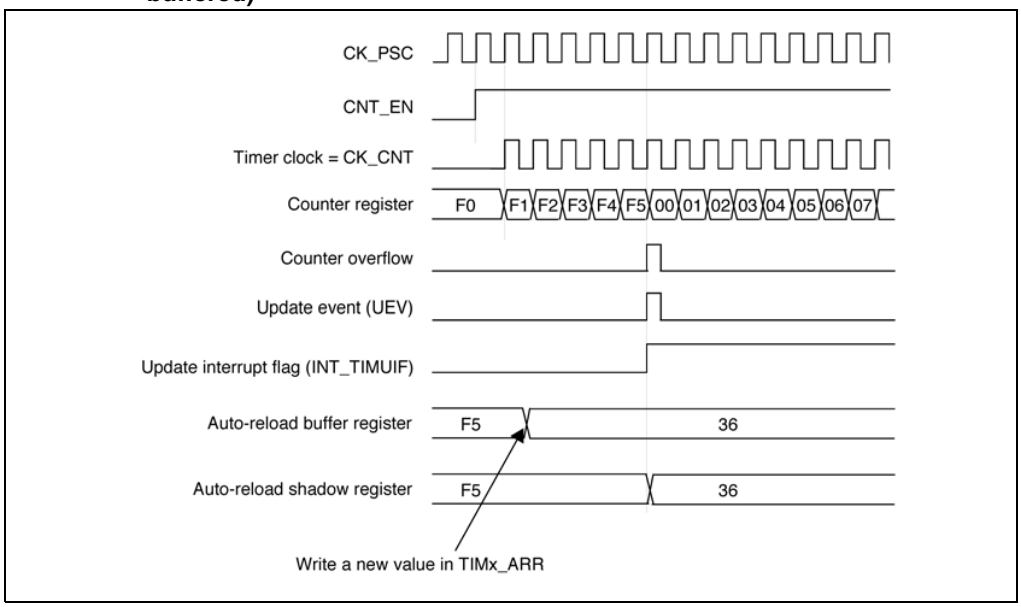


Figure 20. Counter timing diagram, update event when TIM_ARBE = 1 (TIMx_ARR buffered)



Down-counting mode

In down-counting mode, the counter counts from the auto-reload value (contents of the TIMx_ARR register) down to 0, then restarts from the auto-reload value and generates a counter underflow event.

An update event can be generated at each counter underflow, by setting the TIM_UG bit in the TIMx_EGR register, or by using the slave mode controller). Software can disable the update event by setting the TIM_UDIS bit in the TIMx_CR1 register, to avoid updating the shadow registers while writing new values in the buffer registers. No update event occurs until the TIM_UDIS bit is written to 0. However, the counter restarts from the current auto-

reload value, whereas the prescalar's counter restarts from 0, but the prescale rate doesn't change.

In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates an update event, but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_USR is 1) and the following registers are updated:

- The prescaler shadow register is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload active register is updated with the buffer value (contents of the TIMx_ARR register). The auto-reload is updated before the counter is reloaded, so that the next period is the expected one.

Figure 21 and *Figure 22* show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 21. Counter timing diagram, internal clock divided by 1

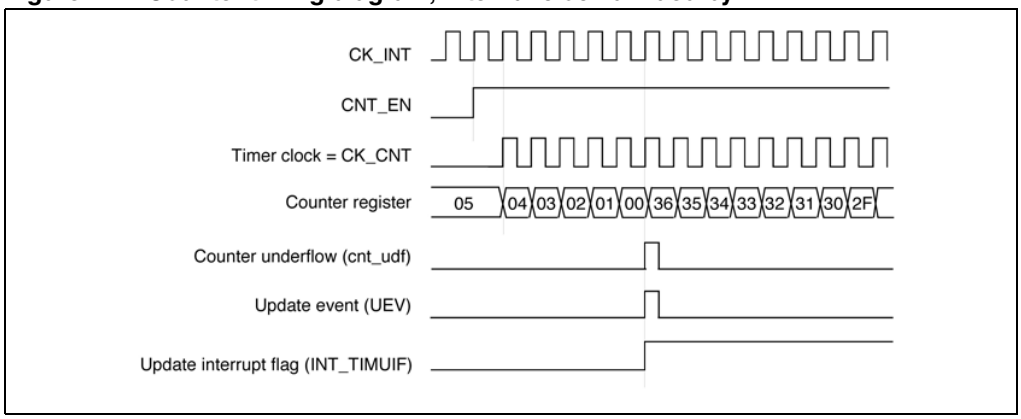
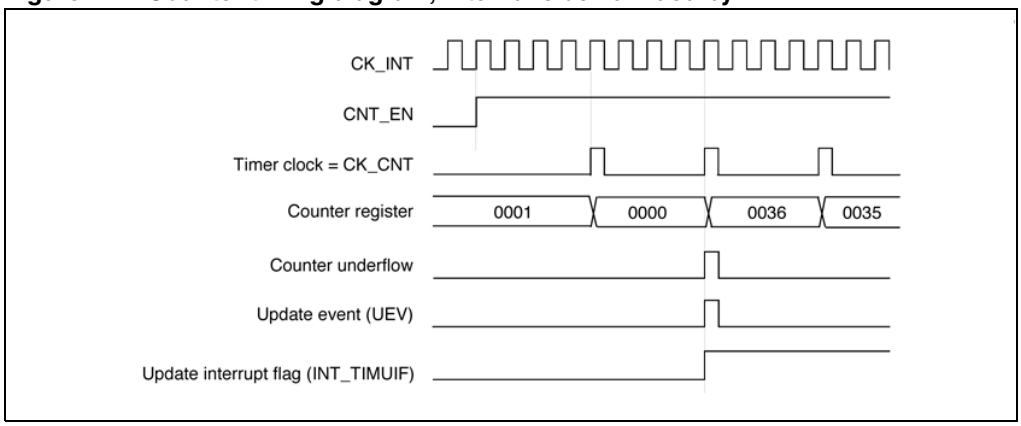


Figure 22. Counter timing diagram, internal clock divided by 4



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (contents of the TIMx_ARR register) - 1 and generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

In this mode, the direction bit (TIM_DIR in the TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow. Setting the TIM_UG bit in the TIMx_EGR register by software or by using the slave mode controller also generates an update event. In this case, the both the counter and the prescalar's counter restart counting from 0.

Software can disable the update event by setting the TIM_UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values in the buffer registers. Then no update event occurs until the TIM_UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current auto-reload value.

In addition, if the TIM_URS bit in the TIMx_CR1 register is set, setting the TIM_UG bit generates an update event, but without setting the INT_TIMUIF flag. Thus no interrupt request is sent. This avoids generating both update and capture interrupt when clearing the counter on the capture event.

When an update event occurs, the update flag (the INT_TIMUIF bit in the INT_TIMxFLAG register) is set (unless TIM_USR is 1) and the following registers are updated:

- The prescaler shadow register is reloaded with the buffer value (contents of the TIMx_PSC register).
- The auto-reload active register is updated with the buffer value (contents of the TIMx_ARR register). If the update source is a counter overflow, the auto-reload is updated before the counter is reloaded, so that the next period is the expected one. The counter is loaded with the new value.

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 23. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

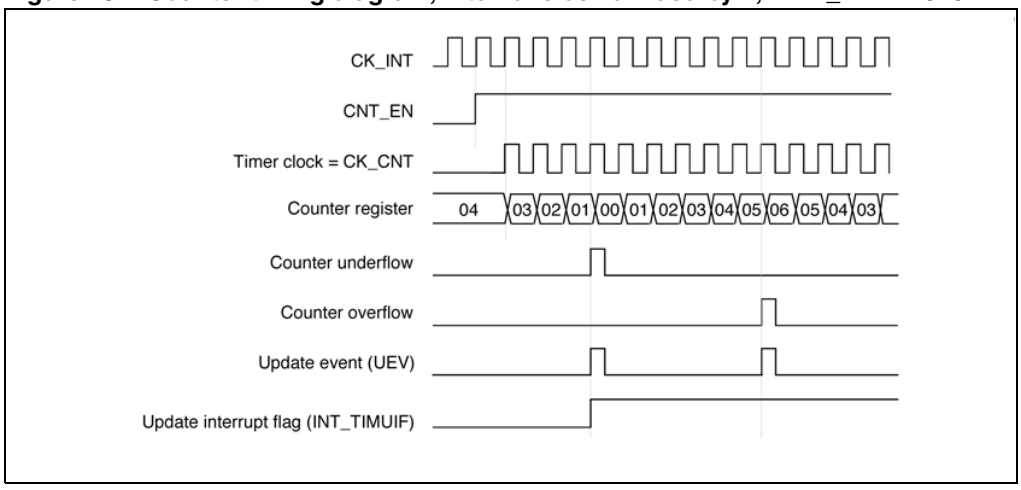


Figure 24. Counter timing diagram, update event with TIM_ARBE = 1 (counter underflow)

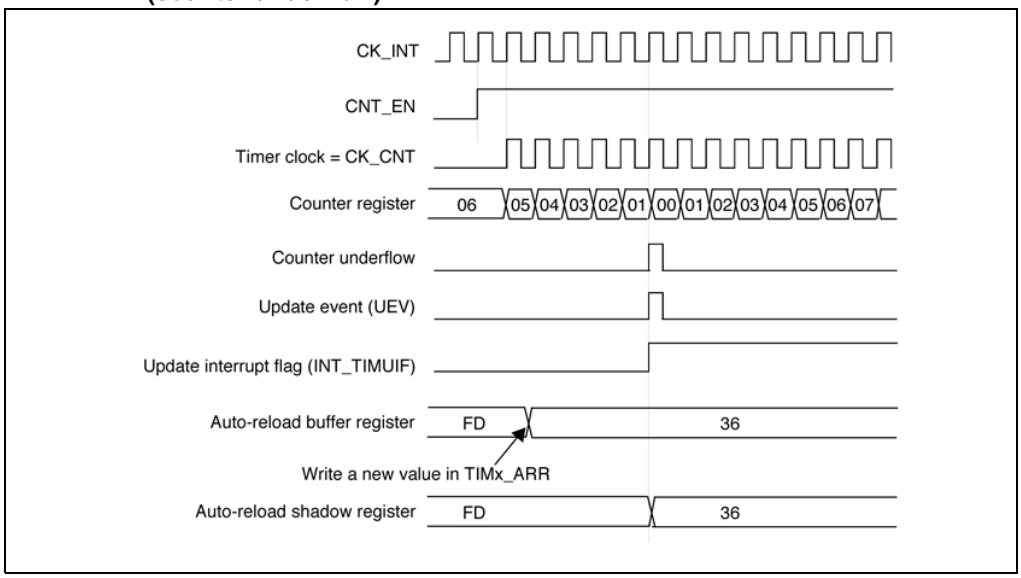
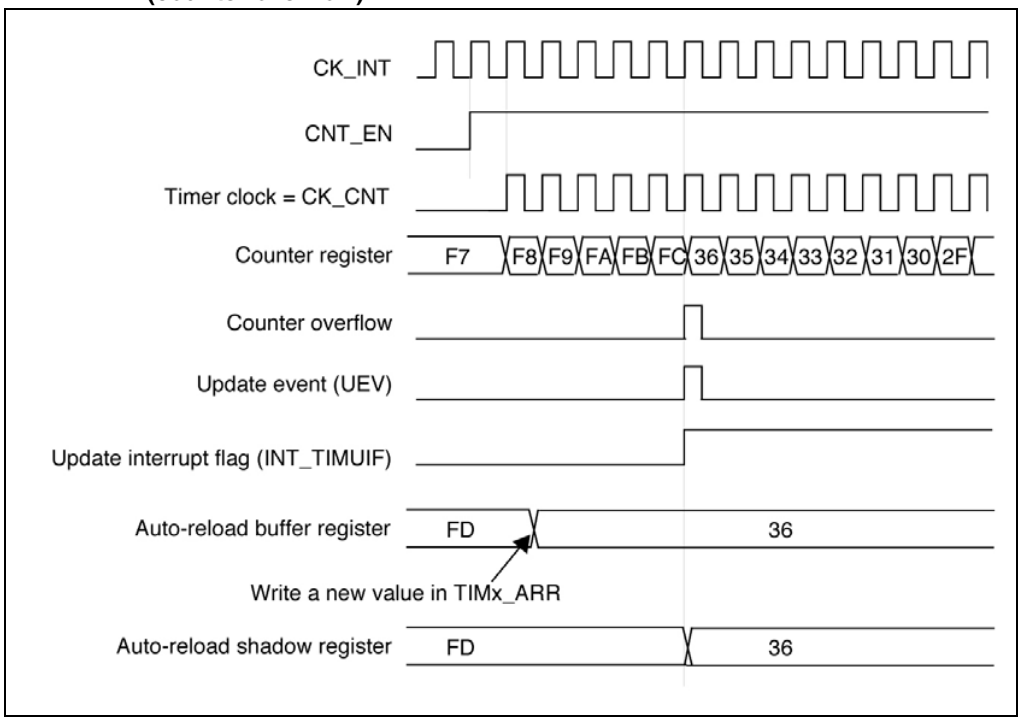


Figure 25. Counter timing diagram, update event with TIM_ARBE = 1 (counter overflow)



10.1.3 Clock selection

The counter clock can be provided by the following clock sources:

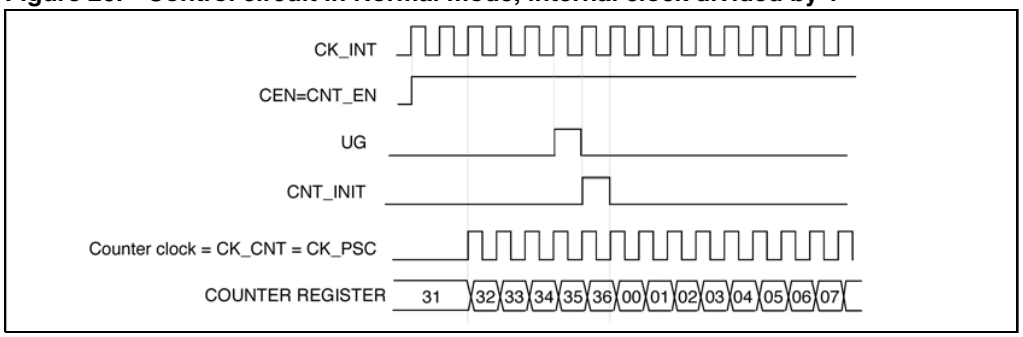
- Internal clock (PCLK)
- External clock mode 1: external input pin (TIy)
- External clock mode 2: external trigger input (ETR)
- Internal trigger input (ITR0): using the other timer as prescaler. Refer to the *Using one timer as prescaler for the other timer* for more details.

Internal clock source (CK_INT)

The internal clock is selected when the slave mode controller is disabled (TIM_SMS = 000 in the TIMx_SMCR register). In this mode, the TIM_CEN, TIM_DIR (in the TIMx_CR1 register), and TIM_UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software, except for TIM_UG, which remains cleared automatically. As soon as the TIM_CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

Figure 26 shows the behavior of the control circuit and the up-counter in normal mode, without prescaling.

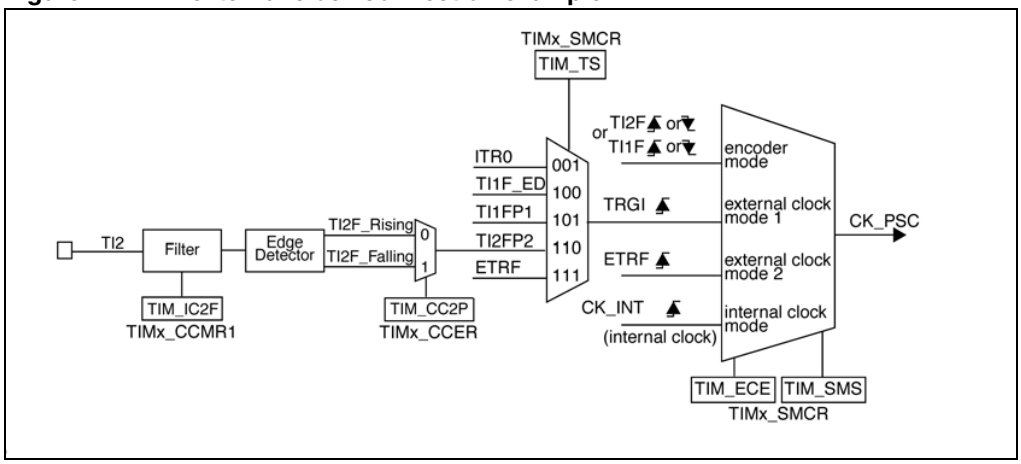
Figure 26. Control circuit in Normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when TIM_SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 27. TI2 external clock connection example



For example, to configure the up-counter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 2 to detect rising edges on the TI2 input by writing TIM_CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the TIM_IC2F bits in the TIMx_CCMR1 register (if no filter is needed, keep TIM_IC2F = 0000).

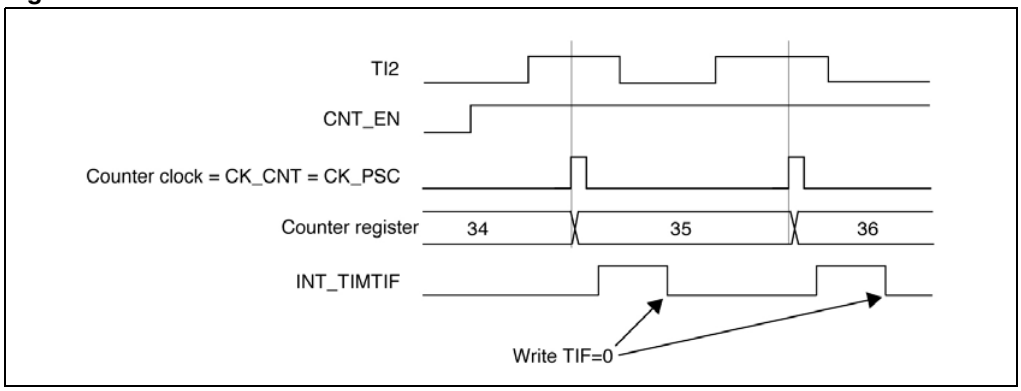
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

3. Select rising edge polarity by writing TIM_CC2P = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing TIM_SMS = 111 in the TIMx_SMCR register.
5. Select TI2 as the input source by writing TIM_TS = 110 in the TIMx_SMCR register.
6. Enable the counter by writing TIM_CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on TI2, the counter counts once and the INT_TIMTIF flag is set.

The delay between the rising edge on TI2 and the actual clock of the counter is due to the resynchronization circuit on the TI2 input.

Figure 28. Control circuit in External Clock mode 1



External clock source mode 2

This mode is selected by writing TIM_ECE = 1 in the TIMx_SMCR register. The counter can count at each rising or falling edge on the external trigger input ETR.

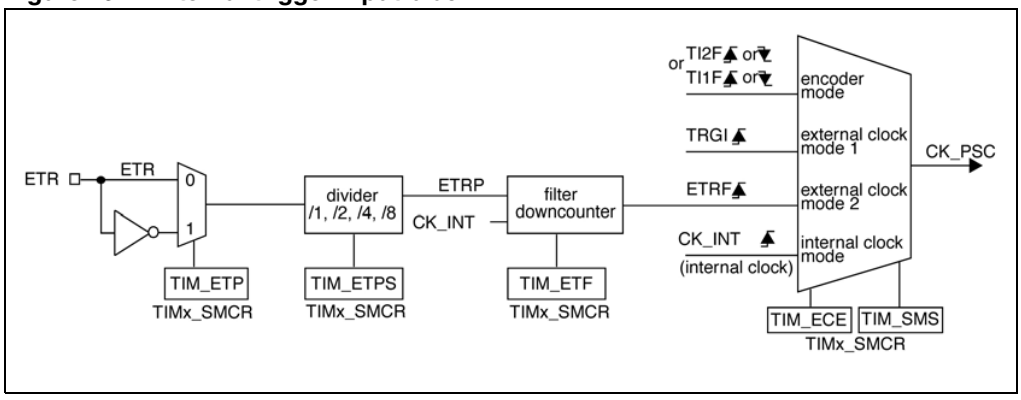
The TIM_EXTRIGSEL bits in the TIMx_OR register select a clock signal that drives ETR, as shown in *Table 85*.

Table 85. TIM_EXTRIGSEL clock signal selection

TIM_EXTRIGSEL bits	Clock signal selection
00	PCLK (peripheral clock). When running from the 24 MHz crystal oscillator, the PCLK frequency is 12 MHz. When the 12M Hz RC oscillator is in use, the frequency is 6 MHz.
01	Calibrated 1 kHz internal RC oscillator
10	Optional 32 kHz clock
11	TIMxCLK pin. If the TIM_CLKMSKEN bit in the TIMx_OR register is set, this signal is AND'ed with the TIMxMSK pin providing a gated clock input.

Figure 29 gives an overview of the external trigger input block.

Figure 29. External trigger input block



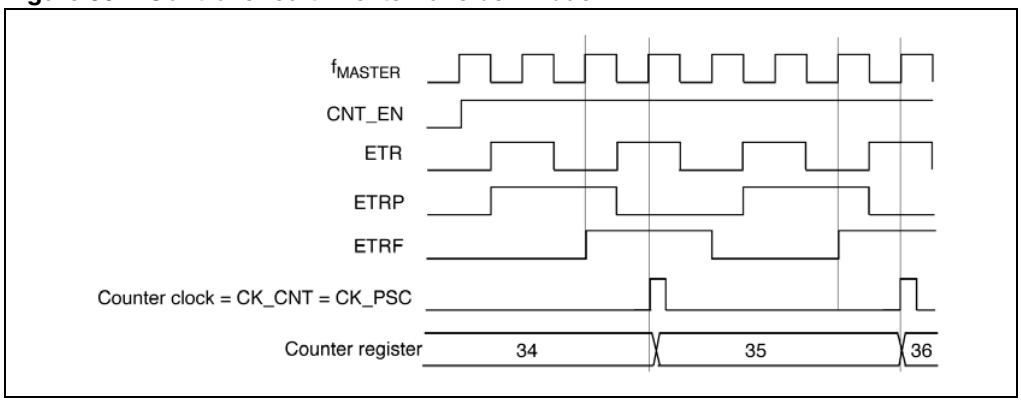
For example, to configure the up-counter to count each 2 rising edges on ETR, use the following procedure:

- As no filter is needed in this example, write TIM_ETF = 0000 in the TIMx_SMCR register.
- Set the prescaler by writing TIM_ETPS = 01 in the TIMx_SMCR register.
- Select rising edge detection on ETR by writing TIM_ETP = 0 in the TIMx_SMCR register.
- Enable external clock mode 2 by writing TIM_ECE = 1 in the TIMx_SMCR register.
- Enable the counter by writing TIM_CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 ETR rising edges.

The delay between the rising edge on ETR and the actual clock of the counter is due to the resynchronization circuit on the ETRP signal.

Figure 30. Control circuit in external clock mode 2

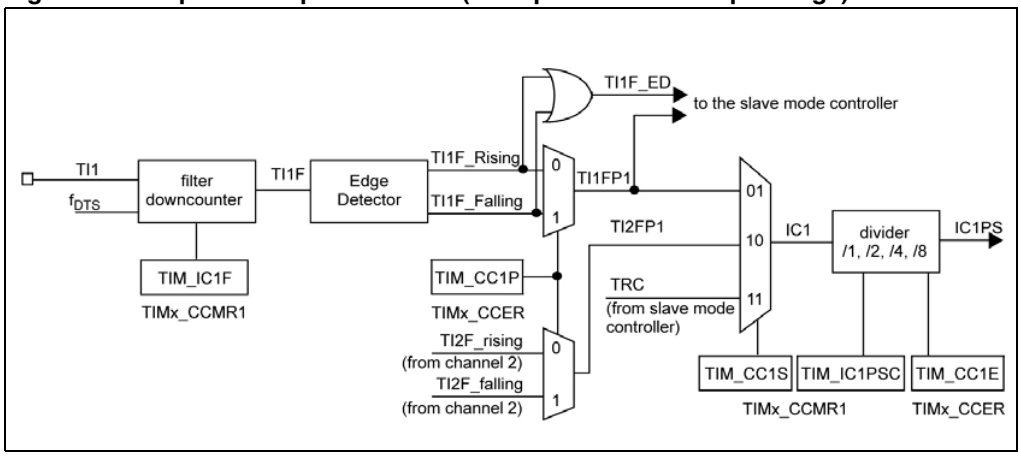


10.1.4 Capture/compare channels

Each capture/compare channel is built around a capture/compare register including a shadow register, an input stage for capture with digital filter, multiplexing and prescaler, and an output stage with comparator and output control.

Figure 31 gives an overview of one capture/compare channel. The input stage samples the corresponding TIy input to generate a filtered signal (TIyF). Then an edge detector with polarity selection generates a signal (TIyFPy) which can be used either as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICyPS).

Figure 31. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate reference signal, OCyREF, which is only used internally. OCyREF is always active high, but it may be inverted to create the output signal, OCy, that controls a GPIO output.

Figure 32. Capture/compare channel 1 main circuit

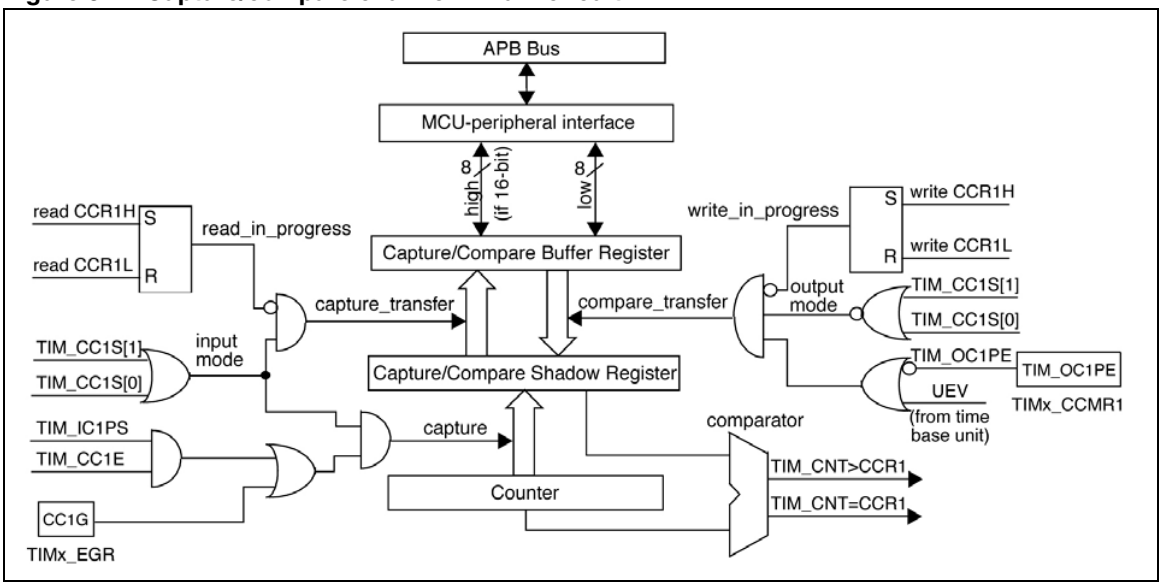
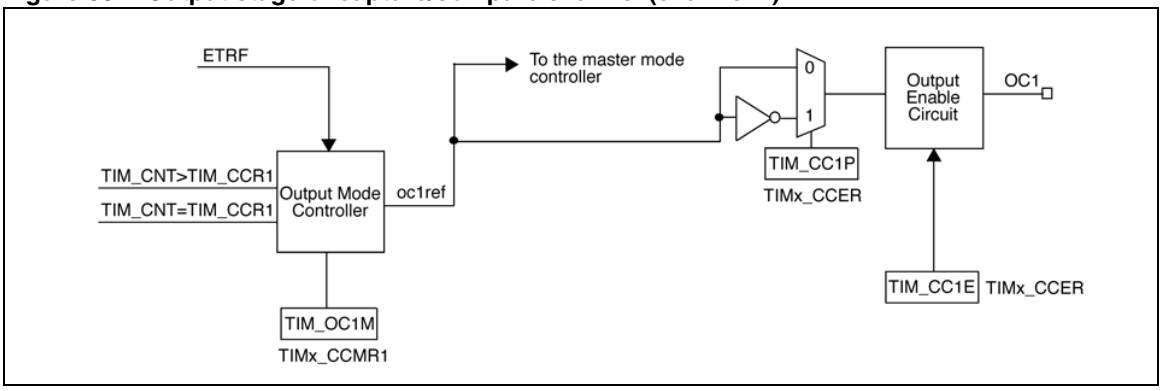


Figure 33. Output stage of capture/compare channel (channel 1)



The capture/compare block is made of a buffer register and a shadow register. Writes and reads always access the buffer register.

In capture mode, captures are first written to the shadow register, then copied into the buffer register.

In compare mode, the content of the buffer register is copied into the shadow register which is compared to the counter.

10.1.5 Input capture mode

In input capture mode, a capture/compare register (TIMx_CCRy) latches the value of the counter after a transition is detected by the corresponding ICy signal. When a capture occurs, the corresponding INT_TIMCCyIF flag in the INT_TIMxFLAG register is set, and an interrupt request is sent if enabled.

If a capture occurs when the INT_TIMCCyIF flag is already high, then the missed capture flag INT_TIMMISSCCyIF in the INT_TIMxMISS register is set. INT_TIMCCyIF can be

cleared by software writing a 1 to its bit or reading the captured data stored in the TIMx_CCRy register. To clear the INT_TIMMISSCCyIF bit, write a 1 to it.

The following example shows how to capture the counter value in the TIMx_CCR1 when the TI1 input rises.

- Select the active input: TIMx_CCR1 must be linked to the TI1 input, so write the TIM_CC1S bits to 01 in the TIMx_CCMR1 register. As soon as TIM_CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
- Program the required input filter duration with respect to the signal connected to the timer, when the input is one of the TIy (ICyF bits in the TIMx_CCMR1 register). Consider a situation in which, when toggling, the input signal is unstable during at most 5 internal clock cycles. The filter duration must be longer than these 5 clock cycles. The transition on TI1 can be validated when 8 consecutive samples with the new level have been detected (sampled at PCLK frequency). To do this, write the TIM_IC1F bits to 0011 in the TIMx_CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing the TIM_CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler: In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write the TIM_IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the TIM_CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the INT_TIMCC1IF bit in the INT_TIMxCFG register.
- When an input capture occurs:
  - The TIMx_CCR1 register gets the value of the counter on the active transition.
  - INT_TIMCC1IF flag is set (capture/compare interrupt flag). The missed capture/compare flag INT_TIMMISSCC1IF in INT_TIMxMISS is also set if another capture occurs before the INT_TIMCC1IF flag is cleared.
  - An interrupt may be generated if enabled by the INT_TIMCC1IF bit.

To detect missed captures reliably, read captured data in TIMxCCRy before checking the missed capture/compare flag. This sequence avoids missing a capture that could happen after reading the flag and before reading the data.

Note: Software can generate IC interrupt requests by setting the corresponding TIM_CCyG bit in the TIMx_EGR register.

10.1.6 PWM input mode

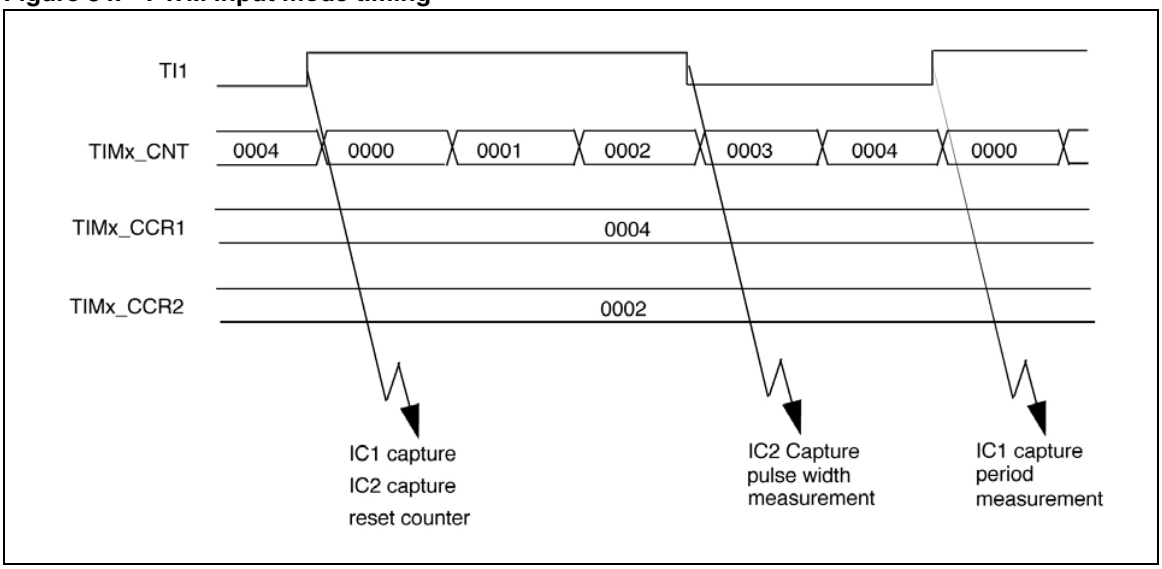
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICy signals are mapped on the same TIy input.
- These two ICy signals are active on edges with opposite polarity.
- One of the two TIyFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, to measure the period in the TIMx_CCR1 register and the duty cycle in the TIMx_CCR2 register of the PWM applied on TI1, use the following procedure depending on CK_INT frequency and prescaler value:

- Select the active input for TIMx_CCR1: write the TIM_CC1S bits to 01 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1, used both for capture in the TIMx_CCR1 and counter clear, by writing the TIM_CC1P bit to 0 (active on rising edge).
- Select the active input for TIMx_CCR2by writing the TIM_CC2S bits to 10 in the TIMx_CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in the TIMx_CCR2) by writing the TIM_CC2P bit to 1 (active on falling edge).
- Select the valid trigger input by writing the TIM_TS bits to 101 in the TIMx_SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode by writing the TIM_SMS bits to 100 in the TIMx_SMCR register.
- Enable the captures by writing the TIM_CC1E and TIM_CC2E bits to 1 in the TIMx_CCER register.

Figure 34. PWM input mode timing



10.1.7 Forced output mode

In output mode (CCyS bits = 00 in the TIMx_CCMR1 register), software can force each output compare signal (OCyREF and then OCy) to an active or inactive level independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCyREF/OCy) to its active level, write 101 in the TIM_OCyM bits in the corresponding TIMx_CCMR1 register. OCyREF is forced high (OCyREF is always active high) and OCy gets the opposite value to the TIM_CCyP polarity bit. For example, TIM_CCyP = 0 defines OCy as active high, so when OCyREF is active, OCy is also set to a high level.

The OCyREF signal can be forced low by writing the TIM_OCyM bits to 100 in the TIMx_CCMR1 register.

The comparison between the TIMx_CCRy shadow register and the counter is still performed and allows the INT_TIMxCCRyIF flag to be set. Interrupt requests can be sent accordingly. This is described in *Section 10.1.8: Output compare mode on page 128*.

10.1.8 Output compare mode

This mode is used to control an output waveform or to indicate when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (the TIM_OCyM bits in the TIMx_CCMR1 register) and the output polarity (the TIM_CCyP bit in the TIMx_CCER register). The output can remain unchanged (TIM_OCyM = 000), be set active (TIM_OCyM = 001), be set inactive (TIM_OCyM = 010), or can toggle (TIM_OCyM = 011) on the match.
- Sets a flag in the interrupt flag register (the INT_TIMCCyIF bit in the INT_TIMxFLAG register).
- Generates an interrupt if the corresponding interrupt mask is set (the TIM_CCyIF bit in the INT_TIMxCFG register).

The TIMx_CCRy registers can be programmed with or without buffer registers using the TIM_OCyBE bit in the TIMx_CCMR1 register.

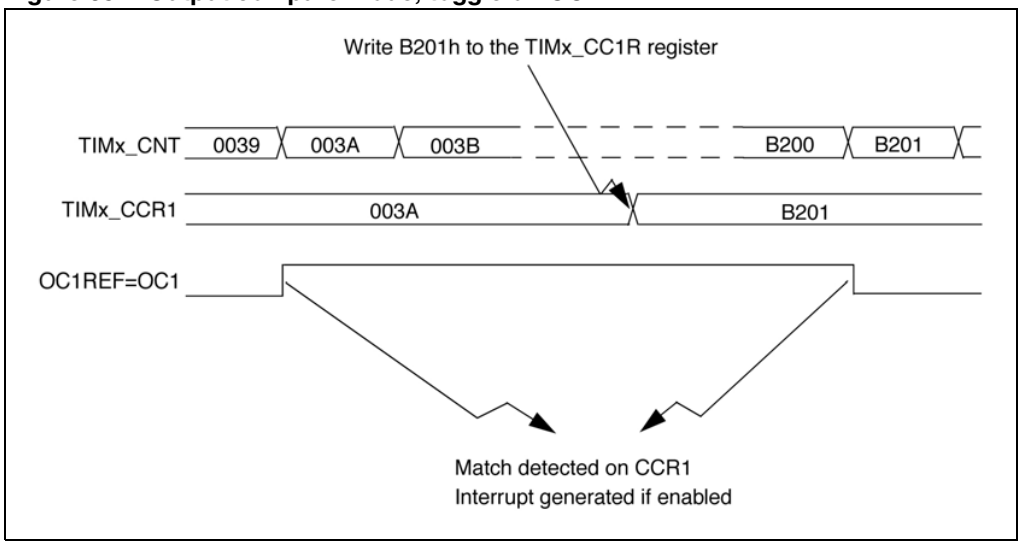
In output compare mode, the update event has no effect on OCyREF or the OCy output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in one pulse mode).

Procedure:

1. Select the counter clock (internal, external, and prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRy registers.
3. Set the INT_TIMCCyIF bit in INT_TIMxCFG if an interrupt request is to be generated.
4. Select the output mode. For example, you must write TIM_OCyM = 011, TIM_OCyBE = 0, TIM_CCyP = 0 and TIM_CCyE = 1 to toggle the OCy output pin when TIMx_CNT matches TIMx_CCRy, TIMx_CCRy buffer is not used, OCy is enabled and active high.
5. Enable the counter by setting the TIM_CEN bit in the TIMx_CR1 register.

To control the output waveform, software can update the TIMx_CCRy register at any time, provided that the buffer register is not enabled (TIM_OCyBE = 0). Otherwise TIMx_CCRy shadow register is updated only at the next update event. An example is given in *Figure 35*.

Figure 35. Output compare mode, toggle on OC1



10.1.9 PWM mode

Pulse width modulation mode allows you to generate a signal with a frequency determined by the value of the TIMx_ARR register, and a duty cycle determined by the value of the TIMx_CCRy register.

PWM mode can be selected independently on each channel (one PWM per OCy output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the TIM_OCyM bits in the TIMx_CCMR1 register. The corresponding buffer register must be enabled by setting the TIM_OCyBE bit in the TIMx_CCMR1 register. Finally, in up-counting or center-aligned mode the auto-reload buffer register must be enabled by setting the TIM_ARBE bit in the TIMx_CR1 register.

Because the buffer registers are only transferred to the shadow registers when an update event occurs, before starting the counter initialize all the registers by setting the TIM_UG bit in the TIMx_EGR register.

OCy polarity is software programmable using the TIM_CCyP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCy output is enabled by the TIM_CCyE bit in the TIMx_CCER register. Refer to the TIMx_CCER register description in the Registers section for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRy are always compared to determine whether TIMx_CCRy ≤TIMx_CNT or TIMx_CNT ≤TIMx_CCRy,depending on the direction of the counter. The OCyREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (TIM_OCyM bits in the TIMx_CCMR1 register) switches from the "frozen" configuration (no comparison, TIM_OCyM = 000) to one of the PWM modes (TIM_OCyM = 110 or 111).

This allows software to force a PWM output to a particular state while the timer is running.

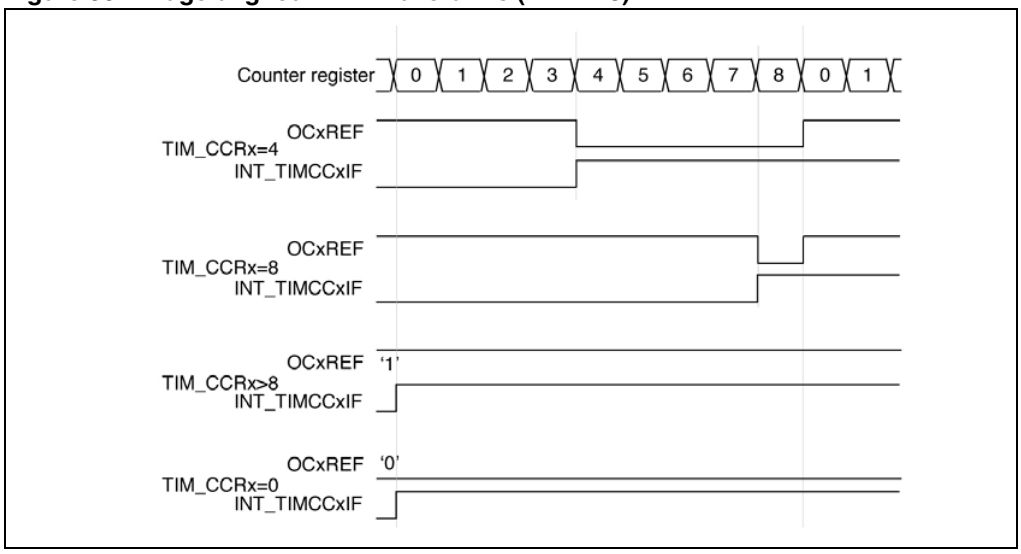
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the TIM_CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode: up-counting configuration

Up-counting is active when the TIM_DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode on page 115*.

The following example uses PWM mode 1. The reference PWM signal OCyREF is high as long as TIMx_CNT < TIMx_CCRy, otherwise it becomes low. If the compare value in TIMx_CCRy is greater than the auto-reload value in TIMx_ARR, then OCyREF is held at 1. If the compare value is 0, then OCyREF is held at 0. *Figure 36* shows some edge-aligned PWM waveforms in an example, where TIMx_ARR = 8.

Figure 36. Edge-aligned PWM waveforms (ARR = 8)



PWM edge-aligned mode: down-counting configuration

Down-counting is active when the TIM_DIR bit in the TIMx_CR1 register is high. Refer to *Down-counting mode on page 117* for more information.

In PWM mode 1, the reference signal OCyREF is low as long as TIMx_CNT > TIMx_CCRy, otherwise it becomes high. If the compare value in TIMx_CCRy is greater than the auto-reload value in TIMx_ARR, then OCyREF is held at 1. Zero-percent PWM is not possible in this mode.

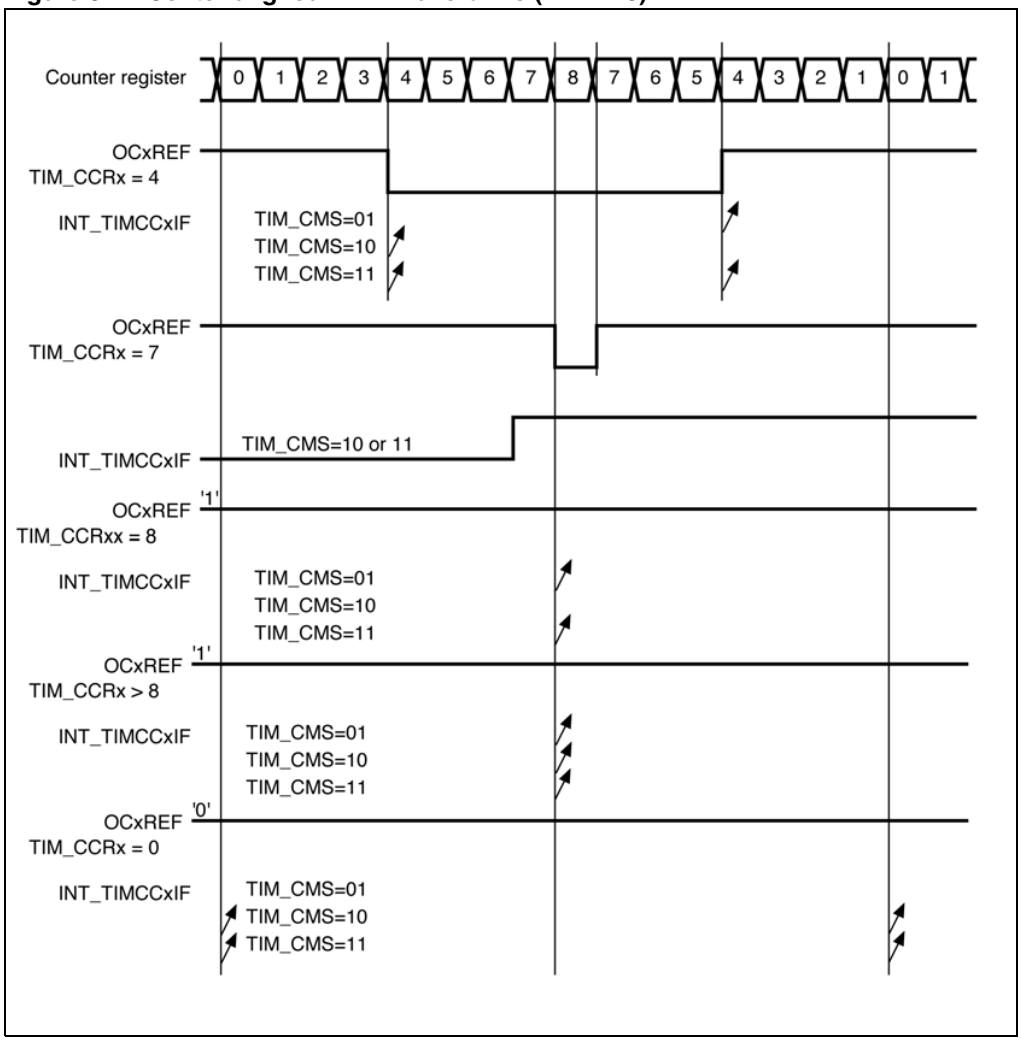
PWM center-aligned mode

Center-aligned mode is active except when the TIM_CMS bits in the TIMx_CR1 register are 00 (all configurations where TIM_CMS is non-zero have the same effect on the OCyREF/OCy signals). The compare flag is set when the counter counts up, when it counts down, or when it counts up and down, depending on the TIM_CMS bits configuration. The direction bit (TIM_DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting) on page 118* for more information.

Figure 37 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8,
- PWM mode is the PWM mode 1,
- The output compare flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for TIM_CMS = 01 in the TIMx_CR1 register.

Figure 37. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. This means that the counter counts up or down depending on the value written in the TIM_DIR bit in the TIMx_CR1 register. The TIM_DIR and TIM_CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:

- The direction is not updated the value written to the counter that is greater than the auto-reload value (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
- The direction is updated if when 0 or the TIMx_ARR value is written to the counter, but no update event is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the TIM_UG bit in the TIMx_EGR register) just before starting the counter, and not to write the counter while it is running.

10.1.10 One-pulse mode

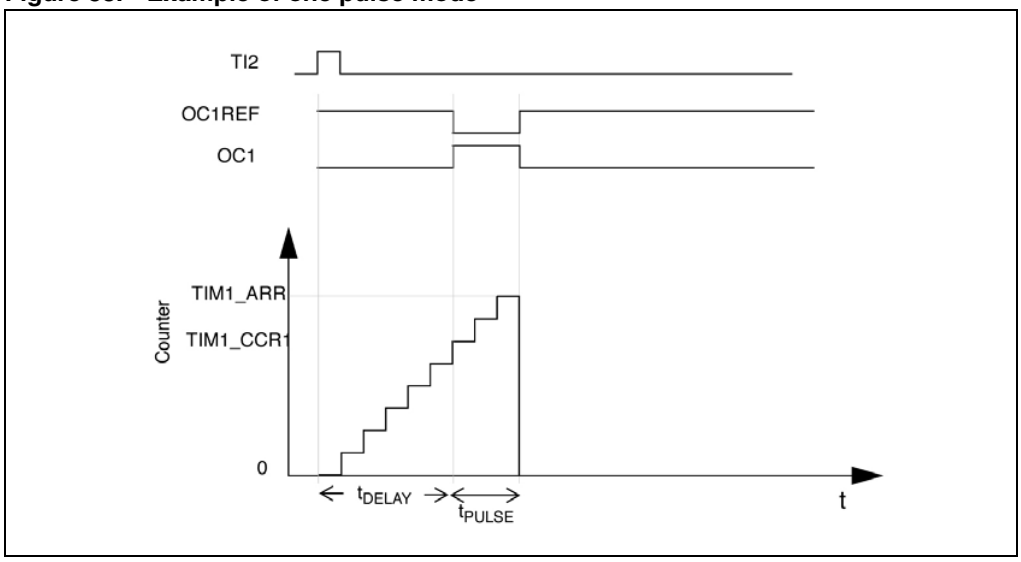
One-pulse mode (OPM) is a special case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. Select OPM by setting the TIM_OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In up-counting: TIMx_CNT < TIMx_CCRy ≤TIMx_ARR (in particular, 0 < TIMx_CCRy),
- In down-counting: TIMx_CNT > TIMx_CCRy.

Figure 38. Example of one pulse mode



For example, to generate a positive pulse on OC1 with a length of tPULSE and after a delay of tDELAY as soon as a rising edge is detected on the TI2 input pin, using TI2FP2 as trigger 1:

- Map TI2FP2 on TI2 by writing TIM_IC2S = 01 in the TIMx_CCMR1 register.
- TI2FP2 must detect a rising edge. Write TIM_CC2P = 0 in the TIMx_CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TIM_TS = 110 in the TIMx_SMCR register.
- TI2FP2 is used to start the counter by writing TIM_SMS to 110 in the TIMx_SMCR register (trigger mode).
- The OPM waveform is defined: Write the compare registers, taking into account the clock frequency and the counter prescaler.

The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.

The tPULSE is defined by the difference between the auto-reload value and the compare value (TIMx_ARR - TIMx_CCR1).

To build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value, enable PWM mode 2 by writing TIM_OC1M = 111 in the TIMx_CCMR1 register. Optionally, enable the buffer registers by writing TIM_OC1BE = 1 in the TIMx_CCMR1 register and TIM_ARBE in the TIMx_CR1 register. In this case, also write the compare value in the TIMx_CCR1 register, the auto-reload value in the TIMx_ARR register, generate an update by setting the TIM_UG bit, and wait for external trigger event on TI2. TIM_CC1P is written to 0 in this example.

In the example, the TIM_DIR and TIM_CMS bits in the TIMx_CR1 register should be low.

Since only one pulse is desired, software should set the TIM_OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0).

A special case: OCy fast enable

In one-pulse mode, the edge detection on the TIy input sets the TIM_CEN bit, which enables the counter. Then the comparison between the counter and the compare value toggles the output. However, several clock cycles are needed for this operation, and it limits the minimum delay (tDELAY min) achievable.

To output a waveform with the minimum delay, set the TIM_OCyFE bit in the TIMx_CCMR1 register. Then OCyREF (and OCy) is forced in response to the stimulus, without taking the comparison into account. Its new level is the same as if a compare match had occurred. TIM_OCyFE acts only if the channel is configured in PWM mode 1 or 2.

10.1.11 Encoder interface mode

To select encoder interface mode, write TIM_SMS = 001 in the TIMx_SMCR register to count only TI2 edges, TIM_SMS = 010 to count only TI1 edges, and TIM_SMS = 011 to count both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the TIM_CC1P and TIM_CC2P bits in the TIMx_CCER register. If needed, program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder (see *Table 86*). Assuming that it is enabled, (the TIM_CEN bit in the TIMx_CR1 register written to 1) the counter is clocked by each valid transition on TI1FP1 or TI2FP2 (TI1 and TI2 after input filter and polarity selection, TI1FP1 = TI1 if not filtered and not inverted, TI2FP2 = TI2 if not

filtered and not inverted.) The sequence of transitions of the two inputs is evaluated, and generates count pulses as well as the direction signal. Depending on the sequence, the counter counts up or down, and hardware modifies the TIM_DIR bit in the TIMx_CR1 register accordingly. The TIM_DIR bit is calculated at each transition on any input (TI1 or TI2), whether the counter is counting on TI1 only, TI2 only, or both TI1 and TI2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TIMx_ARR register (0 to TIMx_ARR or TIMx_ARR down to 0 depending on the direction), so TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, and trigger output features continue to work as normal.

In this mode the counter is modified automatically following the speed and the direction of the incremental encoder, and therefore its contents always represent the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. *Table 86* summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 86. Counting direction versus encoder signals

Active edges	Level on opposite signal (TI1FP1 for TI2, TI2FP2 for TI1)	TI1FP1 signal		TI2FP2 signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No Count	No Count
	Low	Up	Down	No Count	No Count
Counting on TI2 only	High	No Count	No Count	Up	Down
	Low	No Count	No Count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to the MCU without external interface logic. However, comparators are normally used to convert an encoder's differential outputs to digital signals, and this greatly increases noise immunity. If a third encoder output indicates the mechanical zero (or index) position, it may be connected to an external interrupt input and can trigger a counter reset.

Figure 39 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated for when both inputs are used for counting. This might occur if the sensor is positioned near one of the switching points. This example assumes the following configuration:

- TIM_CC1S = 01 (TIMx_CCMR1 register, IC1FP1 mapped on TI1).
- TIM_CC2S = 01 (TIMx_CCMR2 register, IC2FP2 mapped on TI2).
- TIM_CC1P = 0 (TIMx_CCER register, IC1FP1 non-inverted, IC1FP1 = TI1).
- TIM_CC2P = 0 (TIMx_CCER register, IC2FP2 non-inverted, IC2FP2 = TI2).
- TIM_SMS = 011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- TIM_CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 39. Example of counter operation in encoder interface mode

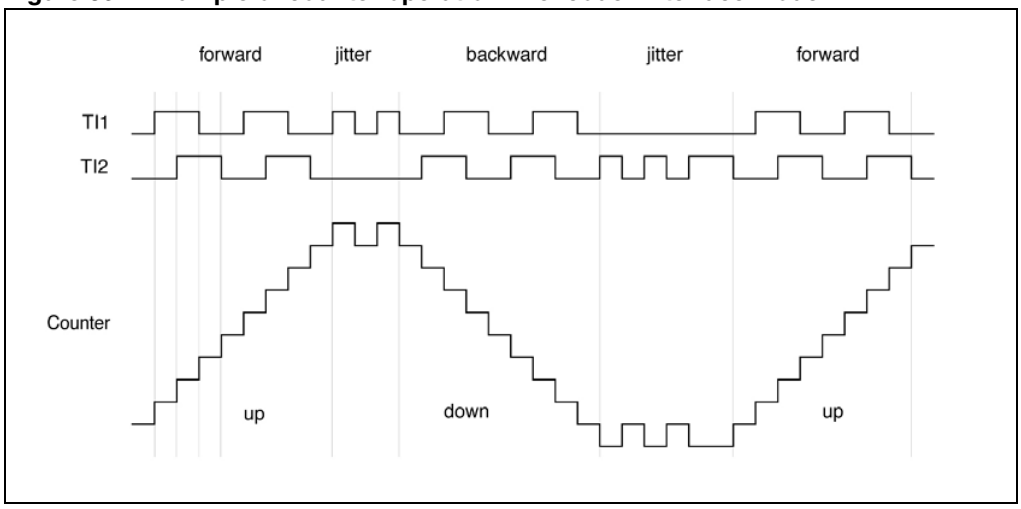
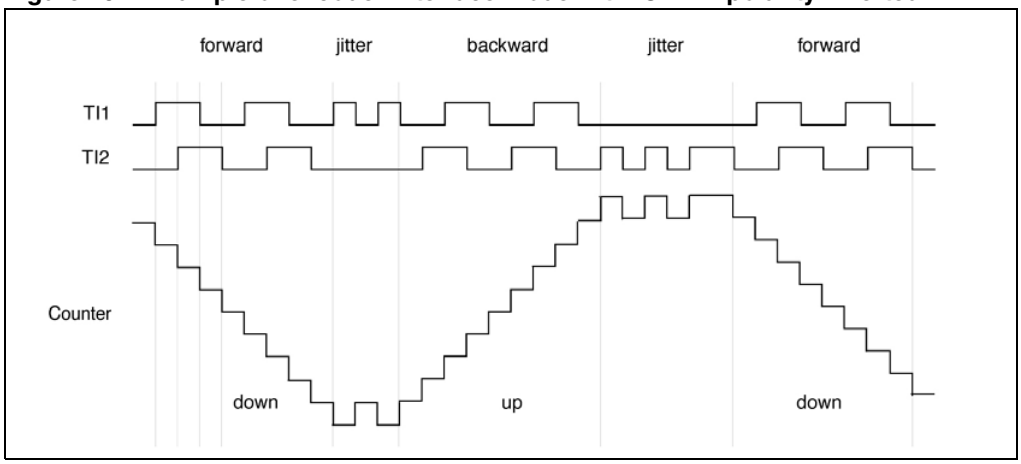


Figure 40 gives an example of counter behavior when IC1FP1 polarity is inverted (same configuration as above except TIM_CC1P = 1).

Figure 40. Example of encoder interface mode with IC1FP1 polarity inverted



The timer configured in encoder interface mode provides information on a sensor's current position. To obtain dynamic information (speed, acceleration/deceleration), measure the period between two encoder events using a second timer configured in capture mode. The output of the encoder that indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. Do this by latching the counter value into a third input capture register. (In this case the capture signal must be periodic and can be generated by another timer).

10.1.12 Timer input XOR function

The TIM_TI1S bit in the TIM1_CR2 register allows the input filter of channel 1 to be connected to the output of a XOR gate that combines the three input pins TIMxC2 to TIMxC4.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is especially useful to interface to Hall effect sensors.

10.1.13 Timers and external trigger synchronization

The timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, and Trigger mode.

Slave mode: Reset mode

Reset mode reinitializes the counter and its prescaler in response to an event on a trigger input. Moreover, if the TIM_URS bit in the TIMx_CR1 register is low, an update event is generated. Then all the buffered registers (TIMx_ARR, TIMx_CCRy) are updated.

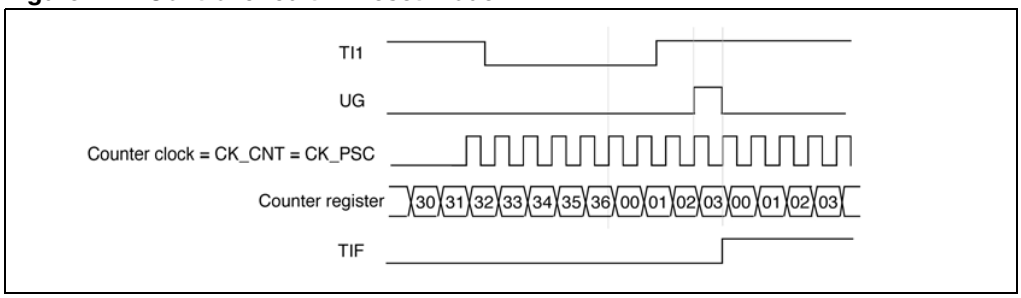
In the following example, the up-counter is cleared in response to a rising edge on the TI1 input:

- Configure the channel 1 to detect rising edges on TI1: Configure the input filter duration. In this example, no filter is required so TIM_IC1F = 0000. The capture prescaler is not used for triggering, so it is not configured. The TIM_CC1S bits select the input capture source only, TIM_CC1S = 01 in the TIMx_CCMR1 register. Write TIM_CC1P = 0 in the TIMx_CCER register to validate the polarity, and detect rising edges only.
- Configure the timer in Reset mode by writing TIM_SMS = 100 in the TIMx_SMCR register. Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.
- Start the counter by writing TIM_CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until the TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (the INT_TIMTIF bit in the INT_TIMxFLAG register) and an interrupt request can be sent if enabled (depending on the INT_TIMTIF bit in the INT_TIMxCFG register).

Figure 41 shows this behavior when the auto-reload register TIMx_ARR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on the TI1 input.

Figure 41. Control circuit in Reset mode



Slave mode: Gated mode

In Gated mode the counter is enabled depending on the level of a selected input.

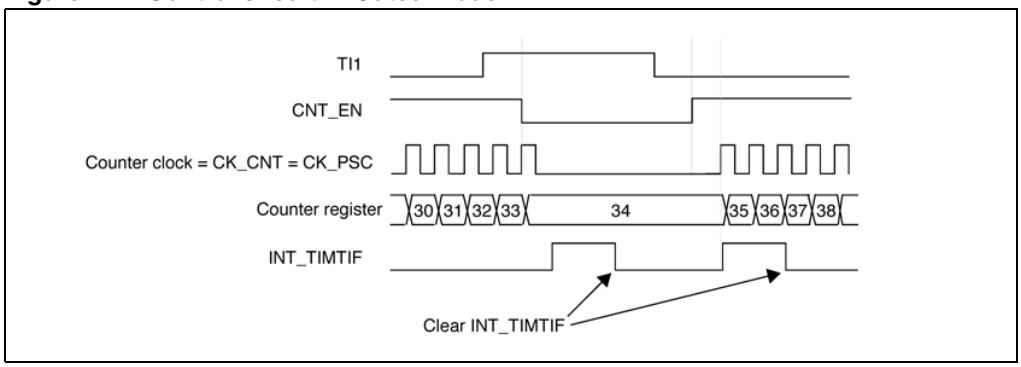
In the following example, the up-counter counts only when the TI1 input is low:

- Configure channel 1 to detect low levels on TI1 Configure the input filter duration. In this example, no filter is required, so TIM_IC1F = 0000. The capture prescaler is not used for triggering, so it is not configured. The TIM_CC1S bits select the input capture source only, TIM_CC1S = 01 in the TIMx_CCMR1 register. Write TIM_CC1P = 1 in the TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in Gated mode by writing TIM_SMS = 101 in the TIMx_SMCR register. Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.
- Enable the counter by writing TIM_CEN = 1 in the TIMx_CR1 register. In Gated mode, the counter does not start if TIM_CEN = 0, regardless of the trigger input level.

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The INT_TIMTIF flag in the INT_TIMxFLAG register is set when the counter starts and when it stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on the TI1 input.

Figure 42. Control circuit in Gated mode



Slave mode: Trigger mode

In Trigger mode the counter starts in response to an event on a selected input.

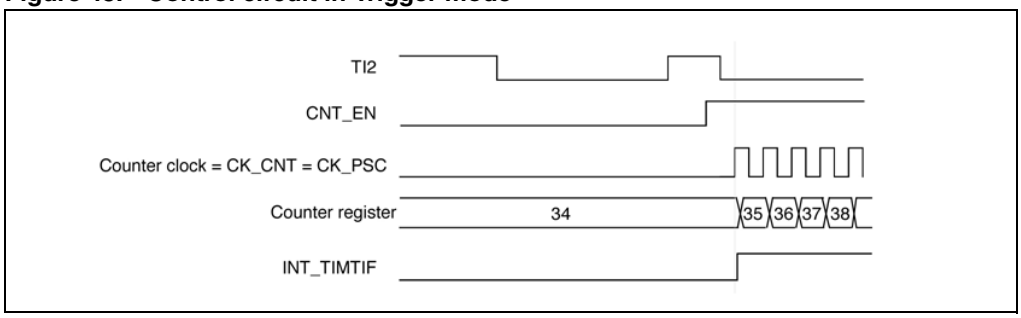
In the following example, the up-counter starts in response to a rising edge on the TI2 input:

- Configure channel 2 to detect rising edges on TI2 Configure the input filter duration. In this example, no filter is required so TIM_IC2F = 0000. The capture prescaler is not used for triggering, so it is not configured. The TIM_CC2S bits select the input capture source only, TIM_CC2S = 01 in the TIMx_CCMR1 register. Write TIM_CC2P = 0 in the TIMx_CCER register to validate the polarity and detect high level only.
- Configure the timer in Trigger mode by writing TIM_SMS = 110 in the TIMx_SMCR register. Select TI2 as the input source by writing TIM_TS = 110 in the TIMx_SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the INT_TIMTIF flag is set.

The delay between the rising edge on TI2 and the actual start of the counter is due to the resynchronization circuit on the TI2 input.

Figure 43. Control circuit in Trigger mode



Slave mode: External clock mode 2 + Trigger mode

External clock mode 2 can be used in combination with another slave mode (except external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode or trigger mode. It is not recommended to select ETR as TRGI through the TIM_TS bits of TIMx_SMCR register.

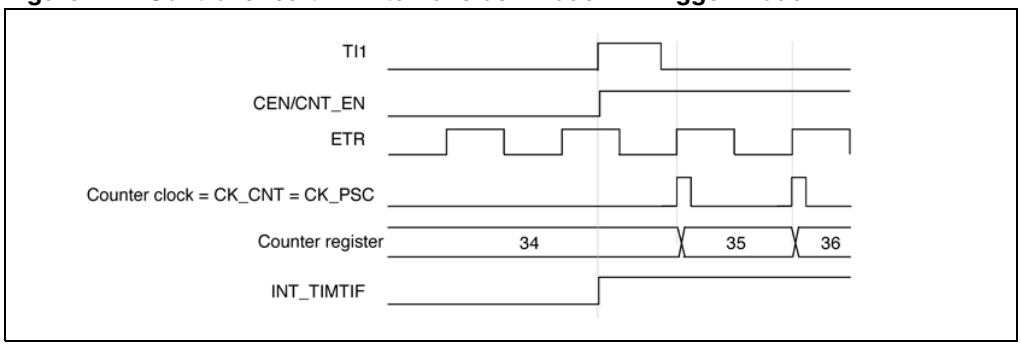
In the following example, the up-counter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

- Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
  - TIM_ETF = 0000: no filter.
  - TIM_ETPS = 00: prescaler disabled.
  - TIM_ETP = 0: detection of rising edges on ETR and TIM_ECE = 1 to enable the external clock mode 2.
- Configure the channel 1 as follows, to detect rising edges on TI:
  - TIM_IC1F = 0000: no filter.
  - The capture prescaler is not used for triggering and does not need to be configured.
  - TIM_CC1S = 01in the TIMx_CCMR1 register to select only the input capture source.
  - TIM_CC1P = 0 in the TIMx_CCER register to validate the polarity (and detect rising edge only).
- Configure the timer in Trigger mode by writing TIM_SMS = 110 in the TIMx_SMCR register. Select TI1 as the input source by writing TIM_TS = 101 in the TIMx_SMCR register.

A rising edge on TI1 enables the counter and sets the INT_TIMTIF flag. The counter then counts on ETR rising edges.

The delay between the rising edge of the ETR signal and the actual reset of the counter is due to the resynchronization circuit on ETRP input.

Figure 44. Control circuit in External clock mode 2 + Trigger mode



10.1.14 Timer synchronization

The two timers can be linked together internally for timer synchronization or chaining. A timer configured in Master mode can reset, start, stop or clock the counter of the other timer configured in Slave mode.

Figure 45 presents an overview of the trigger selection and the master mode selection blocks.

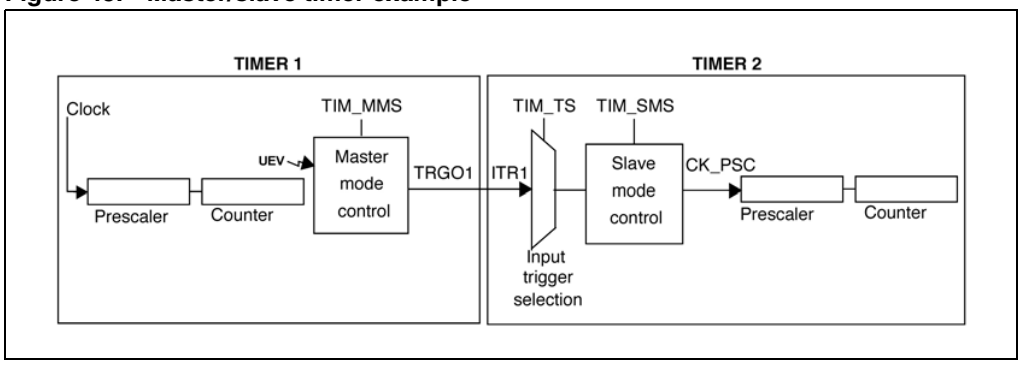
Using one timer as prescaler for the other timer

For example, to configure Timer 1 to act as a prescaler for Timer 2 (see *Figure 45*):

- Configure Timer 1 in Master mode so that it outputs a periodic trigger signal on each update event. Writing TIM_MMS = 010 in the TIM1_CR2 register causes a rising edge to be output on TRGO each time an update event is generated.
- To connect the TRGO output of Timer 1 to Timer 2, configure Timer 2 in slave mode using ITR0 as an internal trigger. Select this through the TIM_TS bits in the TIM2_SMCR register (writing TIM_TS = 000).
- Put the slave mode controller in external clock mode 1 (write TIM_SMS = 111 in the TIM2_SMCR register). This causes Timer 2 to be clocked by the rising edge of the periodic Timer 1 trigger signal (which corresponds to the Timer 1 counter overflow).
- Finally both timers must be enabled by setting their respective TIM_CEN bits (TIMx_CR1 register).

Note: If OCy is selected on Timer 1 as trigger output (TIM_MMS = 1xx), its rising edge is used to clock the counter of Timer 2.

Figure 45. Master/slave timer example



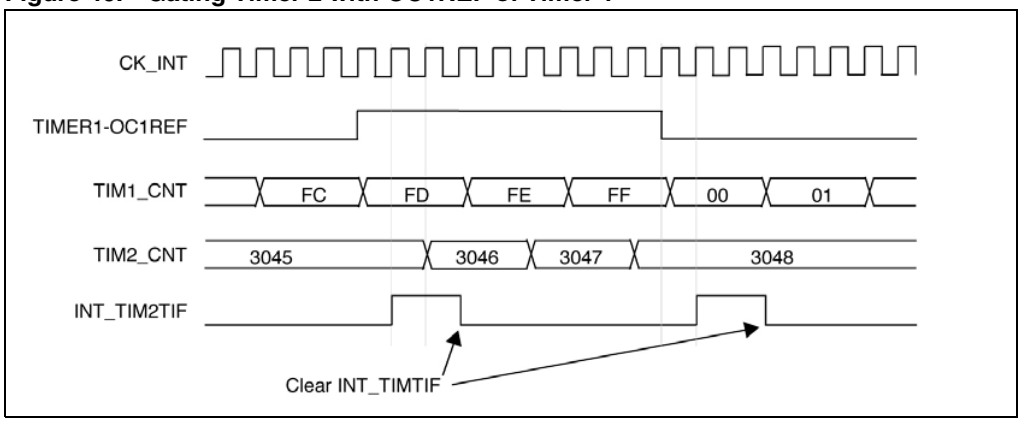
Using one timer to enable the other timer

In this example, the enable of Timer 2 is controlled with the output compare 1 of Timer 1. Refer to *Figure 45* for connections. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (fCK_CNT = fCK_INT /3).

- Configure Timer 1 in master mode to send its Output Compare Reference (OC1REF) signal as trigger output (TIM_MMS = 100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TIM_TS = 000 in the TIM2_SMCR register).
- Configure Timer 2 in Gated mode (TIM_SMS = 101 in the TIM2_SMCR register).
- Enable Timer 2 by writing 1 in the TIM_CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing 1 in the TIM_CEN bit (TIM1_CR1 register).

Note: The counter 2 clock is not synchronized with counter 1, this mode only affects the Timer 2 counter enable signal.

Figure 46. Gating Timer 2 with OC1REF of Timer 1



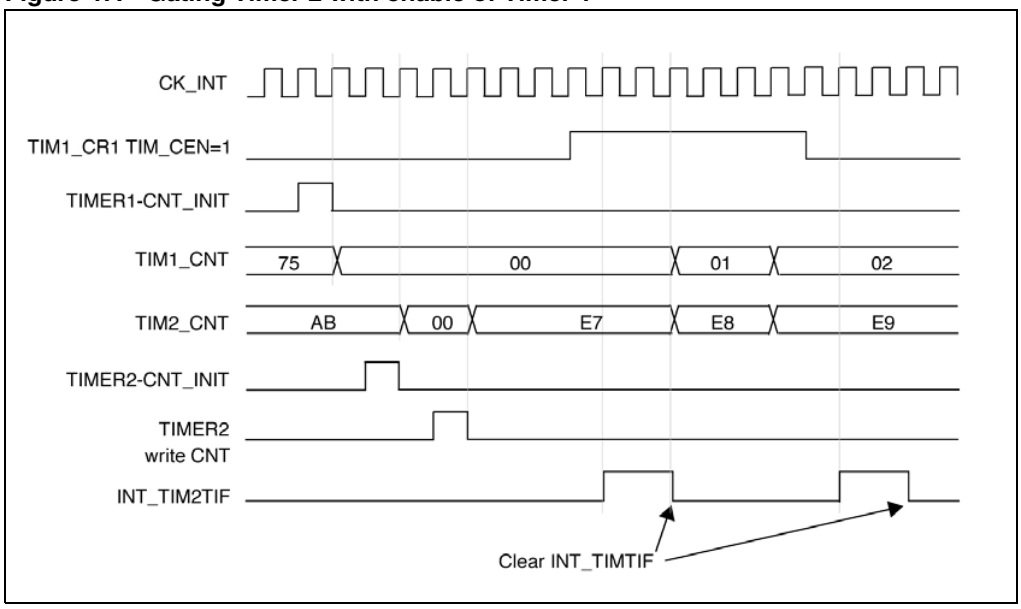
In the example in *Figure 46*, the Timer 2 counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting Timer 1, then writing the desired value in the timer counters. The timers can easily be reset by software using the TIM_UG bit in the TIMx_EGR registers.

The next example, synchronizes Timer 1 and Timer 2. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing 0 to the TIM_CEN bit in the TIM1_CR1 register:

- Configure Timer 1 in master mode to send its Output Compare Reference (OC1REF) signal as trigger output (TIM_MMS = 100 in the TIM1_CR2 register).
- Configure the Timer 1 OC1REF waveform (TIM1_CCMR1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TIM_TS = 000 in the TIM2_SMCR register).
- Configure Timer 2 in gated mode (TIM_SMS = 101 in the TIM2_SMCR register).
- Reset Timer 1 by writing 1 in the TIM_UG bit (TIM1_EGR register).

- Reset Timer 2 by writing 1 in the TIM_UG bit (TIM2_EGR register).
- Initialize Timer 2 to 0xE7 by writing 0xE7 in the Timer 2 counter (TIM2_CNTL).
- Enable Timer 2 by writing 1 in the TIM_CEN bit (TIM2_CR1 register).
- Start Timer 1 by writing 1 in the TIM_CEN bit (TIM1_CR1 register).
- Stop Timer 1 by writing 0 in the TIM_CEN bit (TIM1_CR1 register).

Figure 47. Gating Timer 2 with enable of Timer 1



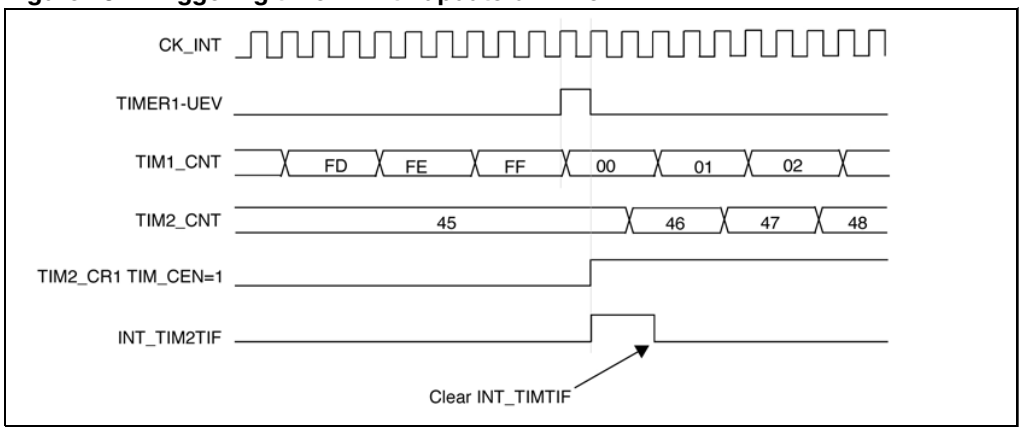
Using one timer to start the other timer

In this example, the enable of Timer 2 is set with the update event of Timer 1. Refer to *Figure 45* for connections. Timer 2 starts counting from its current value (which can be non-zero) on the divided internal clock as soon as Timer 1 generates the update event.

When Timer 2 receives the trigger signal its TIM_CEN bit is automatically set and the counter counts until 0 is written to the TIM_CEN bit in the TIM2_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to CK_INT (fCK_CNT = fCK_INT/3).

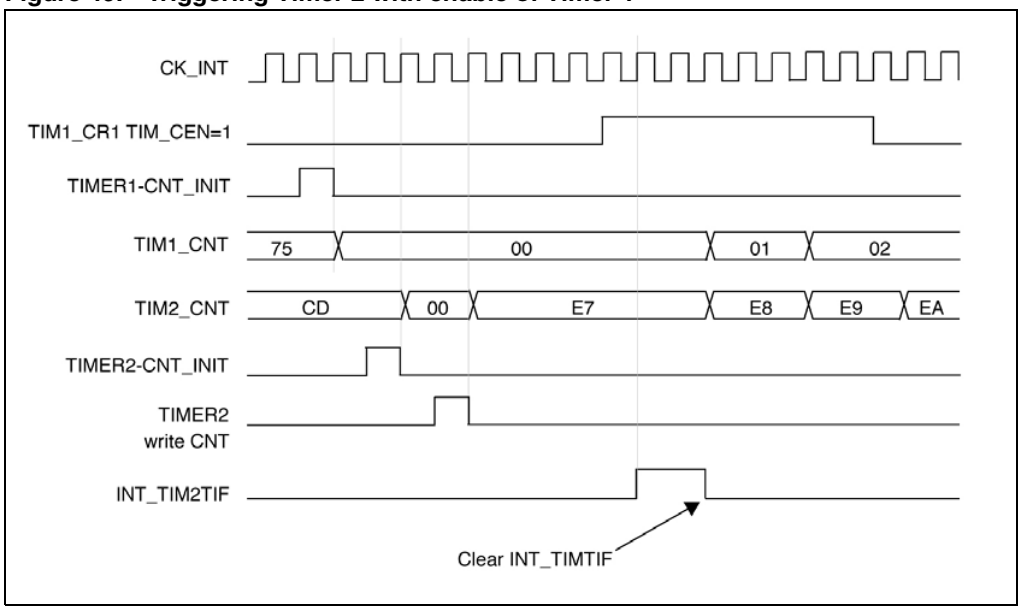
- Configure Timer 1 in master mode to send its update event as trigger output (TIM_MMS = 010 in the TIM1_CR2 register).
- Configure the Timer 1 period (TIM1_ARR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TIM_TS = 000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (TIM_SMS = 110 in the TIM2_SMCR register).
- Start Timer 1: Write 1 in the TIM_CEN bit (TIM1_CR1 register).

Figure 48. Triggering timer 2 with update of Timer 1



As in the previous example, both counters can be initialized before starting counting. *Figure 47* shows the behavior with the same configuration shown in *Figure 48*, but in trigger mode instead of gated mode (TIM_SMS = 110 in the TIM2_SMCR register).

Figure 49. Triggering Timer 2 with enable of Timer 1



Starting both timers synchronously in response to an external trigger

This example, sets the enable of Timer 1 when its TI1 input rises, and the enable of Timer 2 with the enable of Timer 1. Refer to *Figure 45* for connections. To ensure the counters are aligned, Timer 1 must be configured in master/slave mode (slave with respect to TI1, master with respect to Timer 2):

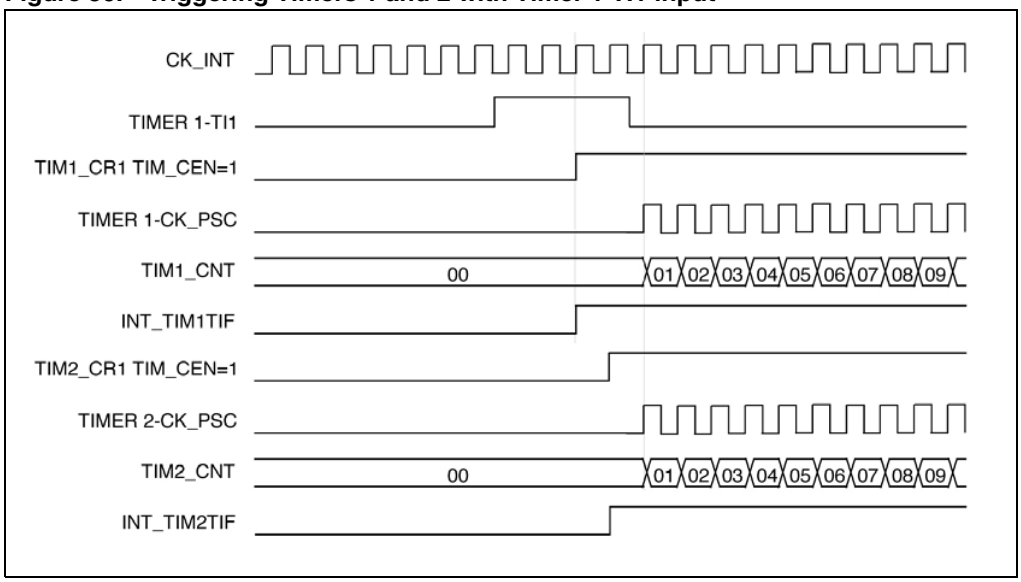
- Configure Timer 1 in master mode to send its Enable as trigger output (TIM_MMS = 001 in the TIM1_CR2 register).
- Configure Timer 1 slave mode to get the input trigger from TI1 (TIM_TS = 100 in the TIM1_SMCR register).
- Configure Timer 1 in trigger mode (TIM_SMS = 110 in the TIM1_SMCR register).

- Configure the Timer 1 in master/slave mode by writing TIM_MSM = 1 (TIM1_SMCR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TIM_TS = 000 in the TIM2_SMCR register).
- Configure Timer 2 in trigger mode (TIM_SMS = 110 in the TIM2_SMCR register).

When a rising edge occurs on TI1 (Timer 1), both counters start counting synchronously on the internal clock and both timers' INT_TIMTIF flags are set.

Note: In this example both timers are initialized before starting by setting their respective TIM_UG bits. Both counters starts from 0, but an offset can be inserted between them by writing any of the counter registers (TIMx_CNT). The master/slave mode inserts a delay between CNT_EN and CK_PSC on Timer 1.

Figure 50. Triggering Timers 1 and 2 with Timer 1 TI1 input



10.1.15 Timer signal descriptions

Table 87. Timer signal descriptions

Signal	Internal/external	Description
CK_INT	Internal	Internal clock source: connects to STM32W108 peripheral clock (PCLK) in internal clock mode.
CK_PSC	Internal	Input to the clock prescaler.
ETR	Internal	External trigger input (used in external timer mode 2): a clock selected by TIM_EXTRIGSEL in TIMx_OR.
ETRF	Internal	External trigger: ETRP after filtering.
ETRP	Internal	External trigger: ETR after polarity selection, edge detection and prescaling.
ICy	External	Input capture or clock: TIy after filtering and edge detection.

Table 87. Timer signal descriptions (continued)

Signal	Internal/external	Description
ICyPS	Internal	Input capture signal after filtering, edge detection and prescaling: input to the capture register.
ITR0	Internal	Internal trigger input: connected to the other timer's output, TRGO.
OCy	External	Output compare: TIMxCy when used as an output. Same as OCyREF but includes possible polarity inversion.
OCyREF	Internal	Output compare reference: always active high, but may be inverted to produce OCy.
PCLK	External	Peripheral clock connects to CK_INT and used to clock input filtering. Its frequency is 12MHz if using the 24MHz crystal oscillator and 6Mhz if using the 12MHz RC oscillator.
TIy	Internal	Timer input: TIMxCy when used as a timer input.
TIyFPy	Internal	Timer input after filtering and polarity selection.
TIMxCy	Internal	Timer channel at a GPIO pin: can be a capture input (ICy) or a compare output (OCy).
TIMxCLK	External	Clock input (if selected) to the external trigger signal (ETR).
TIMxMSK	External	Clock mask (if enabled) AND'ed with the other timer's TIMxCLK signal.
TRGI	Internal	Trigger input for slave mode controller.

10.2 Interrupts

Each timer has its own ARM® Cortex-M3 vectored interrupt with programmable priority. Writing 1 to the INT_TIMx bit in the INT_CFGSET register enables the TIMx interrupt, and writing 1 to the INT_TIMx bit in the INT_CFGCLR register disables it. *Section 12: Interrupts on page 177* describes the interrupt system in detail.

Several kinds of timer events can generate a timer interrupt, and each has a status flag in the INT_TIMxFLAG register to identify the reason(s) for the interrupt:

- INT_TIMTIF - set by a rising edge on an external trigger, either edge in gated mode
- INT_TIMCCRyIF -set by a channel y input capture or output compare event
- INT_TIMUIF - set by an update event

Clear bits in INT_TIMxFLAG by writing a 1 to their bit position. When a channel is in capture mode, reading the TIMx_CCRy register will also clear the INT_TIMCCRyIF bit.

The INT_TIMxCFG register controls whether or not the INT_TIMxFLAG bits actually request an ARM® Cortex-M3 timer interrupt. Only the events whose bits are set to 1 in INT_TIMxCFG can do so.

If an input capture or output compare event occurs and its INT_TIMMISSCCyIF is already set, the corresponding capture/compare missed flag is set in the INT_TMRxMISS register. Clear a bit in the INT_TMRxMISS register by writing a 1 to it.

10.3 General-purpose timer (1 and 2) registers

10.3.1 Timer *x* control register 1 (TIMx_CR1)

Address offset: 0xE000 (TIM1) and 0xF000 (TIM2)
Reset value: 0x0000 0000

Table 88. Timer *x* control register 1 (TIMx_CR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIM_ARBE	TIM_CMS		TIM_DIR	TIM_OPM	TIM_URS	TIM_UDIS	TIM_CEN
								rw	rw		rw	rw	rw	rw	rw

Bit 7 TIM_ARBE: Auto-Reload Buffer Enable

0: TIMx_ARR register is not buffered.

1: TIMx_ARR register is buffered.

Bits [6:5] TIM_CMS: Center-aligned Mode Selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (TIM_DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively.

Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_CCMRy register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively.

Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_CCMRy register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively.

Output compare interrupt flags of configured output channels (TIM_CCyS=00 in TIMx_CCMRy register) are set both when the counter is counting up or down.

Note: Software may not switch from edge-aligned mode to center-aligned mode when the counter is enabled (TIM_CEN=1).

Bit 4 TIM_DIR: Direction

0: Counter used as up-counter.

1: Counter used as down-counter.

Bit 3 TIM_OPM: One Pulse Mode

0: Counter does not stop counting at the next update event.

1: Counter stops counting at the next update event (and clears the bit TIM_CEN).

Bit 2 TIM_URS: Update Request Source

0: When enabled, update interrupt requests are sent as soon as registers are updated (counter overflow/underflow, setting the TIM_UG bit, or update generation through the slave mode controller).

1: When enabled, update interrupt requests are sent only when the counter reaches overflow or underflow.

Bit 1 TIM_UDIS: Update Disable

0: An update event is generated as soon as a counter overflow occurs, a software update is generated, or a hardware reset is generated by the slave mode controller. Shadow registers are then loaded with their buffer register values.

1: An update event is not generated and shadow registers keep their value (TIMx_ARR, TIMx_PSC, TIMx_CCRy). The counter and the prescaler are reinitialized if the TIM_UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 TIM_CEN: Counter Enable

0: Counter disabled.

1: Counter enabled.

Note: External clock, gated mode and encoder mode can work only if the TIM_CEN bit has been previously set by software. Trigger mode sets the TIM_CEN bit automatically through hardware.

10.3.2 Timer *x* control register 2 (TIMx_CR2)

Address offset: 0xE004 (TIM1) and 0xF004 (TIM2)
Reset value: 0x0000 0000

Table 89. Timer *x* control register 2 (TIMx_CR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIM_TI1S	TIM_MMS			Reserved			
								rw	rw						

Bit 7 TIM_TI1S: TI1 Selection

0: TI1M (input of the digital filter) is connected to TI1 input.

1: TI1M is connected to the TI_HALL inputs (XOR combination).

Bits [6:4] TIM_MMS: Master Mode Selection

This selects the information to be sent in master mode to a slave timer for synchronization using the trigger output (TRGO).

000: Reset - the TIM_UG bit in the TMRx_EGR register is trigger output.

If the reset is generated by the trigger input (slave mode controller configured in reset mode), then the signal on TRGO is delayed compared to the actual reset.

001: Enable - counter enable signal CNT_EN is trigger output.

This mode is used to start both timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by either the TIM_CEN control bit or the trigger input when configured in gated mode. When the counter enable signal is controlled by the trigger input there is a delay on TRGO except if the master/slave mode is selected (see the TIM_MSM bit description in TMRx_SMCR register).

010: Update - update event is trigger output.

This mode allows a master timer to be a prescaler for a slave timer.

011: Compare Pulse.

The trigger output sends a positive pulse when the TIM_CC1IF flag is to be set (even if it was already high) as soon as a capture or a compare match occurs.

100: Compare - OC1REF signal is trigger output.

101: Compare - OC2REF signal is trigger output.

110: Compare - OC3REF signal is trigger output.

111: Compare - OC4REF signal is trigger output.

10.3.3 Timer *x* slave mode control register (TIMx_SMCR)

Address offset: 0xE008 (TIM1) and 0xF008 (TIM2)
Reset value: 0x0000 0000

Table 90. Timer *x* slave mode control register (TIMx_SMCR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM_ETP	TIM_ECE	TIM_ETPS		TIM_ETF				TIM_MSM	TIM_TS			Reserved	TIM_SMS		
rw	rw	rw		rw				rw	rw				rw		

Bit 15 TIM_ETP: External Trigger Polarity

This bit selects whether ETR or the inverse of ETR is used for trigger operations.

0: ETR is non-inverted, active at a high level or rising edge.

1: ETR is inverted, active at a low level or falling edge.

Bit 14 TIM_ECE: External Clock Enable

This bit enables external clock mode 2.

0: External clock mode 2 disabled.

1: External clock mode 2 enabled. The counter is clocked by any active edge on the ETRF signal.

Note: Setting the TIM_ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (TIM_SMS=111 and TIM_TS=111).

It is possible to use this mode simultaneously with the following slave modes: reset mode, gated mode and trigger mode. TRGI must not be connected to ETRF in this case (the TIM_TS bits must not be 111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input will be ETRF.

Bits [13:12] TIM_ETPS: External Trigger Prescaler

External trigger signal ETRP frequency must be at most 1/4 of CK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful with fast external clocks.

00: ETRP prescaler off.

01: Divide ETRP frequency by 2.

10: Divide ETRP frequency by 4.

11: Divide ETRP frequency by 8.

Bits [11:8] TIM_ETF: External Trigger Filter

This defines the frequency used to sample the ETRP signal, $f_{Sampling}$, and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: $f_{Sampling}$ = PCLK, no filtering.
0001: $f_{Sampling}$ = PCLK, N=2.
0010: $f_{Sampling}$ = PCLK, N=4.
0011: $f_{Sampling}$ = PCLK, N=8.
0100: $f_{Sampling}$ = PCLK/2, N=6.
0101: $f_{Sampling}$ = PCLK/2, N=8.
0110: $f_{Sampling}$ = PCLK/4, N=6.
0111: $f_{Sampling}$ = PCLK/4, N=8.
1000: $f_{Sampling}$ = PCLK/8, N=6.
1001: $f_{Sampling}$ = PCLK/8, N=8.
1010: $f_{Sampling}$ = PCLK/16, N=5.
1011: $f_{Sampling}$ = PCLK/16, N=6.
1100: $f_{Sampling}$ = PCLK/16, N=8.
1101: $f_{Sampling}$ = PCLK/32, N=5.
1110: $f_{Sampling}$ = PCLK/32, N=6.
1111: $f_{Sampling}$ = PCLK/32, N=8.

Note: PCLK is 12 MHz when the STM32W108 is using the 24 MHz crystal oscillator, and 6 MHz if using the 12 MHz RC oscillator.

Bit 7 TIM_MSM: Master/Slave Mode

0: No action.

1: The effect of an event on the trigger input (TRGI) is delayed to allow exact synchronization between the current timer and the slave (through TRGO). It is useful for synchronizing timers on a single external event.

Bits [6:4] TIM_TS: Trigger Selection

This bit field selects the trigger input used to synchronize the counter.

000 : Internal Trigger 0 (ITR0).

100 : TI1 Edge Detector (TI1F_ED).

101 : Filtered Timer Input 1 (TI1FP1).

110 : Filtered Timer Input 2 (TI2FP2).

111 : External Trigger input (ETRF).

Note: These bits must be changed only when they are not used (when TIM_SMS=000) to avoid detecting spurious edges during the transition.

Bits [2:0] TIM_SMS: Slave Mode Selection

When external signals are selected the active edge of the trigger signal (TRGI) is linked to the polarity selected on the external input.

000: Slave mode disabled.

If TIM_CEN = 1 then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1. Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.

010: Encoder mode 2. Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.

011: Encoder mode 3. Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.

100: Reset Mode. Rising edge of the selected trigger signal (TRGI) >reinitializes the counter and generates an update of the registers.

101: Gated Mode. The counter clock is enabled when the trigger signal (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both starting and stopping the counter are controlled.

110: Trigger Mode. The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only starting the counter is controlled.

111: External Clock Mode 1. Rising edges of the selected trigger (TRGI) clock the counter.

Note: Gated mode must not be used if TI1F_ED is selected as the trigger input (TIM_TS=100). TI1F_ED outputs 1 pulse for each transition on TI1F, whereas gated mode checks the level of the trigger signal.

10.3.4 Timer *x* event generation register (TIMx_EGR)

Address offset: 0xE014 (TIM1) and 0xF014 (TIM2)
Reset value: 0x0000 0000

Table 91. Timer *x* event generation register (TIMx_EGR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									TIM_TG	Reserved	TIM_CC4G	TIM_CC3G	TIM_CC2G	TIM_CC1G	TIM_UG
									w		w	w	w	w	w

Bit 6 TIM_TG: Trigger Generation
0: Does nothing.
1: Sets the TIM_TIF flag in the INT_TIMxFLAG register.

Bit 4 TIM_CC4G: Capture/Compare 4 Generation
0: Does nothing.
1: If CC4 configured as output channel:
The TIM_CC4IF flag is set.
If CC4 configured as input channel:
The TIM_CC4IF flag is set.
The INT_TIMMISSCC4IF flag is set if the TIM_CC4IF flag was already high.
The current value of the counter is captured in TMRx_CCR4 register.

Bit 3 TIM_CC3G: Capture/Compare 3 Generation
0: Does nothing.
1: If CC3 configured as output channel:
The TIM_CC3IF flag is set.
If CC3 configured as input channel:
The TIM_CC3IF flag is set.
The INT_TIMMISSCC3IF flag is set if the TIM_CC3IF flag was already high.
The current value of the counter is captured in TMRx_CCR3 register.

Bit 2 TIM_CC2G: Capture/Compare 2 Generation
0: Does nothing.
1: If CC2 configured as output channel:
The TIM_CC2IF flag is set.
If CC2 configured as input channel:
The TIM_CC2IF flag is set.
The INT_TIMMISSCC2IF flag is set if the TIM_CC2IF flag was already high.
The current value of the counter is captured in TMRx_CCR2 register.

Bit 1 TIM_CC1G: Capture/Compare 1 Generation
0: Does nothing.
1: If CC1 configured as output channel:
The TIM_CC1IF flag is set.
If CC1 configured as input channel:
The TIM_CC1IF flag is set.
The INT_TIMMISSCC1IF flag is set if the TIM_CC1IF flag was already high.
The current value of the counter is captured in TMRx_CCR1 register.

Bit 0 TIM_UG: Update Generation
0: Does nothing.
1: Re-initializes the counter and generates an update of the registers. This also clears the prescaler counter but the prescaler ratio is not affected. The counter is cleared if center-aligned mode is selected or if TIM_DIR=0 (up-counting), otherwise it takes the auto-reload value (TMR1_ARR) if TIM_DIR=1 (down-counting).

10.3.5 Timer x capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0xE018 (TIM1) and 0xF018 (TIM2)
Reset value: 0x0000 0000

Table 92. Timer x capture/compare mode register 1 (TIMx_CCMR1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIM_OC2M			TIM_OC2BE	TIM_OC2FE	TIM_CC2S			TIM_OC1M			TIM_OC1BE	TIM_OC1FE	TIM_CC1S	
TIM_IC2F				TIM_IC2PSC				TIM_IC1F				TIM_IC1PSC			
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	

Timer channels can be programmed as inputs (capture mode) or outputs (compare mode). The direction of channel y is defined by TIM_CCyS in this register.

The other bits in this register have different functions in input and in output modes. The TIM_OC* fields only apply to a channel configured as an output (TIM_CCyS = 0), and the TIM_IC* fields only apply to a channel configured as an input (TIM_CCyS > 0).

Bits [14:12] TIM_OC2M: Output Compare 2 Mode. (Applies only if TIM_CC2S = 0

Define the behavior of the output reference signal OC2REF from which OC2 derives. OC2REF is active high whereas OC2"s active level depends on the TIM_CC2P bit.

000: Frozen - The comparison between the output compare register TIMx_CCR2 and the counter TIMx_CNT has no effect on the outputs.

001: Set OC2REF to active on match. The OC2REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 2 (TIMx_CCR2)

010: Set OC2REF to inactive on match. OC2REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 2 (TIMx_CCR2).

011: Toggle - OC2REF toggles when TIMx_CNT = TIMx_CCR2.

100: Force OC2REF inactive.

101: Force OC2REF active.

110: PWM mode 1 - In up-counting, OC2REF is active as long as TIMx_CNT < TIMx_CCR2, otherwise OC2REF is inactive. In down-counting, OC2REF is inactive if TIMx_CNT > TIMx_CCR2, otherwise OC2REF is active.

111: PWM mode 2 - In up-counting, OC2REF is inactive if TIMx_CNT < TIMx_CCR2, otherwise OC2REF is active. In down-counting, OC2REF is active if TIMx_CNT > TIMx_CCR2, otherwise it is inactive.

Note: In PWM mode 1 or 2, the OC2REF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 11 TIM_OC2BE: Output Compare 2 Buffer Enable. (Applies only if TIM_CC2S = 0

0: Buffer register for TIMx_CCR2 is disabled. TIMx_CCR2 can be written at anytime, the new value is used by the shadow register immediately.

1: Buffer register for TIMx_CCR2 is enabled. Read/write operations access the buffer register. TIMx_CCR2 buffer value is loaded in the shadow register at each update event.

Note: The PWM mode can be used without enabling the buffer register only in one pulse mode (TIM_OPM bit set in the TIMx_CR2 register), otherwise the behavior is undefined.

Bit 10 TIM_OC2FE: Output Compare 2 Fast Enable. (Applies only if TIM_CC2S = 0)

This bit speeds the effect of an event on the trigger in input on the OC2 output.

0: OC2 behaves normally depending on the counter and TIM_CCR2 values even when the trigger is ON. The minimum delay to activate OC2 when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on the OC2 output. OC2 is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate OC2 output is reduced to 3 clock cycles. TIM_OC2FE acts only if the channel is configured in PWM 1 or PWM 2 mode.

Bits [15:12] TIM_IC2F: Input Capture 1 Filter. (Applies only if TIM_CC2S > 0)

This defines the frequency used to sample the TI2 input, Fsampling, and the length of the digital filter applied to TI2. The digital filter requires N consecutive samples in the same state before being output.

0000: Fsampling=PCLK, no filtering.
0001: Fsampling=PCLK, N=2.
0010: Fsampling=PCLK, N=4.
0011: Fsampling=PCLK, N=8.
0100: Fsampling=PCLK/2, N=6.
0101: Fsampling=PCLK/2, N=8.
0110: Fsampling=PCLK/4, N=6.
0111: Fsampling=PCLK/4, N=8.
1000: Fsampling=PCLK/8, N=6.
1001: Fsampling=PCLK/8, N=8.
1010: Fsampling=PCLK/16, N=5.
1011: Fsampling=PCLK/16, N=6.
1100: Fsampling=PCLK/16, N=8.
1101: Fsampling=PCLK/32, N=5.
1110: Fsampling=PCLK/32, N=6.
1111: Fsampling=PCLK/32, N=8.

Note: PCLK is 12 MHz when using the 24 MHz crystal oscillator, and 6 MHz using the 12 MHz RC oscillator.

Bits [11:10] TIM_IC2PSC: Input Capture 1 Prescaler. (Applies only if TIM_CC2S > 0)

00: No prescaling, capture each time an edge is detected on the capture input.

01: Capture once every 2 events.

10: Capture once every 4 events.

11: Capture once every 6 events.

Bits [9:8] TIM_CC2S: Capture / Compare 1 Selection

This configures the channel as an output or an input. If an input, it selects the input source.

00: Channel is an output.

01: Channel is an input and is mapped to TI2.

10: Channel is an input and is mapped to TI1.

11: Channel is an input and is mapped to TRGI. This mode requires an internal trigger input selected by the TIM_TS bit in the TIMx_SMCR register.

Note: TIM_CC2S may be written only when the channel is off (TIM_CC2E = 0 in the TIMx_CCER register).

Bits [6:4] TIM_OC1M: Output Compare 1 Mode. (Applies only if TIM_CC1S = 0)

See TIM_OC2M description above.

Bit 3 TIM_OC1BE: Output Compare 1 Buffer Enable. (Applies only if TIM_CC1S = 0)

See TIM_OC2BE description above.

Bit 2 TIM_OC1FE: Output Compare 1 Fast Enable. (Applies only if TIM_CC1S = 0)

See TIM_OC2FE description above.

Bits [7:4] TIM_IC1F: Input Capture 1 Filter. (Applies only if TIM_CC1S > 0)

See TIM_IC2F description above.

Bits [3:2] TIM_IC1PSC: Input Capture 1 Prescaler. (Applies only if TIM_CC1S > 0)

See TIM_IC2PSC description above.

Bits [1:0] TIM_CC1S: Capture / Compare 1 Selection

This configures the channel as an output or an input. If an input, it selects the input source.

00: Channel is an output.

01: Channel is an input and is mapped to TI1.

10: Channel is an input and is mapped to TI2.

11: Channel is an input and is mapped to TRGI. This requires an internal trigger input selected by the TIM_TS bit in the TIM_SMCR register.

Note: TIM_CC1S may be written only when the channel is off (TIM_CC1E = 0 in the TIMx_CCER register).

10.3.6 Timer x capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0xE01C (TIM1) and 0xF01C (TIM2)
Reset value: 0x0000 0000

Table 93. Timer x capture/compare mode register 2 (TIMx_CCMR2)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	TIM_OC4M			TIM_OC4BE	TIM_OC4FE	TIM_CC4S			TIM_OC3M			TIM_OC3BE	TIM_OC3FE	TIM_CC3S	
	TIM_IC4F			TIM_IC4PSC					TIM_IC3F			TIM_IC3PSC			

Timer channels can be programmed as inputs (capture mode) or outputs (compare mode). The direction of channel y is defined by TIM_CCyS in this register.

The other bits in this register have different functions in input and in output modes. The TIM_OC* fields only apply to a channel configured as an output (TIM_CCyS = 0), and the TIM_IC* fields only apply to a channel configured as an input (TIM_CCyS > 0).

Bits [14:12] TIM_OC4M: Output Compare 4 Mode. (Applies only if TIM_CC4S = 0

Define the behavior of the output reference signal OC4REF from which OC4 derives. OC4REF is active high whereas OC4's active level depends on the TIM_CC4P bit.

000: Frozen - The comparison between the output compare register TIMx_CCR4 and the counter TIMx_CNT has no effect on the outputs.

001: Set OC4REF to active on match. The OC4REF signal is forced high when the counter TIMx_CNT matches the capture/compare register 4 (TIMx_CCR4)

010: Set OC4REF to inactive on match. OC4REF signal is forced low when the counter TIMx_CNT matches the capture/compare register 4 (TIMx_CCR4).

011: Toggle - OC4REF toggles when TIMx_CNT = TIMx_CCR4.

100: Force OC4REF inactive.

101: Force OC4REF active.

110: PWM mode 1 - In up-counting, OC4REF is active as long as TIMx_CNT < TIMx_CCR4, otherwise OC4REF is inactive. In down-counting, OC4REF is inactive if TIMx_CNT > TIMx_CCR4, otherwise OC4REF is active.

111: PWM mode 2 - In up-counting, OC4REF is inactive if TIMx_CNT < TIMx_CCR4, otherwise OC4REF is active. In down-counting, OC4REF is active if TIMx_CNT > TIMx_CCR4, otherwise it is inactive.

Note: In PWM mode 1 or 2, the OC4REF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

Bit 11 TIM_OC4BE: Output Compare 4 Buffer Enable. (Applies only if TIM_CC4S = 0

0: Buffer register for TIMx_CCR4 is disabled. TIMx_CCR4 can be written at anytime, the new value is used by the shadow register immediately.

1: Buffer register for TIMx_CCR4 is enabled. Read/write operations access the buffer register. TIMx_CCR4 buffer value is loaded in the shadow register at each update event.

Note: The PWM mode can be used without enabling the buffer register only in one pulse mode (TIM_OPM bit set in the TIMx_CR2 register), otherwise the behavior is undefined.

Bit 10 TIM_OC4FE: Output Compare 4 Fast Enable. (Applies only if TIM_CC4S = 0)

This bit speeds the effect of an event on the trigger in input on the OC4 output.

0: OC4 behaves normally depending on the counter and TIM_CCR4 values even when the trigger is ON. The minimum delay to activate OC4 when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on the OC4 output. OC4 is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate OC4 output is reduced to 3 clock cycles. TIM_OC4FE acts only if the channel is configured in PWM 1 or PWM 2 mode.

Bits [15:12] TIM_IC4F: Input Capture 1 Filter. (Applies only if TIM_CC4S > 0)

This defines the frequency used to sample the TI4 input, $f_{Sampling}$, and the length of the digital filter applied to TI4. The digital filter requires N consecutive samples in the same state before being output.

0000: $f_{Sampling}$ = PCLK, no filtering.
0001: $f_{Sampling}$ = PCLK, N=2.
0010: $f_{Sampling}$ = PCLK, N=4.
0011: $f_{Sampling}$ = PCLK, N=8.
0100: $f_{Sampling}$ = PCLK/2, N=6.
0101: $f_{Sampling}$ = PCLK/2, N=8.
0110: $f_{Sampling}$ = PCLK/4, N=6.
0111: $f_{Sampling}$ = PCLK/4, N=8.
1000: $f_{Sampling}$ = PCLK/8, N=6.
1001: $f_{Sampling}$ = PCLK/8, N=8.
1010: $f_{Sampling}$ = PCLK/16, N=5.
1011: $f_{Sampling}$ = PCLK/16, N=6.
1100: $f_{Sampling}$ = PCLK/16, N=8.
1101: $f_{Sampling}$ = PCLK/32, N=5.
1110: $f_{Sampling}$ = PCLK/32, N=6.
1111: $f_{Sampling}$ = PCLK/32, N=8.

Note: PCLK is 12 MHz when using the 24 MHz crystal oscillator, and 6 MHz using the 12 MHz RC oscillator.

Bits [11:10] TIM_IC4PSC: Input Capture 1 Prescaler. (Applies only if TIM_CC4S > 0)

00: No prescaling, capture each time an edge is detected on the capture input.
01: Capture once every 2 events.
10: Capture once every 4 events.
11: Capture once every 6 events.

Bits [9:8] TIM_CC4S: Capture / Compare 1 Selection

This configures the channel as an output or an input. If an input, it selects the input source.

00: Channel is an output.
01: Channel is an input and is mapped to TI4.
10: Channel is an input and is mapped to TI3.
11: Channel is an input and is mapped to TRGI. This mode requires an internal trigger input selected by the TIM_TS bit in the TIMx_SMCR register.

Note: TIM_CC2S may be written only when the channel is off (TIM_CC2E = 0 in the TIMx_CCER register).

Bits [6:4] TIM_OC3M: Output Compare 1 Mode. (Applies only if TIM_CC3S = 0)

See TIM_OC4M description above.

Bit 3 TIM_OC3BE: Output Compare 3 Buffer Enable. (Applies only if TIM_CC3S = 0)

See TIM_OC4BE description above.

Bit 2 TIM_OC3FE: Output Compare 3 Fast Enable. (Applies only if TIM_CC3S = 0)

See TIM_OC4FE description above.

Bits [7:4] TIM_IC3F: Input Capture 1 Filter. (Applies only if TIM_CC3S > 0)

See TIM_IC4F description above.

Bits [3:2] TIM_IC3PSC: Input Capture 1 Prescaler. (Applies only if TIM_CC3S > 0)

See TIM_IC4PSC description above.

Bits [1:0] TIM_CC3S: Capture / Compare 3 Selection

This configures the channel as an output or an input. If an input, it selects the input source.

00: Channel is an output.

01: Channel is an input and is mapped to TI3.

10: Channel is an input and is mapped to TI4.

11: Channel is an input and is mapped to TRGI. This requires an internal trigger input selected by the TIM_TS bit in the TIM_SMCR register.

Note: TIM_CC3S may be written only when the channel is off (TIM_CC3E = 0 in the TIMx_CCER register).

10.3.7 Timer *x* capture/compare enable register (TIMx_CCER)

Address offset: 0xE020 (TIM1) and 0xF020 (TIM2)
Reset value: 0x0000 0000

Table 94. Timer *x* capture/compare enable register (TIMx_CCER)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TIM_CC4P	TIM_CC4P	Reserved		TIM_CC3P	TIM_CC3P	Reserved		TIM_CC2P	TIM_CC2P	Reserved		TIM_CC1P	TIM_CC1P
		rw	rw			rw	rw			rw	rw			rw	rw

Bit 13 TIM_CC4P: Capture/Compare 4 output Polarity

If CC4 is configured as an output channel:

0: OC4 is active high.

1: OC4 is active low.

If CC4 configured as an input channel:

0: IC4 is not inverted. Capture occurs on a rising edge of IC4. When used as an external trigger, IC4 is not inverted.

0: IC4 is inverted. Capture occurs on a falling edge of IC4. When used as an external trigger, IC4 is inverted.

1: Capture is enabled.

Bit 12 TIM_CC4E: Capture/Compare 4 output Enable

If CC4 is configured as an output channel:

0: OC4 is disabled.

1: OC4 is enabled.

If CC4 configured as an input channel:

0: Capture is disabled.

1: Capture is enabled.

Bit 9 TIM_CC3P

Refer to the CC4P description above.

Bit 8 TIM_CC3E

Refer to the CC4E description above.

Bit 5 TIM_CC2P
Refer to the CC4P description above.

Bit 4 TIM_CC2E
Refer to the CC43 description above.

Bit 1 TIM_CC1P
Refer to the CC4P description above.

Bit 0 TIM_CC1E
Refer to the CC4E description above.

10.3.8 Timer *x* counter register (TIMx_CNT)

Address offset: 0xE024 (TIM1) and 0xF024 (TIM2)
Reset value: 0x0000 0000

Table 95. Timer *x* counter register (TIMx_CNT)

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

Bits [15:0] TIM_CNT: Counter value

10.3.9 Timer *x* prescaler register (TIMx_PSC)

Address offset: 0xE028 (TIM1) and 0xF028 (TIM2)
Reset value: 0x0000 0000

Table 96. Timer *x* prescaler register (TIMx_PSC)

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

Bits [3:0] TIM_PSC: Prescaler value
The prescaler divides the internal timer clock frequency. The counter clock frequency CK_CNT is equal to fCK_PSC / (2 ^ TIM_PSC). Clock division factors can range from 1 through 32768. The division factor is loaded into the shadow prescaler register at each update event (including when the counter is cleared through TIM_UG bit of TMR1_EGR register or through the trigger controller when configured in reset mode).

10.3.10 Timer *x* auto-reload register (TIMx_ARR)

Address offset: 0xE02C (TIM1) and 0xF02C (TIM2)
Reset value: 0x0000 0000

Table 97. Timer *x* auto-reload register (TIMx_ARR)

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

Bits [15:0] TIM_ARR: Auto-reload value

TIM_ARR is the value to be loaded in the shadow auto-reload register.

The auto-reload register is buffered. Writing or reading the auto-reload register accesses the buffer register. The content of the buffer register is transfered in the shadow register permanently or at each update event UEV, depending on the auto-reload buffer enable bit (TIM_ARBE) in TMRx_CR1 register. The update event is sent when the counter reaches the overflow point (or underflow point when down-counting) and if the TIM_UDIS bit equals 0 in the TMRx_CR1 register. It can also be generated by software. The counter is blocked while the auto-reload value is 0.

10.3.11 Timer *x* capture/compare 1 register (TIMx_CCR1)

Address offset: 0xE034 (TIM1) and 0xF034 (TIM2)
Reset value: 0x0000 0000

Table 98. Timer *x* capture/compare 1 register (TIMx_CCR1)

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

Bits [15:0] TIM_CCR: Capture/compare value

If the CC1 channel is configured as an output (TIM_CC1S = 0):

TIM_CCR1 is the buffer value to be loaded in the actual capture/compare 1 register. It is loaded permanently if the preload feature is not selected in the TMR1_CCMR1 register (bit OC1PE). Otherwise the buffer value is copied to the shadow capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TMR1_CNT and signaled on the OC1 output.

If the CC1 channel is configured as an input (TIM_CC1S is not 0):

CCR1 is the counter value transferred by the last input capture 1 event (IC1).

10.3.12 Timer *x* capture/compare 2 register (TIMx_CCR2)

Address offset: 0xE038 (TIM1) and 0xF038 (TIM2)
Reset value: 0x0000 0000

Table 99. Timer *x* capture/compare 2 register (TIMx_CCR2)

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

Bits [15:0] See description in the TIMx_CCR1 register.

10.3.13 Timer *x* capture/compare 3 register (TIMx_CCR3)

Address offset: 0xE03C (TIM1) and 0xF03C (TIM2)
Reset value: 0x0000 0000

Table 100. Timer *x* capture/compare 3 register (TIMx_CCR3)

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

Bits [15:0] See description in the TIMx_CCR1 register.

10.3.14 Timer *x* capture/compare 4 register (TIMx_CCR4)

Address offset: 0xE040 (TIM1) and 0xF040 (TIM2)
Reset value: 0x0000 0000

Table 101. Timer *x* capture/compare 4 register (TIMx_CCR4)

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

Bits [15:0] See description in the TIMx_CCR1 register.

10.3.15 Timer 1 option register (TIM1_OR)

Address offset: 0xE050
Reset value: 0x0000 0000

Table 102. Timer 1 option register (TIM1_OR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												TIM_ORRSVD	TIM_CLKMSKEN	TIM1_EXTRIGSEL	
												rw	rw	rw	

Bit 3 TIM_ORRSVD
Reserved: this bit must always be set to 0.

Bit 2 TIM_CLKMSKEN
Enables TIM1MSK when TIM1CLK is selected as the external trigger: 0 = TIM1MSK not used, 1 = TIM1CLK is ANDed with the TIM1MSK input.

Bits [1:0] TIM1_EXTRIGSEL
Selects the external trigger used in external clock mode 2: 0 = PCLK, 1 = calibrated 1 kHz clock, 2 = 32 kHz reference clock (if available), 3 = TIM1CLK pin.

10.3.16 Timer 2 option register (TIM2_OR)

Address offset: 0xF050
Reset value: 0x0000 0000

Table 103. Timer 2 option register (TIM2_OR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TIM_REMAPC4	TIM_REMAPC3	TIM_REMAPC2	TIM_REMAPC1	TIM_ORRSVD	TIM_CLKMSKEN	TIM1_EXTRIGSEL	
								rw	rw	rw	rw	rw	rw	rw	

Bit 7 TIM_REMAPC4
Selects the GPIO used for TIM2_CH4: 0 = PA2, 1 = PB4.

Bit 6 TIM_REMAPC3
Selects the GPIO used for TIM2_CH3: 0 = PA1, 1 = PB3.

Bit 5 TIM_REMAPC2
Selects the GPIO used for TIM2_CH2: 0 = PA3, 1 = PB2.

Bit 4 TIM_REMAPC1
Selects the GPIO used for TIM2_CH1: 0 = PA0, 1 = PB1.

Bit 3 TIM_ORRSVD

Reserved: this bit must always be set to 0.

Bit 2 TIM_CLKMSKEN

Enables TIM2MSK when TIM2CLK is selected as the external trigger: 0 = TIM2MSK not used, 1 = TIM2CLK is ANDed with the TIM2MSK input.

Bits [1:0] TIM1_EXTRIGSEL

Selects the external trigger used in external clock mode 2: 0 = PCLK, 1 = calibrated 1 kHz clock, 2 = 32 kHz reference clock (if available), 3 = TIM2CLK pin.

10.3.17 Timer *x* interrupt configuration register (INT_TIMxCFG)

Address offset: 0xA840 (TIM1) and 0xA844 (TIM2)
Reset value: 0x0000 0000

Table 104. Timer *x* interrupt configuration register (INT_TIMxCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved									INT_TIMTIF	Reserved	INT_TIMCC4IF	INT_TIMCC3IF	INT_TIMCC2IF	INT_TIMCC1IF	INT_TIMUIF
									rw		rw	rw	rw	rw	rw

Bit 6 INT_TIMTIF: Trigger interrupt enable.

Bit 4 INT_TIMCC4IF: Capture or compare 4 interrupt enable.

Bit 3 INT_TIMCC3IF: Capture or compare 3 interrupt enable.

Bit 2 INT_TIMCC2IF: Capture or compare 2 interrupt enable.

Bit 1 INT_TIMCC1IF: Capture or compare 1 interrupt enable.

Bit 0 INT_TIMUIF: Update interrupt enable.

10.3.18 Timer *x* interrupt flag register (INT_TIMxFLAG)

Address offset: 0xA800 (TIM1) and 0xA804 (TIM2)
Reset value: 0x0000 0000

Table 105. Timer *x* interrupt flag register (INT_TIMxFLAG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			INT_TIMRSVD				Reserved		INT_TIMTIF	Reserved	INT_TIMCC4IF	INT_TIMCC3IF	INT_TIMCC2IF	INT_TIMCC1IF	INT_TIMUIF
			r						rw		rw	rw	rw	rw	rw

Bits [12:9] INT_TIMRSVD: May change during normal operation.

Bit 6 INT_TIMTIF: Trigger interrupt.

Bit 4 INT_TIMCC4IF: Capture or compare 4 interrupt pending.

Bit 3 INT_TIMCC3IF: Capture or compare 3 interrupt pending.

Bit 2 INT_TIMCC2IF: Capture or compare 2 interrupt pending.

Bit 1 INT_TIMCC1IF: Capture or compare 1 interrupt pending.

Bit 0 INT_TIMUIF: Update interrupt pending.

10.3.19 Timer *x* missed interrupt register (INT_TIMxMISS)

Address offset: 0xA818 (TIM1) and 0xA81C (TIM2)
Reset value: 0x0000 0000

Table 106. Timer *x* missed interrupt register (INT_TIMxMISS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			INT_TIMMISSCC4IF	INT_TIMMISSCC3IF	INT_TIMMISSCC2IF	INT_TIMMISSCC1IF	Reserved		INT_TIMMISSRSVD						
			rw	rw	rw	rw			r						

Bit 12 INT_TIMMISSCC4IF: Capture or compare 4 interrupt missed.

Bit 11 INT_TIMMISSCC3IF: Capture or compare 3 interrupt missed.

Bit 10 INT_TIMMISSCC2IF: Capture or compare 2 interrupt missed.

Bit 9 INT_TIMMISSCC1IF: Capture or compare 1 interrupt missed.

Bits [6:0] INT_TIMMISSRSVD: May change during normal operation.

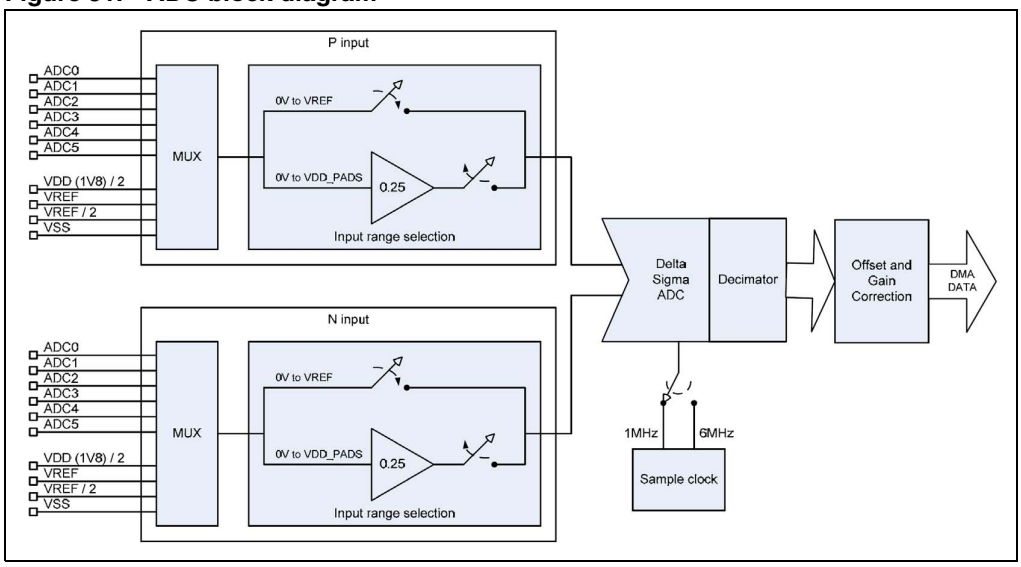
11 Analog-to-digital converter

The STM32W108 analog-to-digital converter (ADC) is a first-order sigma-delta converter with the following features:

- Resolution of up to 12 bits
- Sample times as fast as 5.33 µs (188 kHz)
- Differential and single-ended conversions from six external and four internal sources
- Two voltage ranges (differential): -VREF to +VREF, and –VDD_PADS to +VDD_PADS
- Choice of internal or external VREF: internal VREF may be output
- Digital offset and gain correction
- Dedicated DMA channel with one-shot and continuous operating modes

Figure 51 shows the basic ADC structure.

Figure 51. ADC block diagram



While the ADC Module supports both single-ended and differential inputs, the ADC input stage always operates in differential mode. Single-ended conversions are performed by connecting one of the differential inputs to VREF/2 while fully differential operation uses two external inputs.

Note: This note only applies to STM32W108xB. In high voltage mode, input buffers (with 0.25 gain only) may experience long term drift of its input offset voltage that affects ADC accuracy. In these cases, only the 1.2V input range mode of the ADC should be used. If measurement of signals >1.2V is required, then external attenuation should be added.

Note: For STM32W108CC and STM32108CZ, high voltage mode is supported only in single mode. It is also required by the software to periodically set to 1 bit ADC_HVSELN in the register ADC_CFG.

11.1 Functional description

11.1.1 Setup and configuration

To use the ADC follow this procedure, described in more detail in the next sections:

- Configure any GPIO pins to be used by the ADC in analog mode.
- Configure the voltage reference (internal or external).
- Set the offset and gain values.
- Reset the ADC DMA, define the DMA buffer, and start the DMA in the proper transfer mode.
- If interrupts will be used, configure the primary ADC interrupt and specific mask bits.
- Write the ADC configuration register to define the inputs, voltage range, sample time, and start the conversions.

11.1.2 GPIO usage

A GPIO pin used by the ADC as an input or voltage reference must be configured in analog mode by writing 0 to its 4-bit field in the proper GPIO_PnCFGH/L register. Note that a GPIO pin in analog mode cannot be used for any digital functions, and software always reads it as 1.

Table 107. ADC GPIO pin usage

Analog Signal	GPIO	Configuration control
ADC0 input	PB5	GPIO_PBCFGH[7:4]
ADC1 input	PB6	GPIO_PBCFGH[11:8]
ADC2 input	PB7	GPIO_PBCFGH[15:12]
ADC3 input	PC1	GPIO_PCCFGH[7:4]
ADC4 input	PA4	GPIO_PACFGH[3:0]
ADC5 input	PA5	GPIO_PACFGH[7:4]
VREF input or output	PB0	GPIO_PBCFGH[3:0]

See *Section 8: General-purpose input/outputs on page 58* for more information about how to configure the GPIO pins.

11.1.3 Voltage reference

The ADC voltage reference (VREF), may be internally generated or externally sourced from PB0. If internally generated, it may optionally be output on PB0.

To use an external reference, an ST system function must be called after reset and after waking from deep sleep. PB0 must also be configured in analog mode using GPIO_PBCFGH[3:0]. See the STM32W108 HAL documentation for more information on the system functions required to use an external reference.

11.1.4 Offset/gain correction

When a conversion is complete, the 16-bit converted data is processed by offset/gain correction logic:

- The basic ADC conversion result is added to the 16-bit signed (two's complement) value of the ADC offset register (ADC_OFFSET).
- The offset-corrected data is multiplied by the 16-bit ADC gain register, ADC_GAIN, to produce a 16-bit signed result. If the product is greater than 0x7FFF (32767), or less than 0x8000 (-32768), it is limited to that value and the INT_ADCSAT bit is set in the INT_ADCFLAG register.

ADC_GAIN is an unsigned scaled 16-bit value: ADC_GAIN[15] is the integer part of the gain factor and ADC_GAIN[14:0] is the fractional part. As a result, ADC_GAIN values can represent gain factors from 0 through $(2 - 2^{-15})$.

Reset initializes the offset to zero (ADC_OFFSET = 0) and gain factor to one (ADC_GAIN = 0x8000).

11.1.5 DMA

The ADC DMA channel writes converted data, which incorporates the offset/gain correction, into a DMA buffer in RAM.

The ADC DMA buffer is defined by two registers:

- ADC_DMABEG is the start address of the buffer and must be even.
- ADC_DMASIZE specifies the size of the buffer in 16-bit samples, or half its length in bytes.

To prepare the DMA channel for operation, reset it by writing the ADC_DMARST bit in the ADC_DMACFG register, then start the DMA in either linear or auto wrap mode by setting the ADC_DMALOAD bit in the ADC_DMACFG register. The ADC_DMAAUTOWRAP bit in the ADC_DMACFG register selects the DMA mode: 0 for linear mode, 1 for auto wrap mode.

- In linear mode the DMA writes to the buffer until the number of samples given by ADC_DMASIZE has been output. Then the DMA stops and sets the INT_ADCULDFULL bit in the INT_ADCFLAG register. If another ADC conversion completes before the DMA is reset or the ADC is disabled, the INT_ADCOVF bit in the INT_ADCFLAG register is set.
- In auto wrap mode the DMA writes to the buffer until it reaches the end, then resets its pointer to the start of the buffer and continues writing samples. The DMA transfers continue until the ADC is disabled or the DMA is reset.

When the DMA fills the lower and upper halves of the buffer, it sets the INT_ADCULDHALF and INT_ADCULDFULL bits, respectively, in the INT_ADCFLAG register. The current location to which the DMA is writing can also be determined by reading the ADC_DMACUR register.

11.1.6 ADC configuration register

The ADC configuration register (ADC_CFG) sets up most of the ADC operating parameters.

Input

The analog input of the ADC can be chosen from various sources. The analog input is configured with the ADC_MUXP and ADC_MUXN bits within the ADC_CFG register. *Table 108* shows the possible input selections.

Table 108. ADC inputs

ADC_MUXn (1)	Analog source at ADC	GPIO pin	Purpose
0	ADC0	PB5	
1	ADC1	PB6	
2	ADC2	PB7	
3	ADC3	PC0	
4	ADC4	PA4	
5	ADC5	PA5	
6	No connection		
7	No connection		
8	GND	Internal connection	Calibration
9	VREF/2	Internal connection	Calibration
10	VREF	Internal connection	Calibration
11	1V8 VREG/2	Internal connection	Supply monitoring and calibration
12	No connection		
13	No connection		
14	No connection		
15	No connection		

1. Denotes bits ADC_MUXP or ADC_MUXN in register ADC_CFG.

Table 109 shows the typical configurations of ADC inputs.

Table 109. Typical ADC input configurations

ADC P input	ADC N input	ADC_MUXP	ADC_MUXN	Purpose
ADC0	VREF/2	0	9	Single-ended
ADC1	VREF/2	1	9	Single-ended
ADC2	VREF/2	2	9	Single-ended
ADC3	VREF/2	3	9	Single-ended
ADC4	VREF/2	4	9	Single-ended
ADC5	VREF/2	5	9	Single-ended
ADC1	ADC0	1	0	Differential
ADC3	ADC2	3	2	Differential

Table 109. Typical ADC input configurations (continued)

ADC P input	ADC N input	ADC_MUXP	ADC_MUXN	Purpose
ADC5	ADC4	5	4	Differential
GND	VREF/2	8	9	Calibration
VREF	VREF/2	10	9	Calibration
VDD_PADSA/2	VREF/2	11	9	Calibration

Input range

ADC inputs can be routed through input buffers to expand the input voltage range. The input buffers have a fixed 0.25 gain and the converted data is scaled by that factor.

With the input buffers disabled the single-ended input range is 0 to VREF and the differential input range is -VREF to +VREF. With the input buffers enabled the single-ended range is 0 to VDD_PADS and the differential range is -VDD_PADS to +VDD_PADS.

The input buffers are enabled for the ADC P and N inputs by setting the ADC_HVSELP and ADC_HVSELN bits respectively, in the ADC_CFG register. The ADC accuracy is reduced when the input buffer is selected.

Sample time

ADC sample time is programmed by selecting the sampling clock and the clocks per sample.

- The sampling clock may be either 1 MHz or 6 MHz. If the ADC_1MHZCLK bit in the ADC_CFG register is clear, the 6 MHz clock is used; if it is set, the 1 MHz clock is selected. The 6 MHz sample clock offers faster conversion times but the ADC resolution is lower than that achieved with the 1 MHz clock.
- The number of clocks per sample is determined by the ADC_PERIOD bits in the ADC_CFG register. ADC_PERIOD values select from 32 to 4096 sampling clocks in powers of two. Longer sample times produce more significant bits. Regardless of the sample time, converted samples are always 16-bits in size with the significant bits left-aligned within the value.

Table 110 shows the options for ADC sample times and the significant bits in the conversion results.

Table 110. ADC sample times

ADC_PERIOD	Sample clocks	Sample time (µs)		Sample frequency (kHz)		Significant bits
		1 MHz clock	6 MHz clock	1 MHz clock	6 MHz clock	
0	32	32	5.33	31.3	188	5
1	64	64	10.7	15.6	93.8	6
2	128	128	21.3	7.81	46.9	7
3	256	256	42.7	3.91	23.4	8
4	512	512	85.3	1.95	11.7	9
5	1024	1024	170	0.977	5.86	10

Table 110. ADC sample times (continued)

ADC_PERIOD	Sample clocks	Sample time (µs)		Sample frequency (kHz)		Significant bits
		1 MHz clock	6 MHz clock	1 MHz clock	6 MHz clock	
6	2048	2048	341	0.488	2.93	11
7	4096	4096	682	0.244	1.47	12

Note: ADC sample timing is the same whether the STM32W108 is using the 24 MHz crystal oscillator or the 12 MHz high-speed RC oscillator. This facilitates using the ADC soon after the CPU wakes from deep sleep, before switching to the crystal oscillator.

11.1.7 Operation

Setting the ADC_EN bit in the ADC_CFG register enables the ADC; once enabled, it performs conversions continuously until it is disabled. If the ADC had previously been disabled, a 21 µs analog startup delay is imposed before the ADC starts conversions. The delay timing is performed in hardware and is simply added to the time until the first conversion result is output.

When the ADC is first enabled, and or if any change is made to ADC_CFG after it is enabled, the time until a result is output is double the normal sample time. This is because the ADC's internal design requires it to discard the first conversion after startup or a configuration change. This is done automatically and is hidden from software except for the longer timing. Switching the processor clock between the RC and crystal oscillator also causes the ADC to go through this startup cycle. If the ADC was newly enabled, the analog delay time is added to the doubled sample time.

If the DMA is running when ADC_CFG is modified, the DMA does not stop, so the DMA buffer may contain conversion results from both the old and new configurations.

The following procedure illustrates a simple polled method of using the ADC. After completing the procedure, the latest conversion results is available in the location written to by the DMA. This assumes that any GPIOs and the voltage reference have already been configured.

1. Allocate a 16-bit signed variable, for example analogData, to receive the ADC output. (Make sure that analogData is half-word aligned – that is, at an even address.)
2. Disable all ADC interrupts – write 0 to INT_ADCCFG.
3. Set up the DMA to output conversion results to the variable, analogData.
   Reset the DMA – set the ADC_DMARST bit in ADC_DMACFG.
   Define a one sample buffer – write analogData's address to ADC_DMABEG, set ADC_DMASIZE to 1.
4. Write the desired offset and gain correction values to the ADC_OFFSET and ADC_GAIN registers.
5. Start the ADC and the DMA.
   Write the desired conversion configuration, with the ADC_EN bit set, to ADC_CFG.
   Clear the ADC buffer full flag – write INT_ADCULDFULL to INT_ADCFLAG.
   Start the DMA in auto wrap mode – set the ADC_DMAAUTOWRAP and ADC_DMALOAD bits in ADC_DMACFG.
6. Wait until the INT_ADCULDFULL bit is set in INT_ADCFLAG, then read the result from analogData.

To convert multiple inputs using this approach, repeat Steps 4 through 6, loading the desired input configurations to ADC_CFG in Step 5. If the inputs can use the same offset/gain correction, just repeat Steps 5 and 6.

11.1.8 Calibration

Sampling of internal connections GND, VREF/2, and VREF allow for offset and gain calibration of the ADC in applications where absolute accuracy is important. Offset error is calculated from the minimum input and gain error is calculated from the full scale input range. Correction using VREF is recommended because VREF is calibrated by the ST software against VDD_PADSA. The VDD_PADSA regulator is factory-trimmed to 1.80 V ± 20 mV. If better absolute accuracy is required, the ADC can be configured to use an external reference. The ADC calibrates as a single-ended measurement. Differential signals require correction of both their inputs.

Table 111 shows the equations used to calculate the gain and offset correction values.

Table 111. ADC gain and offset correction equations

Calibration	Correction value
Gain, buffer disabled	$0x8000 \times \frac{(N_{VREF} - N_{GND})}{0x4000}$
Gain, buffer enabled	$0x8000 \times \frac{(N_{VREF} - N_{VREF/2})}{0x2000} \times \frac{1}{4}$
Offset, buffer disabled (after applying gain correction)	$\frac{1}{2} \times (N_{GND} - 0xE000)$
Offset, buffer enabled (after applying gain correction)	$\frac{1}{2} \times (N_{VREF/2} - 0xE800)$

Equation notes

- All N are 16-bit two's complement numbers.
- N_{GND} is a sampling of ground. Due to the ADC's internal design, VGND does not yield the minimum two's complement value 0x8000 as the conversion result. Instead, VGND yields a two's complement value close to 0xE000 when the input buffer is not selected. VGND cannot be measured when the input buffer is enabled because it is outside the buffer's input range.
- N_{VREF} is a sampling of VREF. Due to the ADC's internal design, VREF does not yield the maximum positive two's complement 0x7FFF as the conversion result. Instead, VREF yields a two's complement value close to 0x2000 when the input buffer is not selected and yields a two's complement value close to 0xF000 when the input buffer is selected.
- $N_{VREF/2}$ is a sampling of VREF/2. VREF/2 yields a two's complement value close to 0x0000 when the input buffer is not selected, and yields a two's complement value close to 0xE800 when the input buffer is selected.
- Offset correction is affected by the gain correction value. Offset correction is calculated after gain correction has been applied.

11.2 Interrupts

The ADC has its own ARM® Cortex-M3 vectored interrupt with programmable priority. The ADC interrupt is enabled by writing the INT_ADC bit to the INT_CFGSET register, and cleared by writing the INT_ADC bit to the INT_CFGCLR register. *Section 12: Interrupts on page 177* describes the interrupt system in detail.

Four kinds of ADC events can generate an ADC interrupt, and each has a bit flag in the INT_ADCFLAG register to identify the reason(s) for the interrupt:

- INT_ADCOVF – an ADC conversion result was ready but the DMA was disabled (DMA buffer overflow).
- INT_ADCSAT– the gain correction multiplication exceeded the limits for a signed 16-bit number (gain saturation).
- INT_ADCULDFULL – the DMA wrote to the last location in the buffer (DMA buffer full).
- INT_ADCULDHALF – the DMA wrote to the last location of the first half of the DMA buffer (DMA buffer half full).

Bits in INT_ADCFLAG may be cleared by writing a 1 to their position.

The INT_ADCCFG register controls whether or not INT_ADCFLAG bits actually request the ARM® Cortex-M3 ADC interrupt; only the events whose bits are 1 in INT_ADCCFG can do so.

For non-interrupt (polled) ADC operation set INT_ADCCFG to zero, and read the bit flags in INT_ADCFLAG to determine the ADC status.

Note: When making changes to the ADC configuration it is best to disable the DMA beforehand. If this isn't done it can be difficult to determine at which point the sample data in the DMA buffer switch from the old configuration to the new configuration. However, since the ADC will be left running, if it completes a conversion after the DMA is disabled, the INT_ADCOVF flag will be set. To prevent these unwanted DMA buffer overflow indications, clear the INT_ADCOVF flag immediately after enabling the DMA, preferably with interrupts off. Disabling the ADC in addition to the DMA is often undesirable because of the additional analog startup time when it is re-enabled.

11.3 Analog-to-digital converter (ADC) registers

11.3.1 ADC configuration register (ADC_CFG)

Address offset: 0xD004
Reset value: 0x0000 1800

Table 112. ADC configuration register (ADC_CFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC_PERIOD			ADC_HVSELP	ADC_HVSELN	ADC_MUXP			ADC_MUXP	ADC_MUXN				ADC_1MHZCLK	ST Reserved	ADC_ENABLE
rw			rw	rw	rw			rw	rw				rw	rw	rw

Bits [15:13] ADC_PERIOD: ADC sample time in clocks and the equivalent significant bits in the conversion.
0: 32 clocks (5 bits).4: 512 clocks (9 bits).
1: 64 clocks (6 bits).5: 1024 clocks (10 bits).
2: 128 clocks (7 bits).6: 2048 clocks (11 bits).
3: 256 clocks (8 bits).7: 4096 clocks (12 bits).

Bit 12 ADC_HVSELP: Select voltage range for the P input channel.
0: Low voltage range (input buffer disabled).
1: High voltage range (input buffer enabled).

Bit 11 ADC_HVSELN: Select voltage range for the N input channel.
0: Low voltage range (input buffer disabled).
1: High voltage range (input buffer enabled).

Bits [10:7] ADC_MUXP: Input selection for the P channel.
0x0: PB5 pin.
0x1: PB6 pin.
0x2: PB7 pin.
0x3: PC1 pin.
0x4: PA4 pin.
0x5: PA5 pin.
0x8: GND (0V) (not for high voltage range).
0x9: VREF/2 (0.6V).
0xA: VREF (1.2V).
0xB: VREG/2 (0.9V) (not for high voltage range).
0x6, 0x7, 0xC-0xF: Reserved.

Bits [6:3] ADC_MUXN: Input selection for the N channel.
Refer to ADC_MUXP above for choices.

Bit 2 ADC_1MHZCLK: Select ADC clock:
0: 6 MHz1: 1 MHz.

Bit 1 Reserved: this bit must always be set to 0.

Bit 0 ADC_ENABLE: Enable the ADC: write 1 to enable continuous conversions, write 0 to stop.
When the ADC is started the first conversion takes twice the usual number of clocks plus 21 microseconds. If anything in this register is modified while the ADC is running, the next conversion takes twice the usual number of clocks.

11.3.2 ADC offset register (ADC_OFFSET)

Address offset: 0xD008
Reset value: 0x0000 0000

Table 113. ADC offset register (ADC_OFFSET)

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

Bits [15:0] ADC_OFFSET_FIELD: 16-bit signed offset added to the basic ADC conversion result before gain correction is applied.

11.3.3 ADC gain register (ADC_GAIN)

Address offset: 0xD00C
Reset value: 0x0000 8000

Table 114. ADC gain register (ADC_GAIN)

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

Bits [15:0] ADC_GAIN_FIELD: Gain factor that is multiplied by the offset-corrected ADC result to produce the output value. The gain is a 16-bit unsigned scaled integer value with a binary decimal point between bits 15 and 14. It can represent values from 0 to (almost) 2. The reset value is a gain factor of 1.

11.3.4 ADC DMA configuration register (ADC_DMACFG)

Address offset: 0xD010
Reset value: 0x0000 0000

Table 115. ADC DMA configuration register (ADC_DMACFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											ADC_DMARST	Reserved		ADC_DMAAUTOWRAP	ADC_DMALOAD
											w			rw	rw

Bit 4 ADC_DMARST: Write 1 to reset the ADC DMA. This bit auto-clears.

Bit 1 ADC_DMAAUTOWRAP: Selects DMA mode.
0: Linear mode, the DMA stops when the buffer is full.
1: Auto-wrap mode, the DMA output wraps back to the start when the buffer is full.

Bit 0 ADC_DMALOAD: Loads the DMA buffer.
Write 1 to start DMA (writing 0 has no effect). Cleared when DMA starts or is reset.

11.3.5 ADC DMA status register (ADC_DMASTAT)

Address offset: 0xD014
Reset value: 0x0000 0000

Table 116. ADC DMA status register (ADC_DMASTAT)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														ADC_DMAOVF	ADC_DMAACT
														r	r

Bit 1 ADC_DMAOVF: DMA overflow: occurs when an ADC result is ready and the DMA is not active. Cleared by DMA reset.

Bit 0 ADC_DMAACT: DMA status: reads 1 if DMA is active.

11.3.6 ADC DMA begin address register (ADC_DMABEG)

Address offset: 0xD018
Reset value: 0x2000 0000

Table 117. ADC DMA begin address register (ADC_DMABEG)

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

Bits [13:0] ADC_DMABEG: ADC buffer start address.
Caution: This must be an even address - the least significant bit of this register is fixed at zero by hardware.

11.3.7 ADC DMA buffer size register (ADC_DMASIZE)

Address offset: 0xD01C
Reset value: 0x0000 0000

Table 118. ADC DMA buffer size register (ADC_DMASIZE)

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

Bits [12:0] ADC_DMASIZE_FIELD: ADC buffer size. This is the number of 16-bit ADC conversion results the buffer can hold, not its length in bytes. (The length in bytes is twice this value.)

11.3.8 ADC DMA current address register (ADC_DMACUR)

Address offset: 0xD020
Reset value: 0x2000 0000

Table 119. ADC DMA current address register (ADC_DMACUR)

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

Bits [13:1] ADC_DMACUR_FIELD: Current DMA address: the location that will be written next by the DMA.

11.3.9 ADC DMA count register (ADC_DMACNT)

Address offset: 0xD024
Reset value: 0x0000 0000

Table 120. ADC DMA count register (ADC_DMACNT)

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

Bits [12:0] ADC_DMACNT_FIELD: DMA count: the number of 16-bit conversion results that have been written to the buffer.

11.3.10 ADC interrupt flag register (INT_ADCFLAG)

Address offset: 0xA810
Reset value: 0x0000 0000

Table 121. ADC interrupt flag register (INT_ADCFLAG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											INT_ADCOVF	INT_ADCSAT	INT_ADCULDFULL	INT_ADCULDHALF	ST Reserved
											rw	rw	rw	rw	rw

Bit 4 INT_ADCOVF: DMA buffer overflow interrupt pending.

Bit 3 INT_ADCSAT: Gain correction saturation interrupt pending.

Bit 2 INT_ADCULDFULL: DMA buffer full interrupt pending.

Bit 1 INT_ADCULDHALF: DMA buffer half full interrupt pending.

Bit 0 Reserved: this bit should always be set to 1.

11.3.11 ADC interrupt configuration register (INT_ADCCFG)

Address offset: 0xA850
Reset value: 0x0000 0000

Table 122. ADC interrupt configuration register (INT_ADCCFG)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											INT_ADCOVF	INT_ADCSAT	INT_ADCULDFULL	INT_ADCULDHALF	ST Reserved
											rw	rw	rw	rw	

Bit 4 INT_ADCOVF: DMA buffer overflow interrupt enable.

Bit 3 INT_ADCSAT: Gain correction saturation interrupt enable.

Bit 2 INT_ADCULDFULL: DMA buffer full interrupt enable.

Bit 1 INT_ADCULDHALF: DMA buffer half full interrupt enable.

Bit 0 ST Reserved: this bit must always be set to 0.

12 Interrupts

The STM32W108's interrupt system is composed of two parts: a standard ARM® Cortex-M3 Nested Vectored Interrupt Controller (NVIC) that provides top level interrupts, and an Event Manager (EM) that provides second level interrupts. The NVIC and EM provide a simple hierarchy. All second level interrupts from the EM feed into top level interrupts in the NVIC. This two-level hierarchy allows for both fine granular control of interrupt sources and coarse granular control over entire peripherals, while allowing peripherals to have their own interrupt vector.

The *Section 12.3: Nested vectored interrupt controller (NVIC) interrupts* provides a description of the NVIC and an overview of the exception table (ARM nomenclature refers to interrupts as exceptions) and *Section 12.2: Event manager* provides a more detailed description of the Event Manager including a table of all top-level peripheral interrupts and their second-level interrupt sources.

In practice, top-level peripheral interrupts are only used to enable or disable interrupts for an entire peripheral. Second-level interrupts originate from hardware sources, and therefore are the main focus of applications using interrupts.

12.1 Nested vectored interrupt controller (NVIC)

The ARM® Cortex-M3 Nested Vectored Interrupt Controller (NVIC) facilitates low-latency exception and interrupt handling. The NVIC and the processor core interface are closely coupled, which enables low-latency interrupt processing and efficient processing of late arriving interrupts. The NVIC also maintains knowledge of the stacked (nested) interrupts to enable tail-chaining of interrupts.

The ARM® Cortex-M3 NVIC contains 10 standard interrupts that are related to chip and CPU operation and management. In addition to the 10 standard interrupts, it contains 17 individually vectored peripheral interrupts specific to the STM32W108.

The NVIC defines a list of exceptions. These exceptions include not only traditional peripheral interrupts, but also more specialized events such as faults and CPU reset. In the ARM® Cortex-M3 NVIC, a CPU reset event is considered an exception of the highest priority, and the stack pointer is loaded from the first position in the NVIC exception table. The NVIC exception table defines all exceptions and their position, including peripheral interrupts. The position of each exception is important since it directly translates to the location of a 32-bit interrupt vector for each interrupt, and defines the hardware priority of exceptions. Each exception in the table is a 32-bit address that is loaded into the program counter when that exception occurs. *Table 123* lists the entire exception table. Exceptions 0 (stack pointer) through 15 (SysTick) are part of the standard ARM® Cortex-M3 NVIC, while exceptions 16 (Timer 1) through 32 (Debug) are the peripheral interrupts specific to the STM32W108 peripherals. The peripheral interrupts are listed in greater detail in *Table 124*.

Table 123. NVIC exception table

Exception	Position	Description
-	0	Stack top is loaded from first entry of vector table on reset.
Reset	1	Invoked on power up and warm reset. On first instruction, drops to lowest priority (Thread mode). Asynchronous.

Table 123. NVIC exception table (continued)

Exception	Position	Description
NMI	2	Cannot be stopped or preempted by any exception but reset. Asynchronous.
Hard Fault	3	All classes of fault, when the fault cannot activate because of priority or the Configurable Fault handler has been disabled. Synchronous.
Memory Fault	4	MPU mismatch, including access violation and no match. Synchronous.
Bus Fault	5	Pre-fetch, memory access, and other address/memory-related faults. Synchronous when precise and asynchronous when imprecise.
Usage Fault	6	Usage fault, such as 'undefined instruction executed' or 'illegal state transition attempt'. Synchronous.
-	7-10	Reserved.
SVCall	11	System service call with SVC instruction. Synchronous.
Debug Monitor	12	Debug monitor, when not halting. Synchronous, but only active when enabled. It does not activate if lower priority than the current activation.
-	13	Reserved.
PendSV	14	Pendable request for system service. Asynchronous and only pended by software.
SysTick	15	System tick timer has fired. Asynchronous.
Timer 1	16	Timer 1 peripheral interrupt.
Timer 2	17	Timer 2 peripheral interrupt.
Management	18	Management peripheral interrupt.
Baseband	19	Baseband peripheral interrupt.
Sleep Timer	20	Sleep Timer peripheral interrupt.
Serial Controller 1	21	Serial Controller 1 peripheral interrupt.
Serial Controller 2	22	Serial Controller 2 peripheral interrupt.
Security	23	Security peripheral interrupt.
MAC Timer	24	MAC Timer peripheral interrupt.
MAC Transmit	25	MAC Transmit peripheral interrupt.
MAC Receive	26	MAC Receive peripheral interrupt.
ADC	27	ADC peripheral interrupt.
IRQA	28	IRQA peripheral interrupt.
IRQB	29	IRQB peripheral interrupt.
IRQC	30	IRQC peripheral interrupt.
IRQD	31	IRQD peripheral interrupt.
Debug	32	Debug peripheral interrupt.

The NVIC also contains a software-configurable interrupt prioritization mechanism. The Reset, NMI, and Hard Fault exceptions, in that order, are always the highest priority, and are not software-configurable. All other exceptions can be assigned a 5-bit priority number, with low values representing higher priority. If any exceptions have the same software-configurable priority, then the NVIC uses the hardware-defined priority. The hardware-defined priority number is the same as the position of the exception in the exception table. For example, if IRQA and IRQB both fire at the same time and have the same software-defined priority, the NVIC handles IRQA, with priority number 28, first because it has a higher hardware priority than IRQB with priority number 29.

The top level interrupts are controlled through five ARM® Cortex-M3 NVIC registers: INT_CFGSET, INT_CFGCLR, INT_PENDSET, INT_PENDCLR, and INT_ACTIVE. Writing 0 into any bit in any of these five register is ineffectual.

- INT_CFGSET - Writing 1 to a bit in INT_CFGSET enables that top level interrupt.
- INT_CFGCLR - Writing 1 to a bit in INT_CFGCLR disables that top level interrupt.
- INT_PENDSET - Writing 1 to a bit in INT_PENDSET triggers that top level interrupt.
- INT_PENDCLR - Writing 1 to a bit in INT_PENDCLR clear that top level interrupt.
- INT_ACTIVE cannot be written to and is used for indicating which interrupts are currently active.

INT_PENDSET and INT_PENDCLR set and clear a simple latch; INT_CFGSET and INT_CFGCLR set and clear a mask on the output of the latch. Interrupts may be pended and cleared at any time, but any pended interrupt will not be taken unless the corresponding mask (INT_CFGSET) is set, which allows that interrupt to propagate. If an INT_CFGSET bit is set and the corresponding INT_PENDSET bit is set, then the interrupt will propagate and be taken. If INT_CFGSET is set after INT_PENDSET is set, then the interrupt will also propagate and be taken. Interrupt flags (signals) from the top level interrupts are level-sensitive.

The second-level interrupt registers, which provide control of the second-level Event Manager peripheral interrupts, are described in *Section 12.2: Event manager*.

For further information on the NVIC and Cortex-M3 exceptions, refer to the ARM® Cortex-M3 Technical Reference Manual and the ARM ARMv7-M Architecture Reference Manual.

12.1.1 Non-maskable interrupt (NMI)

The non-maskable interrupt (NMI) is a special case. Despite being one of the 10 standard ARM® Cortex-M3 NVIC interrupts, it is sourced from the Event Manager like a peripheral interrupt. The NMI has two second-level sources; failure of the 24 MHz crystal and watchdog low water mark.

1. Failure of the 24 MHz crystal: If the STM32W108's main clock, SCLK, is operating from the 24 MHz crystal and the crystal fails, the STM32W108 detects the failure and automatically switch to the internal 12 MHz RC clock. When this failure detection and switch has occurred, the STM32W108 triggers the CLK24M_FAIL second-level interrupt, which then triggers the NMI.
2. Watchdog low water mark: If the STM32W108's watchdog is active and the watchdog counter has not been reset for 1.792 seconds, the watchdog triggers the WATCHDOG_INT second level interrupt, which then triggers the NMI.

12.1.2 Faults

Four of the exceptions in the NVIC are faults: Hard Fault, Memory Fault, Bus Fault, and Usage Fault. Of these four, three of the faults (Hard Fault, Memory Fault, and Usage Fault) are all standard ARM® Cortex-M3 exceptions.

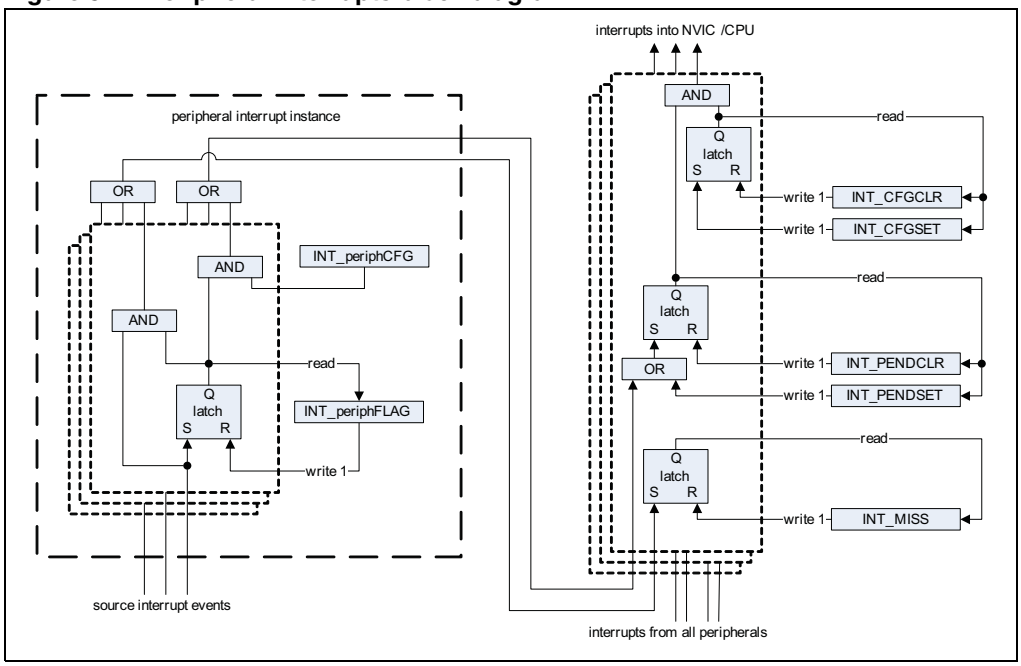
The Bus Fault, though, is derived from STM32W108-specific sources. The Bus Fault sources are recorded in the SCS_AFSR register. Note that it is possible for one access to set multiple SCS_AFSR bits. Also note that MPU configurations could prevent most of these bus fault accesses from occurring, with the advantage that illegal writes are made precise faults. The four bus faults are:

- WRONGSIZE - Generated by an 8-bit or 16-bit read or write of an APB peripheral register. This fault can also result from an unaligned 32-bit access.
- PROTECTED - Generated by a user mode (unprivileged) write to a system APB or AHB peripheral or protected RAM.
- RESERVED - Generated by a read or write to an address within an APB peripheral's 4-Kbyte block range, but the address is above the last physical register in that block range. Also generated by a read or write to an address above the top of RAM or Flash memory.
- MISSED - Generated by a second SCS_AFSR fault. In practice, this bit is not seen since a second fault also generates a hard fault, and the hard fault preempts the bus fault.

12.2 Event manager

While the standard ARM® Cortex-M3 Nested Vectored Interrupt Controller provides top-level interrupts into the CPU, the Event Manager provides second-level interrupts. The Event Manager takes a large variety of hardware interrupt sources from the peripherals and merges them into a smaller group of interrupts in the NVIC. Effectively, all second-level interrupts from a peripheral are "ORed" together into a single interrupt in the NVIC. In addition, the Event Manager provides missed indicators for the top-level peripheral interrupts with the register INT_MISS.

Figure 52. Peripheral interrupts block diagram



The description of each peripheral's interrupt configuration and flag registers can be found in the chapters of this datasheet describing each peripheral.

Given a peripheral, 'periph', the Event Manager registers (INT_periphCFG and INT_periphFLAG) follow the form:

- INT_periphCFG enables and disables second-level interrupts. Writing 1 to a bit in the INT_periphCFG register enables the second-level interrupt. Writing 0 to a bit in the INT_periphCFG register disables it. The INT_periphCFG register behaves like a mask, and is responsible for allowing the INT_periphFLAG bits to propagate into the top level NVIC interrupts.
- INT_periphFLAG indicates second-level interrupts that have occurred. Writing 1 to a bit in a INT_periphFLAG register clears the second-level interrupt. Writing 0 to any bit in the INT_periphFLAG register is ineffective. The INT_periphFLAG register is always active and may be set or cleared at any time, meaning if any second-level interrupt occurs, then the corresponding bit in the INT_periphFLAG register is set regardless of the state of INT_periphCFG.

If a bit in the INT_periphCFG register is set after the corresponding bit in the INT_periphFLAG register is set then the second-level interrupt propagates into the top level interrupts. The interrupt flags (signals) from the second-level interrupts into the top-level interrupts are level-sensitive. If a top-level NVIC interrupt is driven by a second-level EM interrupt, then the top-level NVIC interrupt cannot be cleared until all second-level EM interrupts are cleared.

The INT_periphFLAG register bits are designed to remain set if the second-level interrupt event re-occurs at the same moment as the INT_periphFLAG register bit is being cleared. This ensures the re-occurring second-level interrupt event is not missed.

If another enabled second-level interrupt event of the same type occurs before the first interrupt event is cleared, the second interrupt event is lost because no counting or queuing is used. However, this condition is detected and stored in the top-level INT_MISS register to

facilitate software detection of such problems. The INT_MISS register is "acknowledged" in the same way as the INT_periphFLAG register-by writing a 1 into the corresponding bit to be cleared.

Table 124 provides a map of all peripheral interrupts. This map lists the top level NVIC Interrupt bits and, if there is one, the corresponding second level EM Interrupt register bits that feed the top level interrupts.

Table 124. NVIC and EM peripheral interrupt map

NVIC Interrupt (top level)		EM Interrupt (second level)		NVIC Interrupt (top level)		EM Interrupt (second level)	
16	INT_DEBUG			5	INT_SC1	INT_SC1FLAG register	
15	INT_IRQD					14	INT_SC1PARERR
14	INT_IRQC					13	INT_SC1FRMERR
13	INT_IRQB					12	INT_SCTXULDB
12	INT_IRQA					11	INT_SCTXULDA
11	INT_ADC	INT_ADCFLAG register				10	INT_SCRXULDB
		4	INT_ADCOVF			9	INT_SCRXULDA
		3	INT_ADCSAT			8	INT_SCNAK
		2	INT_ADCULDFULL			7	INT_SCCDMFIN
		1	INT_ADCULDHALF			6	INT_SCTXFIN
		0	INT_ADCDATA			5	INT_SCRXFIN
10	INT_MACRX					4	INT_SCTXUND
9	INT_MACTX					3	INT_SCRXOVF
8	INT_MACTMR					2	INT_SCTXIDLE
7	INT_SEC					1	INT_SCTXFREE
6	INT_SC2	INT_SC2FLAG register				0	INT_SCRXVAL
		12	INT_SCTXULDB	4	INT_SLEEPTMR		
		11	INT_SCTXULDA	3	INT_BB		
		10	INT_SCRXULDB	2	INT_MGMT		
		9	INT_SCRXULDA	1	INT_TMR2	INT_TMR2FLAG register	
		8	INT_SCNAK			6	INT_TMRTIF
		7	INT_SCCDMFIN			4	INT_TMRCC4IF
		6	INT_SCTXFIN			3	INT_TMRCC3IF
		5	INT_SCRXFIN			2	INT_TMRCC2IF
		4	INT_SCTXUND			1	INT_TMRCC1IF
		3	INT_SCRXOVF			0	INT_TMRUIF
		2	INT_SCTXIDLE	0	INT_TMR1	INT_TMR1FLAG register	
		1	INT_SCTXFREE			6	INT_TMRTIF
		0	INT_SCRXVAL			4	INT_TMRCC4IF
						3	INT_TMRCC3IF
						2	INT_TMRCC2IF
						1	INT_TMRCC1IF
						0	INT_TMRUIF

12.3 Nested vectored interrupt controller (NVIC) interrupts

12.3.1 Top-level set interrupts configuration register (INT_CFGSET)

Address: 0xE000E100
Reset value: 0x0000 0000

Table 125. Top-level set interrupts configuration register (INT_CFGSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															INT_DEBUG
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 INT_DEBUG: Write 1 to enable debug interrupt. (Writing 0 has no effect.)

Bit 15 INT_IRQD: Write 1 to enable IRQD interrupt. (Writing 0 has no effect.)

Bit 14 INT_IRQC: Write 1 to enable IRQC interrupt. (Writing 0 has no effect.)

Bit 13 INT_IRQB: Write 1 to enable IRQB interrupt. (Writing 0 has no effect.)

Bit 12 INT_IRQA: Write 1 to enable IRQA interrupt. (Writing 0 has no effect.)

Bit 11 INT_ADC: Write 1 to enable ADC interrupt. (Writing 0 has no effect.)

Bit 10 INT_MACRX: Write 1 to enable MAC receive interrupt. (Writing 0 has no effect.)

Bit 9 INT_MACTX: Write 1 to enable MAC transmit interrupt. (Writing 0 has no effect.)

Bit 8 INT_MACTMR: Write 1 to enable MAC timer interrupt. (Writing 0 has no effect.)

Bit 7 INT_SEC: Write 1 to enable security interrupt. (Writing 0 has no effect.)

Bit 6 INT_SC2: Write 1 to enable serial controller 2 interrupt. (Writing 0 has no effect.)

Bit 5 INT_SC1: Write 1 to enable serial controller 1 interrupt. (Writing 0 has no effect.)

Bit 4 INT_SLEEPTMR: Write 1 to enable sleep timer interrupt. (Writing 0 has no effect.)

Bit 3 INT_BB: Write 1 to enable baseband interrupt. (Writing 0 has no effect.)

Bit 2 INT_MGMT: Write 1 to enable management interrupt. (Writing 0 has no effect.)

Bit 1 INT_TIM2: Write 1 to enable timer 2 interrupt. (Writing 0 has no effect.)

Bit 0 INT_TIM1: Write 1 to enable timer 1 interrupt. (Writing 0 has no effect.)

12.3.2 Top-level clear interrupts configuration register (INT_CFGCLR)

Address: 0xE000E180
Reset value: 0x0000 0000

Table 126. Top-level clear interrupts configuration register (INT_CFGCLR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															INT_DEBUG
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 INT_DEBUG: Write 1 to disable debug interrupt. (Writing 0 has no effect.)

Bit 15 INT_IRQD: Write 1 to disable IRQD interrupt. (Writing 0 has no effect.)

Bit 14 INT_IRQC: Write 1 to disable IRQC interrupt. (Writing 0 has no effect.)

Bit 13 INT_IRQB: Write 1 to disable IRQB interrupt. (Writing 0 has no effect.)

Bit 12 INT_IRQA: Write 1 to disable IRQA interrupt. (Writing 0 has no effect.)

Bit 11 INT_ADC: Write 1 to disable ADC interrupt. (Writing 0 has no effect.)

Bit 10 INT_MACRX: Write 1 to disable MAC receive interrupt. (Writing 0 has no effect.)

Bit 9 INT_MACTX: Write 1 to disable MAC transmit interrupt. (Writing 0 has no effect.)

Bit 8 INT_MACTMR: Write 1 to disable MAC timer interrupt. (Writing 0 has no effect.)

Bit 7 INT_SEC: Write 1 to disable security interrupt. (Writing 0 has no effect.)

Bit 6 INT_SC2: Write 1 to disable serial controller 2 interrupt. (Writing 0 has no effect.)

Bit 5 INT_SC1: Write 1 to disable serial controller 1 interrupt. (Writing 0 has no effect.)

Bit 4 INT_SLEEPTMR: Write 1 to disable sleep timer interrupt. (Writing 0 has no effect.)

Bit 3 INT_BB: Write 1 to disable baseband interrupt. (Writing 0 has no effect.)

Bit 2 INT_MGMT: Write 1 to disable management interrupt. (Writing 0 has no effect.)

Bit 1 INT_TIM2: Write 1 to disable timer 2 interrupt. (Writing 0 has no effect.)

Bit 0 INT_TIM1: Write 1 to disable timer 1 interrupt. (Writing 0 has no effect.)

12.3.3 Top-level set interrupts pending register (INT_PENDSET)

Address: 0xE000E200
Reset value: 0x0000 0000

Table 127. Top-level set interrupts pending register (INT_PENDSET)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															INT_DEBUG
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 INT_DEBUG: Write 1 to pend debug interrupt. (Writing 0 has no effect.)

Bit 15 INT_IRQD: Write 1 to pend IRQD interrupt. (Writing 0 has no effect.)

Bit 14 INT_IRQC: Write 1 to pend IRQC interrupt. (Writing 0 has no effect.)

Bit 13 INT_IRQB: Write 1 to pend IRQB interrupt. (Writing 0 has no effect.)

Bit 12 INT_IRQA: Write 1 to pend IRQA interrupt. (Writing 0 has no effect.)

Bit 11 INT_ADC: Write 1 to pend ADC interrupt. (Writing 0 has no effect.)

Bit 10 INT_MACRX: Write 1 to pend MAC receive interrupt. (Writing 0 has no effect.)

Bit 9 INT_MACTX: Write 1 to pend MAC transmit interrupt. (Writing 0 has no effect.)

Bit 8 INT_MACTMR: Write 1 to pend MAC timer interrupt. (Writing 0 has no effect.)

Bit 7 INT_SEC: Write 1 to pend security interrupt. (Writing 0 has no effect.)

Bit 6 INT_SC2: Write 1 to pend serial controller 2 interrupt. (Writing 0 has no effect.)

Bit 5 INT_SC1: Write 1 to pend serial controller 1 interrupt. (Writing 0 has no effect.)

Bit 4 INT_SLEEPTMR: Write 1 to pend sleep timer interrupt. (Writing 0 has no effect.)

Bit 3 INT_BB: Write 1 to pend baseband interrupt. (Writing 0 has no effect.)

Bit 2 INT_MGMT: Write 1 to pend management interrupt. (Writing 0 has no effect.)

Bit 1 INT_TIM2: Write 1 to pend timer 2 interrupt. (Writing 0 has no effect.)

Bit 0 INT_TIM1: Write 1 to pend timer 1 interrupt. (Writing 0 has no effect.)

12.3.4 Top-level clear interrupts pending register (INT_PENDCLR)

Address: 0xE000E280
Reset value: 0x0000 0000

Table 128. Top-level clear interrupts pending register (INT_PENDCLR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															INT_DEBUG
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 INT_DEBUG: Write 1 to unpend debug interrupt. (Writing 0 has no effect.)

Bit 15 INT_IRQD: Write 1 to unpend IRQD interrupt. (Writing 0 has no effect.)

Bit 14 INT_IRQC: Write 1 to unpend IRQC interrupt. (Writing 0 has no effect.)

Bit 13 INT_IRQB: Write 1 to unpend IRQB interrupt. (Writing 0 has no effect.)

Bit 12 INT_IRQA: Write 1 to unpend IRQA interrupt. (Writing 0 has no effect.)

Bit 11 INT_ADC: Write 1 to unpend ADC interrupt. (Writing 0 has no effect.)

Bit 10 INT_MACRX: Write 1 to unpend MAC receive interrupt. (Writing 0 has no effect.)

Bit 9 INT_MACTX: Write 1 to unpend MAC transmit interrupt. (Writing 0 has no effect.)

Bit 8 INT_MACTMR: Write 1 to unpend MAC timer interrupt. (Writing 0 has no effect.)

Bit 7 INT_SEC: Write 1 to unpend security interrupt. (Writing 0 has no effect.)

Bit 6 INT_SC2: Write 1 to unpend serial controller 2 interrupt. (Writing 0 has no effect.)

Bit 5 INT_SC1: Write 1 to unpend serial controller 1 interrupt. (Writing 0 has no effect.)

Bit 4 INT_SLEEPTMR: Write 1 to unpend sleep timer interrupt. (Writing 0 has no effect.)

Bit 3 INT_BB: Write 1 to unpend baseband interrupt. (Writing 0 has no effect.)

Bit 2 INT_MGMT: Write 1 to unpend management interrupt. (Writing 0 has no effect.)

Bit 1 INT_TIM2: Write 1 to unpend timer 2 interrupt. (Writing 0 has no effect.)

Bit 0 INT_TIM1: Write 1 to unpend timer 1 interrupt. (Writing 0 has no effect.)

12.3.5 Top-level active interrupts register (INT_ACTIVE)

Address: 0xE000E300
Reset value: 0x0000 0000

Table 129. Top-level active interrupts register (INT_ACTIVE)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															INT_DEBUG
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_IRQD	INT_IRQC	INT_IRQB	INT_IRQA	INT_ADC	INT_MACRX	INT_MACTX	INT_MACTMR	INT_SEC	INT_SC2	INT_SC1	INT_SLEEPTMR	INT_BB	INT_MGMT	INT_TIM2	INT_TIM1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 16 INT_DEBUG: Debug interrupt active.

Bit 15 INT_IRQD: IRQD interrupt active.

Bit 14 INT_IRQC: IRQC interrupt active.

Bit 13 INT_IRQB: IRQB interrupt active.

Bit 12 INT_IRQA: IRQA interrupt active.

Bit 11 INT_ADC: ADC interrupt active.

Bit 10 INT_MACRX: MAC receive interrupt active.

Bit 9 INT_MACTX: MAC interrupt active.

Bit 8 INT_MACTMR: MAC timer interrupt active.

Bit 7 INT_SEC: Ssecurity interrupt active.

Bit 6 INT_SC2: Serial controller 2 interrupt active.

Bit 5 INT_SC1: Serial controller 1 interrupt active.

Bit 4 INT_SLEEPTMR: Sleep timer interrupt active.

Bit 3 INT_BB: Baseband interrupt active.

Bit 2 INT_MGMT: Management interrupt active.

Bit 1 INT_TIM2: Timer 2 interrupt active.

Bit 0 INT_TIM1: Timer 1 interrupt active.

12.3.6 Top-level missed interrupts register (INT_MISS)

Address: 0x4000 A820
Reset value: 0x0000 0000

Table 130. Top-level missed interrupts register (INT_MISS)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_MISSIRQD	INT_MISSIRQC	INT_MISSIRQB	INT_MISSIRQA	INT_MISSADC	INT_MISSMACRX	INT_MISSMACTX	INT_MISSMACTMR	INT_MISSSEC	INT_MISSSC2	INT_MISSSC1	INT_MISSSLEEP	INT_MISSBB	INT_MISSMGMT	Reserved	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 15 INT_MISSIRQD: IRQD interrupt missed.

Bit 14 INT_MISSIRQC: IRQC interrupt missed.

Bit 13 INT_MISSIRQB: IRQB interrupt missed.

Bit 12 INT_MISSIRQA: IRQA interrupt missed.

Bit 11 INT_MISSADC: ADC interrupt missed.

Bit 10 INT_MISSMACRX: MAC receive interrupt missed.

Bit 9 INT_MISSMACTX: MAC transmit interrupt missed.

Bit 8 INT_MISSMACTMR: MAC Timer interrupt missed.

Bit 7 INT_MISSSEC: Security interrupt missed.

Bit 6 INT_MISSSC2: Serial controller 2 interrupt missed.

Bit 5 INT_MISSSC1: Serial controller 1 interrupt missed.

Bit 4 INT_MISSSLEEP: Sleep timer interrupt missed.

Bit 3 INT_MISSBB: Baseband interrupt missed.

Bit 2 INT_MISSMGMT: Management interrupt missed.

12.3.7 Auxiliary fault status register (SCS_AFSR)

Address: 0xE000ED3C
Reset value: 0x0000 0000

Table 131. Auxiliary fault status register (SCS_AFSR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												WRONGSIZE	PROTECTED	RESERVED	MISSED
												rw	rw	rw	rw

Bit 3 WRONGSIZE
A bus fault resulted from an 8-bit or 16-bit read or write of an APB peripheral register. This fault can also result from an unaligned 32-bit access.

Bit 2 PROTECTED
A bus fault resulted from a user mode (unprivileged) write to a system APB or AHB peripheral or protected RAM.

Bit 1 RESERVED
A bus fault resulted from a read or write to an address within an APB peripheral's 4-Kbyte block range, but above the last physical register in that block. Can also result from a read or write to an address above the top of RAM or flash.

Bit 0 MISSED
A bus fault occurred when a bit was already set in this register.

13 Debug support

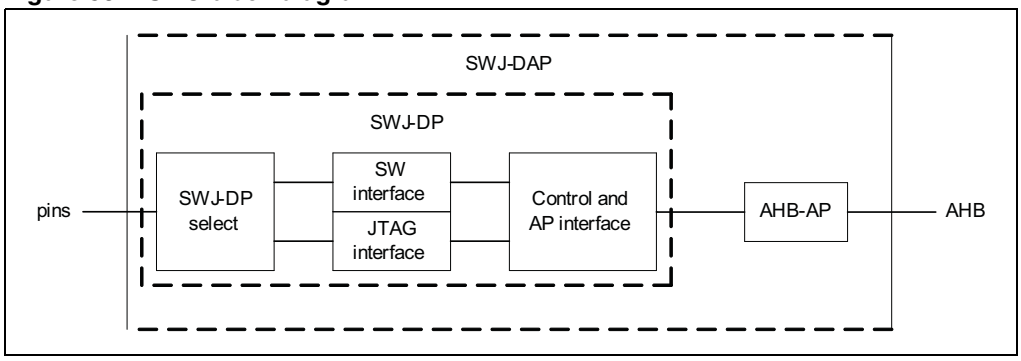
The STM32W108 includes a standard Serial Wire and JTAG (SWJ) Interface. The SWJ is the primary debug and programming interface of the STM32W108. The SWJ gives debug tools access to the internal buses of the STM32W108, and allows for non-intrusive memory and register access as well as CPU halt-step style debugging. Therefore, any design implementing the STM32W108 should make the SWJ signals readily available.

Serial Wire is an ARM® standard, bi-directional, two-wire protocol designed to replace JTAG, and provides all the normal JTAG debug and test functionality. JTAG is a standard five-wire protocol providing debug and test functionality. In addition, the two Serial Wire signals (SWDIO and SWCLK) are overlaid on two of the JTAG signals (JTMS and JTCK). This keeps the design compact and allows debug tools to switch between Serial Wire and JTAG as needed, without changing pin connections.

While Serial Wire and JTAG offer the same debug and test functionality, ST recommends Serial Wire. Serial Wire uses only two pins instead of five, and offers a simple communication protocol, high performance data rates, low power, built-in error detection, and protection from glitches.

The ARM® CoreSight Debug Access Port (DAP) comprises the Serial Wire and JTAG Interface (SWJ).The DAP includes two primary components: a debug port (the SWJ-DP) and an access port (the AHB-AP). The SWJ-DP provides external debug access, while the AHB-AP provides internal bus access. An external debug tool connected to the STM32W108's debug pins communicates with the SWJ-DP. The SWJ-DP then communicates with the AHB-AP. Finally, the AHB-AP communicates on the internal bus.

Figure 53. SWJ block diagram



Serial Wire and JTAG share five pins:

- JRST
- JTDO
- JTDI
- SWDIO/JTMS
- SWCLK/JTCK

Since these pins can be repurposed, refer to *Section 3: Pinout and pin description on page 15* and *Section 8: General-purpose input/outputs on page 58* for complete pin descriptions and configurations.

13.1 STM32W108 JTAG TAP connection

The STM32W108 MCU integrates two serially-connected JTAG TAPs in the following order; the TMC TAP dedicated for Test (IR is 4-bit wide) and the Cortex™-M3 TAP (IR is 4-bit wide).

To access the TAP of the Cortex-M3 for debug purposes:

1. First, it is necessary to shift the BYPASS instruction of the TMC TAP.
2. Then, for each IR shift, the scan chain contains 8 bits (= 4 + 4) and the unused TAP instruction must be shifted in using the BYPASS instruction.
3. For each data shift, the unused TAP, which is in BYPASS mode, adds 1 extra data bit in the data scan chain.

Note: ***Important****: Once Serial-Wire is selected using the dedicated ARM JTAG sequence, the TMC TAP is automatically disabled (JTMS forced high).*

14 Electrical characteristics

14.1 Parameter conditions

Unless otherwise specified, all voltages are referenced to V_{SS}.

14.1.1 Minimum and maximum values

Unless otherwise specified the minimum and maximum values are guaranteed in the worst conditions of ambient temperature, supply voltage and frequencies by tests in production on 100% of the devices with an ambient temperature at T_A = 25 °C and T_A = $T_A$max (given by the selected temperature range).

Data based on characterization results, design simulation and/or technology characteristics are indicated in the table footnotes and are not tested in production. Based on characterization, the minimum and maximum values refer to sample tests and represent the mean value plus or minus three times the standard deviation (mean±3Σ).

14.1.2 Typical values

Unless otherwise specified, typical data are based on T_A = 25 °C, V_{DD} = 3.3 V (for the 2 V ≤V_{DD} ≤3.6 V voltage range). They are given only as design guidelines and are not tested.

Typical ADC accuracy values are determined by characterization of a batch of samples from a standard diffusion lot over the full temperature range, where 95% of the devices have an error less than or equal to the value indicated (mean±2Σ).

14.1.3 Typical curves

Unless otherwise specified, all typical curves are given only as design guidelines and are not tested.

14.1.4 Loading capacitor

The loading conditions used for pin parameter measurement are shown in *Figure 54*.

14.1.5 Pin input voltage

The input voltage measurement on a pin of the device is described in *Figure 55*.

Figure 54. Pin loading conditions

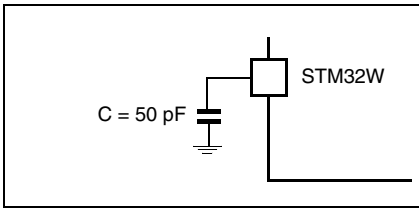
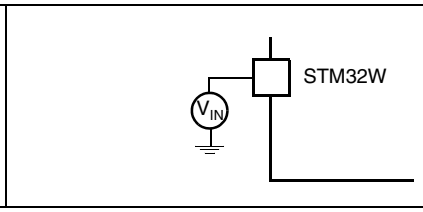


Figure 55. Pin input voltage



14.2 Absolute maximum ratings

Stresses above the absolute maximum ratings listed in *Table 132: Voltage characteristics*, *Table 133: Current characteristics*, and *Table 134: Thermal characteristics* may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Table 132. Voltage characteristics

Ratings	Min.	Max.	Unit
Regulator input voltage (VDD_PADS)	-0.3	+3.6	V
Analog, Memory and Core voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_MEM, VDD_PRE, VDD_SYNTH, VDD_CORE)	-0.3	+2.0	V
Voltage on RF_P,N; RF_TX_ALT_P,N	-0.3	+3.6	V
RF Input Power (for max level for correct packet reception see *Table 153: Receive characteristics*) RX signal into a lossless balun	–	+15	dBm
Voltage on any GPIO (PA[7:0], PB[7:0], PC[7:0]), SWCLK, NRST, VREG_OUT	-0.3	VDD_PADS +0.3	V
Voltage on BIAS_R, OSCA, OSCB	-0.3	VDD_PADSA +0.3	V

Table 133. Current characteristics

Symbol	Ratings	Max.	Unit
I_{VDD}	Total current into V_{DD}/V_{DDA} power lines (source)	150	mA
I_{VSS}	Total current out of V_{SS} ground lines (sink)	150	
I_{IO}	Output current sunk by any I/O and control pin	25	
	Output current source by any I/Os and control pin	–25	
$I_{INJ(PIN)}$	Injected current on NRST pin	± 5	
	Injected current on HSE OSC_IN and LSE OSC_IN pins	± 5	
	Injected current on any other pin	± 5	
$\Sigma I_{INJ(PIN)}$	Total injected current (sum of all I/O and control pins)	± 25	

Table 134. Thermal characteristics

Symbol	Ratings	Value	Unit
T_{STG}	Storage temperature range	–40 to +140	°C
T_J	Maximum junction temperature	150	°C

14.3 Operating conditions

14.3.1 General operating conditions

Table 135. General operating conditions

Symbol	Parameter	Min.	Typ.	Max.	Unit
–	Regulator input voltage (VDD_PADS)	2.1	–	3.6	V
–	Analog and memory input voltage (VDD_24MHZ, VDD_VCO, VDD_RF, VDD_IF, VDD_PADSA, VDD_MEM, VDD_PRE, and VDD_SYNTH)	1.7	1.8	1.9	V
–	Core input voltage (VDD_CORE)	1.18	1.25	1.32	V
T_{OPER}	Operating temperature range	-40	–	+85	°C

14.3.2 Operating conditions at power-up

Power-on resets (POR HV and POR LV)

The STM32W108 measures the voltage levels supplied to the three power domains. If a supply voltage drops below a low threshold, then a reset is applied. The reset is released if the supply voltage rises above a high threshold. There are three detection circuits for power on reset as follows:

- POR HV monitors the always on domain supply voltage. Thresholds are given in *Table 136*.
- POR LVcore monitors the core domain supply voltage. Thresholds are given in *Table 137*.
- POR LVmem monitors the memory supply voltage. Thresholds are given in *Table 138*.

Table 136. POR HV thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
Always-on domain release		1.0	1.2	1.4	V
Always-on domain assert		0.5	0.6	0.7	V
Supply rise time	From 0.5 V to 1.7 V	–	–	250	µs

Table 137. POR LVcore thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
1.25 V domain release		0.9	1.0	1.1	V
1.25 V domain assert		0.8	0.9	1.0	V

Table 138. POR LVmem thresholds

Parameter	Test conditions	Min	Typ	Max	Unit
1.8 V domain release		1.35	1.5	1.65	V
1.8 V domain assert		1.26	1.4	1.54	V

The POR LVcore and POR LVmem reset sources are merged to provide a single reset source, POR LV, to the Reset Generation module, since the detection of either event needs to reset the same system modules.

NRST pin

A single active low pin, NRST, is provided to reset the system. This pin has a Schmitt triggered input.

To afford good noise immunity and resistance to switch bounce, the pin is filtered with the Reset Filter module and generates the reset source RSTB to the Reset Generation module.

Table 139. Reset filter specification for RSTB

Parameter	Min	Typ	Max	Unit
Reset filter time constant	2.1	12.0	16.0	µs
Reset pulse width to guarantee a reset	26.0	–	–	µs
Reset pulse width guaranteed not to cause a reset	0	–	1.0	µs

14.3.3 Absolute maximum ratings (electrical sensitivity)

Based on three different tests (ESD, LU) using specific measurement methods, the device is stressed in order to determine its performance in terms of electrical sensitivity.

Electrostatic discharge (ESD)

Electrostatic discharges (a positive then a negative pulse separated by 1 second) are applied to the pins of each sample according to each pin combination. The sample size depends on the number of supply pins in the device (3 parts × (n+1) supply pins). This test conforms to the JESD22-A114/C101 standard.

Table 140. ESD absolute maximum ratings

Symbol	Ratings	Conditions	Class	Maximum value(1)	Unit
$V_{ESD(HBM)}$	Electrostatic discharge voltage (human body model)	T_A = +25 °C in compliance with JESD22-A114	2	±2000	V
$V_{ESD(CDM)}$	Electrostatic discharge voltage (charge device model) for non-RF pins	T_A = +25 °C in compliance with JESD22-A114	II	±400	
	Electrostatic discharge voltage (charge device model) for RF pins			±225	
MSL	Moisture sensitivity level	–	–	MSL3	–

1. Based on characterization results, not tested in production.

Static latch-up

Two complementary static tests are required on six parts to assess the latch-up performance:

- A supply overvoltage is applied to each power supply pin
- A current injection is applied to each input, output and configurable I/O pin

These tests are compliant with EIA/JESD 78A IC latch-up standard.

Table 141. Electrical sensitivities

Symbol	Parameter	Conditions	Class
LU	Static latch-up class	T_A = +105 °C conforming to JESD78A	II level A

14.4 ADC characteristics

Table 142 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling clock of 1 MHz. ADC_HVSELP and ADC_HVSELN are programmed to 0 to disable the input buffer. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; 0 dBFS level (where full-scale is a 1.2 V p-p swing). The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$; -6 dBFS level (where full-scale is a 2.4 V p-p swing).

Table 142. ADC module key parameters for 1 MHz sampling(1)

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	32	64	128	256	512	1024	2048	4096
Nyquist Freq (kHz)	15.6	7.81	3.91	1.95	0.977	0.488	0.244	0.122
3 dB Cut-off (kHz)	9.43	4.71	2.36	1.18	0.589	0.295	0.147	0.0737
INL (codes peak)	0.083	0.092	0.163	0.306	0.624	1.229	2.451	4.926
INL (codes RMS)	0.047	0.051	0.093	0.176	0.362	0.719	1.435	2.848
DNL (codes peak)	0.028	0.035	0.038	0.044	0.074	0.113	0.184	0.333
DNL (codes RMS)	0.008	0.009	0.011	0.014	0.019	0.029	0.048	0.079
ENOB (from single-cycle test)	5.6	7.0	8.6	10.1	11.5	12.6	13.0	13.2
SNR (dB) Single-Ended Differential	 35 35	 44 44	 53 53	 62 62	 70 71	 75 77	 77 79	 77 80
SINAD (dB) Single-Ended Differential	 35 35	 44 44	 53 53	 61 62	 67 70	 69 75	 70 76	 70 76
SDFR (dB) Single-Ended Differential	 59 60	 68 69	 72 77	 72 80	 72 81	 72 81	 72 81	 73 81

Table 142. ADC module key parameters for 1 MHz sampling(1) (continued)

Parameter	Performance							
THD (dB) Single-Ended Differential	 -45 -45	 -54 -54	 -62 -63	 -67 -71	 -69 -75	 -69 -76	 -69 -76	 - 69 -76
ENOB (from SNR) Single-Ended Differential	 5.6 5.6	 7.1 7.1	 8.6 8.6	 10.0 10.1	 11.3 11.4	 12.2 12.5	 12.4 12.9	 12.5 12.9
ENOB (from SINAD) Single-Ended Differential	 5.5 5.6	 7.0 7.0	 8.5 8.5	 9.9 10.0	 10.9 11.3	 11.2 12.1	 11.3 12.3	 11.3 12.4
Equivalent ADC Bits	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]	13 [15:3]	14 [15:2]

1. INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of *Table 142*. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).

Table 143 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling rate of 6 MHz. ADC_HVSELP and ADC_HVSELN are programmed to 0 to disable the input buffer. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$; 0 dBFS level (where full-scale is a 1.2 V p-p swing). The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$; -6 dBFS level (where full-scale is a 2.4 V p-p swing) and a common mode voltage of 0.6 V.

Table 143. ADC module key parameters for input buffer disabled and 6 MHz sampling(1)

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	5.33	10.7	21.3	42.7	85.3	171	341	683
Nyquist Freq (kHz)	93.8	46.9	23.4	11.7	5.86	2.93	1.47	0.732
3 dB Cut-off (kHz)	56.6	28.3	14.1	7.07	3.54	1.77	0.884	0.442
INL (codes peak)	0.084	0.084	0.15	0.274	0.518	1.057	2.106	4.174
INL (codes RMS)	0.046	0.044	0.076	0.147	0.292	0.58	1.14	2.352
DNL (codes peak)	0.026	0.023	0.044	0.052	0.096	0.119	0.196	0.371
DNL (codes RMS)	0.007	0.009	0.013	0.015	0.024	0.03	0.05	0.082
ENOB (from single-cycle test)	5.6	7.0	8.5	10.0	11.4	12.6	13.1	13.2
SNR (dB) Single-Ended Differential	 35 35	 44 44	 53 53	 62 62	 70 71	 75 77	 76 79	 77 80
SINAD (dB) Single-Ended Differential	 35 35	 44 44	 53 53	 62 62	 68 70	 71 75	 71 77	 71 77

Table 143. ADC module key parameters for input buffer disabled and 6 MHz sampling(1) (continued)

Parameter	Performance							
SDFR (dB)								
Single-Ended	60	68	75	75	75	75	75	75
Differential	60	69	77	80	80	80	80	80
THD (dB)								
Single-Ended	-45	-54	-63	-68	-70	-70	-70	-70
Differential	-45	-54	-63	-71	-76	-77	-78	-78
ENOB (from SNR)								
Single-Ended	5.6	7.1	8.6	10.0	11.4	12.1	12.4	12.5
Differential	5.6	7.1	8.6	10.1	11.5	12.5	12.9	13.0
ENOB (from SINAD)								
Single-Ended	5.5	7.0	8.5	9.9	11.0	11.4	11.5	11.5
Differential	5.6	7.1	8.6	10.1	11.4	12.4	12.8	13.0
Equivalent ADC Bits	5 [15:11]	6 [15:10]	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]

1. *INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 143. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).*

Table 144 describes the key ADC parameters measured at 25°C and VDD_PADS at 3.0 V, for a sampling rate of 6 MHz. ADC_HVSELP and ADC_HVSELN are programmed to 1 to enable the input buffer. The single-ended measurements were done at f_{input} = 7.7% $f_{Nyquist}$, level = 1.2 V p-p swing centered on 1.5 V. The differential measurements were done at f_{input} = 7.7% $f_{Nyquist}$, level = 1.2 V p-p swing and a comon mode voltage of 1.5 V.

Table 144. ADC module key parameters for input buffer enabled and 6MHz sampling(1)

Parameter	Performance							
ADC_PERIOD	0	1	2	3	4	5	6	7
Conversion Time (µs)	32	64	128	256	512	1024	2048	4096
Nyquist Freq (kHz)	93.8	46.9	23.4	11.7	5.86	2.93	1.47	0.732
3 dB Cut-off (kHz)	56.6	28.3	14.1	7.07	3.54	1.77k	0.884	0.442
INL (codes peak)	0.055	0.032	0.038	0.07	0.123	0.261	0.522	1.028
INL (codes RMS)	0.028	0.017	0.02	0.04	0.077	0.167	0.326	0.65
DNL (codes peak)	0.028	0.017	0.02	0.04	0.077	0.167	0.326	0.65
DNL (codes RMS)	0.01	0.006	0.006	0.007	0.008	0.013	0.023	0.038
ENOB (from single-cycle test)	3.6	5.0	6.6	8.1	9.5	10.7	11.3	11.6
SNR (dB) Single-Ended Differential	 23 23	32 32	 41 41	 50 50	 59 59	 65 66	 67 69	 68 71
SINAD (dB) Single-Ended Differential	 23 23	 32 32	 41 41	 50 50	 58 59	 64 66	 66 69	 66 71
SDFR (dB) Single-Ended Differential	 48 48	 56 57	 65 65	 72 74	 72 82	 72 88	 73 88	 73 88
THD (dB) Single-Ended Differential	 -33 -33	 -42 -42	 -51 -51	 -59 -60	 -66 -69	 -68 -76	 -68 -80	 -68 -82
ENOB (from SNR) Single-Ended Differential	 3.6 3.6	 5.1 5.1	 6.6 6.6	 8.1 8.1	 9.5 9.5	 10.5 10.7	 10.9 11.3	 11 11.5
ENOB (from SINAD) Single-Ended Differential	 3.6 3.6	 5.0 5.1	 6.5 6.6	 8.0 8.0	 9.4 9.5	 10.3 10.6	 10.7 11.3	 10.7 11.4
Equivalent ADC Bits	7 [15:9]	8 [15:8]	9 [15:7]	10 [15:6]	11 [15:5]	12 [15:4]	13 [15:3]	14 [15:2]

1. *INL and DNL are referenced to a LSB of the Equivalent ADC Bits shown in the last row of Table 144. ENOB (effective number of bits) can be calculated from either SNR (signal to non-harmonic noise ratio) or SINAD (signal-to-noise and distortion ratio).*

Table 145 lists other specifications for the ADC not covered in *Table 142*, *Table 143*, and *Table 144*.

Table 145. ADC characteristics

Parameter	Min.	Typ.	Max.	Units
VREF	1.17	1.2	1.23	V
VREF output current	–	–	1	mA
VREF load capacitance	–	–	10	nF
External VREF voltage range	1.1	1.2	1.3	V
External VREF input impedance	1	–	–	MΩ
Minimum input voltage Input buffer disabled Input buffer enabled	 0 0.1	 – –	 – –	V
Maximum input voltage Input buffer disabled Input buffer enabled	 – –	 – –	 VREF VDD_PADS - 0.1	V
Single-ended signal range Input buffer disabled Input buffer enabled	 0 0.1	 – –	 VREF VDD_PADS – 0.1	V
Differential signal range Input buffer disabled Input buffer enabled	 -VREF -VDD_PADS + 0.1	 – –	 +VREF +VDD_PADS - 0.1	V
Common mode range Input buffer disabled Input buffer enabled	 0	 VDD_PADS/2	 VREF	V
Input referred ADC offset	-10	–	10	mV
Input Impedance 1 MHz sample clock 6 MHz sample clock Not sampling	 1 0.5 10	 – – –	 – – –	MΩ

Note: The signal-ended ADC measurements are limited in their range and only guaranteed for accuracy within the limits shown in this table. The ADC's internal design allows for measurements outside of this range (±200 mV) when the input buffer is disabled, but the accuracy of such measurements is not guaranteed. The maximum input voltage is of more interest to the differential sampling where a differential measurement might be small, but a common mode can push the actual input voltage on one of the signals towards the upper voltage limit.

14.5 Clock frequencies

14.5.1 High frequency internal clock characteristics

Table 146. High-frequency RC oscillator characteristics

Parameter	Test conditions	Min.	Typ.	Max.	Unit
Frequency at reset		6	12	20	MHz
Frequency Steps			0.5		MHz
Duty cycle		40		60	%
Supply dependence	Change in supply = 0.1 V				
Test at supply changes: 1.8 V to 1.7 V			5	%	

14.5.2 High frequency external clock characteristics

Table 147. High-frequency crystal oscillator characteristics

Parameter	Test conditions	Min.	Typ.	Max.	Unit
Frequency	–	–	24	–	MHz
Accuracy	–	-40	–	+40	ppm
Duty cycle	–	40	–	60	%
Phase noise (at 100 kHz offset)	–	–	–	-120	dBc/Hz
Start-up time at max bias	–	–	–	1	ms
Start up time at optimal bias	–	–	–	2	ms
Current consumption	–	–	200	300	µA
Current consumption at max bias	–	–	–	1	mA
Crystal with high ESR	–	–	–	100	Ω
– Load capacitance	–	–	–	10	pF
– Crystal capacitance	–	–	–	7	pF
– Crystal power dissipation	–	–	–	200	µW
Crystal with low ESR	–	–	–	60	Ω
– Load capacitance	–	–	–	18	pF
– Crystal capacitance	–	–	–	7	pF
– Crystal power dissipation	–	–	–	1	mW

14.5.3 Low frequency internal clock characteristics

Table 148. Low-frequency RC oscillator characteristics

Parameter	Test conditions	Min.	Typ.	Max.	Unit
Nominal Frequency	After trimming	9	10	11	kHz
Analog trim step size	–	–	1	–	kHz

Table 148. Low-frequency RC oscillator characteristics (continued)

Parameter	Test conditions	Min.	Typ.	Max.	Unit
Supply dependence	For a voltage drop from 3.6 V to 3.1 V or 2.6 V to 2.1 V (without re-calibration)	–	–	1	%
Frequency dependence	Frequency variation with temperature for a change from -40 oC to +85oC (without re-calibration)	–	2	–	%

14.5.4 Low frequency external clock characteristics

Table 149. Low-frequency crystal oscillator characteristics

Parameter	Test conditions	Min.	Typ.	Max.	Unit
Frequency	–	–	32.768	–	kHz
Accuracy	Initial, temperature, and ageing	-100	–	+100	ppm
Load cap xin	–	–	27	–	pF
Load cap xout	–	–	18	–	pF
Crystal ESR	–	–	–	100	kΩ
Start-up time	–	–	–	2	s
Current consumption	At 25°C, VDD_PADS = 3.0 V	–	–	0.5	µA

14.6 DC electrical characteristics

Table 150. DC electrical characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
Regulator input voltage (VDD_PADS)		2.1	–	3.6	V
Power supply range (VDD_MEM)	Regulator output or external input	1.7	1.8	1.9	V
Power supply range (VDD_CORE)	Regulator output	1.18	1.25	1.32	V
Deep Sleep Current					
Quiescent current, internal RC oscillator disabled	-40°C, VDD_PADS = 3.6 V	–	0.4	–	µA
	+25°C, VDD_PADS = 3.6 V	–	0.4	–	µA
	+85°C, VDD_PADS = 3.6 V	–	0.7	–	µA
Quiescent current, including internal RC oscillator	-40°C, VDD_PADS = 3.6 V	–	0.7	–	µA
	+25°C, VDD_PADS = 3.6 V	–	0.7	–	µA
	+85°C, VDD_PADS = 3.6 V	–	1.1	–	µA
Quiescent current, including 32.768 kHz oscillator	-40°C, VDD_PADS = 3.6 V	–	0.8	–	µA
	+25°C, VDD_PADS = 3.6 V	–	1.0	–	µA
	+85°C, VDD_PADS = 3.6 V	–	1.5	–	µA
Quiescent current, including internal RC oscillator and 32.768 kHz oscillator	-40°C, VDD_PADS = 3.6 V	–	1.1	–	µA
	+25°C, VDD_PADS = 3.6 V	–	1.3	–	µA
	+85°C, VDD_PADS = 3.6 V	–	1.8	–	µA
Simulated deep sleep (debug mode) current	With no debugger activity	–	300	–	µA
Reset current					
Quiescent current, NRST asserted	Typ at 25°C/3 V Max at 85°C/3.6 V	–	1.2	2.0	mA
Processor and peripheral currents					
ARM® Cortex-M3, RAM, and flash memory	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex-M3 running at 12 MHz from crystal oscillator Radio and all peripherals off	–	6.0	–	mA
ARM® Cortex-M3, RAM, and flash memory	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex-M3 running at 24 MHz from crystal oscillator Radio and all peripherals off	–	7.5	–	mA

Table 150. DC electrical characteristics (continued)

Parameter	Conditions	Min.	Typ.	Max.	Unit
ARM® Cortex-M3, RAM, and flash memory sleep current	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex-M3 clocked at 12 MHz from the crystal oscillator Radio and all peripherals off	–	3.0	–	mA
ARM® Cortex-M3, RAM, and flash memory sleep current	25 °C, 1.8 V memory and 1.25 V core ARM® Cortex-M3 clocked at 6 MHz from the high frequency RC oscillator Radio and all peripherals off	–	2.0	–	mA
Serial controller current	For each controller at maximum data rate	–	0.2	–	mA
General purpose timer current	For each timer at maximum clock rate	–	0.25	–	mA
General purpose ADC current	At maximum sample rate, DMA enabled	–	1.1	–	mA
Rx current					
Radio receiver, MAC, and baseband	ARM® Cortex-M3 sleeping	–	22.0	–	mA
Total RX current (= $I_{\text{Radio receiver, MAC and baseband, CPU}} + I_{\text{RAM, and Flash memory}}$)	VDD_PADS = 3.0 V, 25 °C, ARM® Cortex-M3 running at 12 MHz	–	25.0	–	mA
	VDD_PADS = 3.0 V, 25 °C, ARM® Cortex-M3 running at 24 MHz	–	26.5	–	mA
Boost mode total RX current (= $I_{\text{Radio receiver, MAC and baseband, CPU}} + I_{\text{RAM, and Flash memory}}$)	VDD_PADS = 3.0 V, 25 °C, ARM® Cortex-M3 running at 12 MHz	–	27.0	–	mA
	VDD_PADS = 3.0 V, 2 5°C, ARM® Cortex-M3 running at 24 MHz	–	28.5	–	mA
Tx current					
Radio transmitter, MAC, and baseband	25 °C and 1.8 V core; max. power out (+3 dBm typical) ARM® Cortex-M3 sleeping	–	26.0	–	mA

Table 150. DC electrical characteristics (continued)

Parameter	Conditions	Min.	Typ.	Max.	Unit
Total Tx current (= $I_{Radio\ transmitter,\ MAC\ and\ baseband,\ CPU}$ + $I_{RAM,\ and\ Flash\ memory}$)	VDD_PADS = 3.0 V, 25 °C; maximum power setting (+7 dBm); ARM® Cortex-M3 running at 12 MHz	–	42.0	–	mA
	VDD_PADS = 3.0 V, 25 °C; +3 dBm power setting; ARM® Cortex-M3 running at 12 MHz	–	29.5	–	mA
	VDD_PADS = 3.0 V, 25 °C; 0dBm power setting; ARM® Cortex-M3 running at 12 MHz	–	27.0	–	mA
	VDD_PADS = 3.0 V, 25 °C; minimum power setting; ARM® Cortex-M3 running at 12 MHz	–	21.0	–	mA
	VDD_PADS = 3.0 V, 25 °C; maximum power setting (+7 dBm); ARM® Cortex-M3 running at 24 MHz	–	43.5	–	mA
	VDD_PADS = 3.0 V, 25 °C; +3 dBm power setting; ARM® Cortex-M3 running at 24 MHz	–	31.0	–	mA
	VDD_PADS = 3.0 V, 25 °C; 0dBm power setting; ARM® Cortex-M3 running at 24 MHz	–	28.5	–	mA
	VDD_PADS = 3.0 V, 25 °C; minimum power setting; ARM® Cortex-M3 running at 24 MHz	–	22.5	–	mA

Figure 56 shows the variation of current in Transmit mode (with the ARM® Cortex-M3 running at 12 MHz).

Figure 56. Transmit power consumption

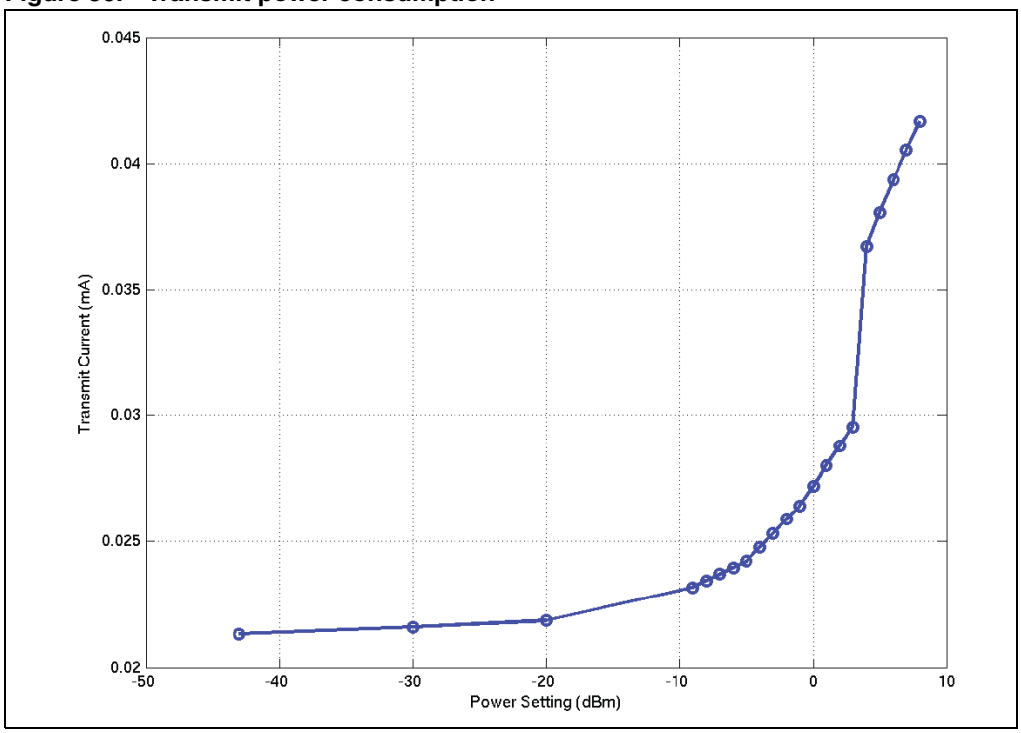
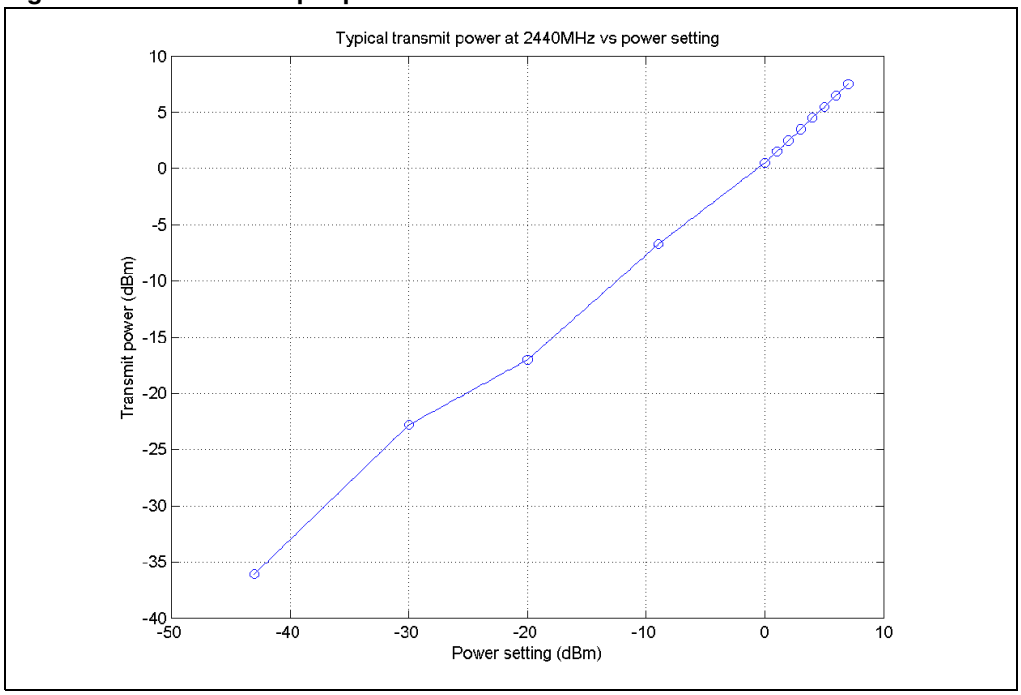


Figure 57 shows typical output power against power setting on the ST reference design.

Figure 57. Transmit output power



14.7 Digital I/O specifications

Table 151 lists the digital I/O specifications for the STM32W. The digital I/O power (named VDD_PADS) comes from three dedicated pins (Pins 23, 28 and 37). The voltage applied to these pins sets the I/O voltage.

Table 151. Digital I/O characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
Voltage supply (Regulator Input)	VDD_PADS	2.1	–	3.6	V
Low Schmitt switching threshold	V_{SWIL} Schmitt input threshold going from high to low	0.42 x VDD_PADS	–	0.50 x VDD_PADS	V
High Schmitt switching threshold	V_{SWIH} Schmitt input threshold going from low to high	0.62 x VDD_PADS	–	0.80 x VDD_PADS	V
Input current for logic 0	I_{IL}	–	–	-0.5	µA
Input current for logic 1	I_{IH}	–	–	+0.5	µA
Input pull-up resistor value	R_{IPU}	24	29	34	kΩ
Input pull-down resistor value	R_{IPD}	24	29	34	kΩ

Table 151. Digital I/O characteristics (continued)

Parameter	Conditions	Min.	Typ.	Max.	Unit
Output voltage for logic 0	V_{OL} (I_{OL} = 4 mA for standard pads, 8 mA for high current pads)	0	–	0.18 x VDD_PADS	V
Output voltage for logic 1	V_{OH} (I_{OH} = 4 mA for standard pads, 8 mA for high current pads)	0.82 x VDD_PADS	–	VDD_PADS	V
Output source current (standard current pad)	I_{OHS}	–	–	4	mA
Output sink current (standard current pad)	I_{OLS}	–	–	4	mA
Output source current high current pad: PA6, PA7, PB6, PB7, PC0	I_{OHH}	–	–	8	mA
Output sink current high current pad: PA6, PA7, PB6, PB7, PC0	I_{OLH}	–	–	8	mA
Total output current (for I/O Pads)	$I_{OH} + I_{OL}$	–	–	40	mA
Input voltage threshold for OSC32A		0.2 x VDD_PADS	–	0.8 x VDD_PADS	V
Input voltage threshold for OSCA		0.2 x VDD_PADSA	–	0.8 x VDD_PADSA	V

14.8 Non-RF system electrical characteristics

Table 152 lists the non-RF system level characteristics for the STM32W.

Table 152. Non-RF system electrical characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
System wakeup time from deep sleep	From wakeup event to first ARM® Cortex-M3 instruction running from 6MHz internal RC clock Includes supply ramp time and oscillator startup time	–	110	–	µs
Shutdown time going into deep sleep	From last ARM® Cortex-M3 instruction to deep sleep mode	–	5	–	µs

14.9 RF electrical characteristics

14.9.1 Receive

Table 153 lists the key parameters of the integrated IEEE 802.15.4 receiver on the STM32W.

Note: Receive measurements were collected with ST's STM32W Ceramic Balun Reference Design (Version A0) at 2440 MHz. The Typical number indicates one standard deviation above the mean, measured at room temperature (25°C). The Min and Max numbers were measured over process corners at room temperature

Table 153. Receive characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
Frequency range		2400	–	2500	MHz
Sensitivity (boost mode)	1% PER, 20 byte packet defined by IEEE 802.15.4-2003	–	-102	-96	dBm
Sensitivity	1% PER, 20 byte packet defined by IEEE 802.15.4-2003	–	-100	-94	dBm
High-side adjacent channel rejection	IEEE 802.15.4 signal at -82 dBm	–	35	–	dB
Low-side adjacent channel rejection	IEEE 802.15.4 signal at -82 dBm	–	35	–	dB
2nd high-side adjacent channel rejection	IEEE 802.15.4 signal at -82 dBm	–	46	–	dB
2nd low-side adjacent channel rejection	IEEE 802.15.4 signal at -82 dBm	–	46	–	dB
Channel rejection for all other channels	IEEE 802.15.4 signal at -82 dBm	–	40	–	dB
802.11g rejection centered at +12 MHz or -13 MHz	IEEE 802.15.4 signal at -82 dBm	–	36	–	dB
Maximum input signal level for correct operation		0	–	–	dBm
Co-channel rejection	IEEE 802.15.4 signal at -82 dBm	–	-6	–	dBc
Relative frequency error (2x40 ppm required by IEEE 802.15.4)		-120	–	+120	ppm
Relative timing error (2x40 ppm required by IEEE 802.15.4)		-120	–	+120	ppm
Linear RSSI range	As defined by IEEE 802.15.4	40	–	–	dB
RSSI Range		-90	–	-40	dBm

14.9.2 Transmit

Table 154 lists the key parameters of the integrated IEEE 802.15.4 transmitter on the STM32W.

Note: Transmit measurements were collected with ST's STM32W Ceramic Balun Reference Design (Version A0) at 2440 MHz. The Typical number indicates one standard deviation above the mean, measured at room temperature (25°C). The Min and Max numbers were measured over process corners at room temperature

Table 154. Transmit characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
Maximum output power (boost mode)	At highest power setting	–	8	–	dBm
Maximum output power	At highest power setting	1	5	–	dBm
Minimum output power	At lowest power setting	–	-55	–	dBm
Error vector magnitude	As defined by IEEE 802.15.4, which sets a 35% maximum	–	5	15	%
Carrier frequency error		-40	–	+40	ppm
PSD mask relative	3.5 MHz away	-20	–	–	dB
PSD mask absolute	3.5 MHz away	-30	–	–	dBm

14.9.3 Synthesizer

Table 155 lists the key parameters of the integrated synthesizer on the STM32W.

Table 155. Synthesizer characteristics

Parameter	Conditions	Min.	Typ.	Max.	Unit
Frequency range		2400	–	2500	MHz
Frequency resolution		–	11.7	–	kHz
Lock time	From off, with correct VCO DAC setting	–	–	100	µs
Relock time	Channel change or RX/TX turnaround (IEEE 802.15.4 defines 192 µs turnaround time)	–	–	100	µs
Phase noise at 100 kHz offset		–	-71	–	dBc/Hz
Phase noise at 1 MHz offset		–	-91	–	dBc/Hz
Phase noise at 4 MHz offset		–	-103	–	dBc/Hz
Phase noise at 10 MHz offset		–	-111	–	dBc/Hz

15 Package characteristics

In order to meet environmental requirements, ST offers these devices in different grades of ECOPACK® packages, depending on their level of environmental compliance. ECOPACK® specifications, grade definitions and product status are available at: *www.st.com*. ECOPACK® is an ST trademark.

Figure 58. VFQFPN48 7x7mm package outline

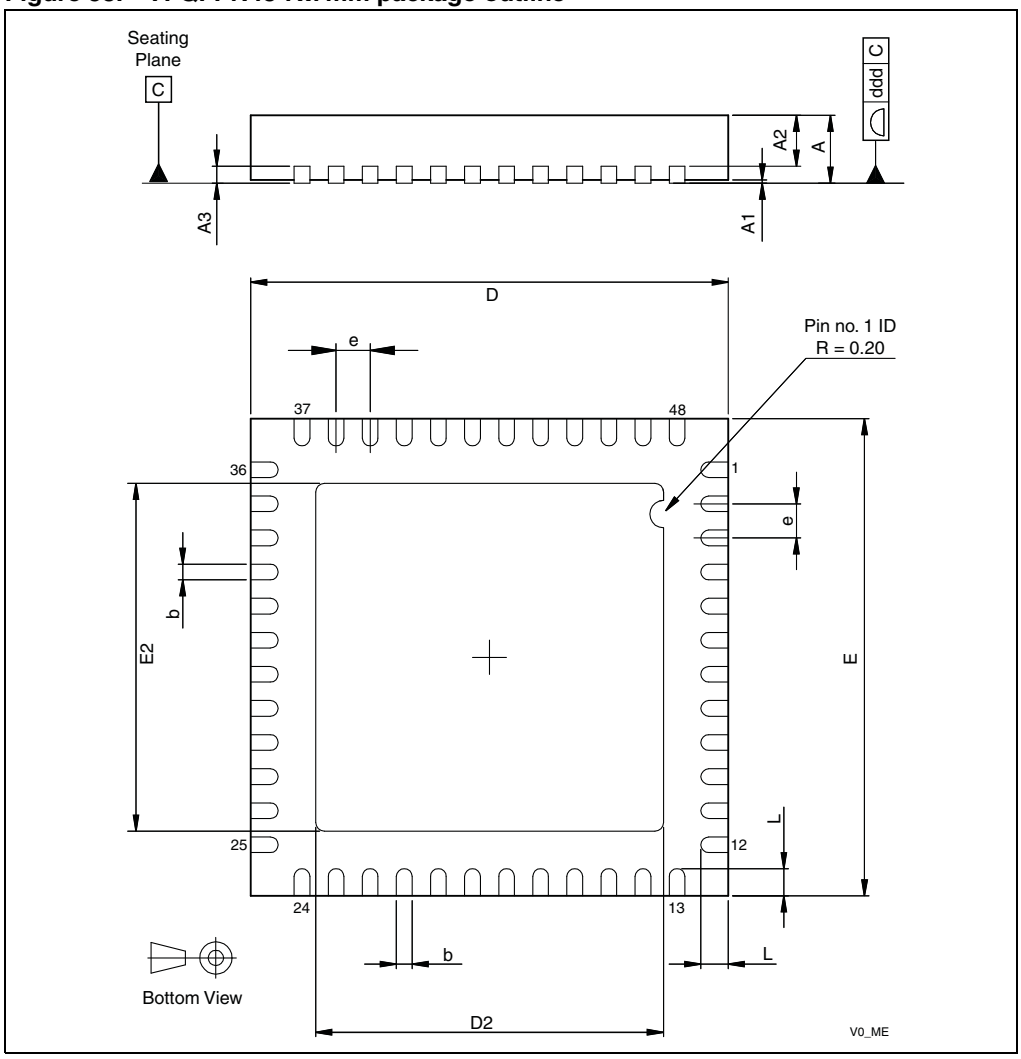
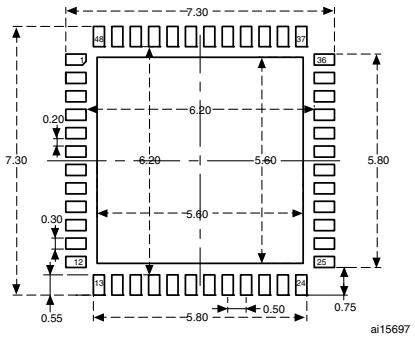


Table 156. VFQFPN48 7x7mm package mechanical data

Symbol	Millimeters			Inches[1]		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	0.800	0.900	1.000	0.0315	0.0354	0.0394
A1		0.020	0.050		0.0008	0.0020
A2		0.650	1.000		0.0256	0.0394
A3		0.250			0.0098	
b	0.180	0.230	0.300	0.0071	0.0091	0.0118
D	6.850	7.000	7.150	0.2697	0.2756	0.2815
D2	2.250	4.700	5.250	0.0886	0.1850	0.2067
E	6.850	7.000	7.150	0.2697	0.2756	0.2815
E2	2.250	4.700	5.250	0.0886	0.1850	0.2067
e	0.450	0.500	0.550	0.0177	0.0197	0.0217
L	0.300	0.400	0.500	0.0118	0.0157	0.0197
ddd			0.080			0.0031

1. Values in inches are converted from mm and rounded to 4 decimal digits.

Figure 59. Recommended footprint (dimensions in mm)[1]



1. Drawing is not to scale

Figure 60. VFQFPN40 6x6mm pitch 0.5 package outline

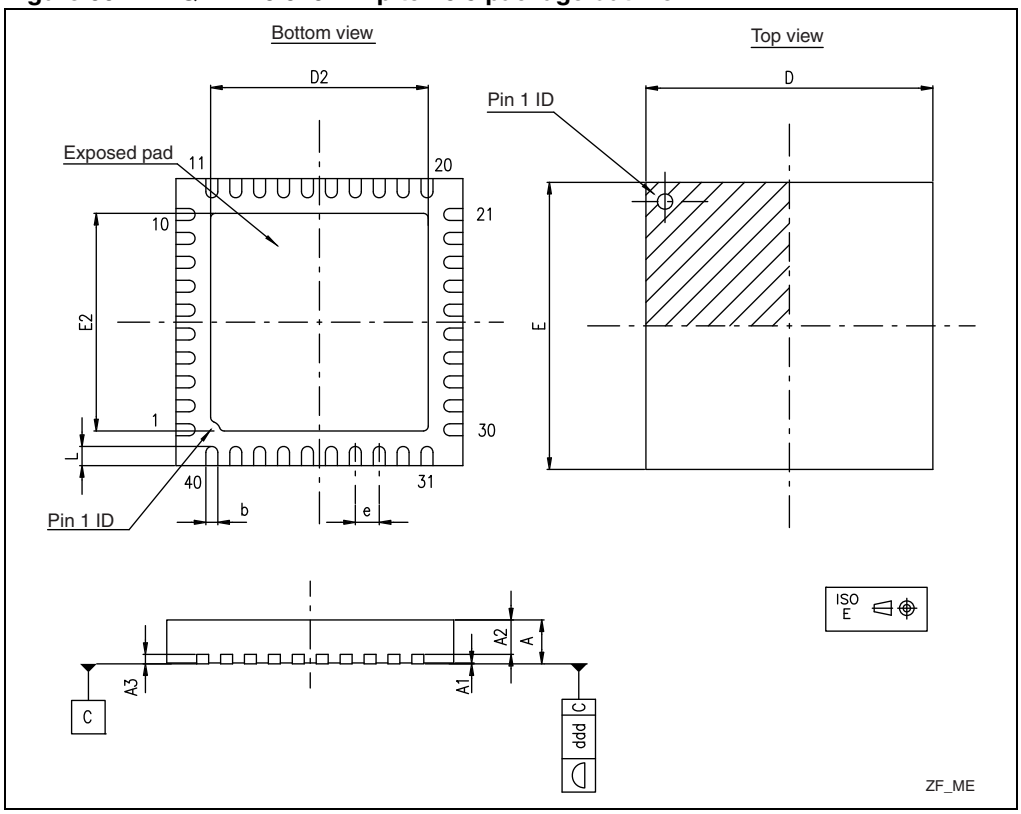


Table 157. VFQFPN40 6x6mm package mechanical data

Symbol	millimeters			inches[1]		
	Min	Typ	Max	Min	Typ	Max
A	0.800	0.900	1.000	0.0315	0.0354	0.0394
A1		0.020	0.050		0.0008	0.0020
A2		0.720	1.070		0.0283	0.0421
A3		0.200			0.0079	
b	0.180	0.250	0.300	0.0071	0.0098	0.0118
D	5.900	6.000	6.100	0.2323	0.2362	0.2402
D2	4.500	4.550	4.700	0.1772	0.1791	0.1850
E		6.000			0.2362	
E2	4.500	4.550	4.700	0.1772	0.1791	0.1850
e		0.500			0.0197	
L	0.350	0.400	0.450	0.0138	0.0157	0.0177
ddd			0.080			0.0031

1. Values in inches are converted from mm and rounded to 4 decimal digits.

16 Ordering information scheme

Table 158.

Example: STM32 W 108 C B U 6 x

Device family

STM32 = ARM-based 32-bit microcontroller

Product type

W = wireless system-on-chip

Sub-family

108 = IEEE 802.15.4 specification

Pin count

H = 40 pins
C = 48 pins

Code size

B = 128 Kbytes of Flash memory

C= 256 Kbytes of Flash memory

Z=192 Kbytes of Flash memory

Package

U = QFN

Temperature range

6 = –40 °C to +85 °C

7= –40 °C to +105 °C

Enabled protocol stack

"Blank" = Development sample platform (1)

1 = Ember ZigBee stack

3 = RF4CE stack

4 = IEEE 802.15.4 media access control

1. This P/N is under specific ordering conditions. Please refer to your nearest ST sales office.

For a list of available options (speed, package, etc.) or for further information on any aspect of this device, please contact your nearest ST sales office.

17 Revision history

Table 159. Document revision history

Date	Revision	Changes
16-Sep-2009	1	Initial release.
21-Sep-2009	2	Modified document status to Preliminary Data.
24-Nov-2009	3	Major revision of *Section 1.2: Overview on page 12* and *Section 9: Serial interfaces on page 75*. Added *Section 2: Documentation conventions on page 14*, *Section 4: Embedded memory on page 27* and *Section 7: Integrated voltage regulator on page 56*.
01-Mar-2010	4	Updated *Section 11.1.8: Calibration on page 169*. Added notes to *Software reset on page 37*, *Section 9.6.1: Setup and configuration on page 88* and *Section 11: Analog-to-digital converter on page 163*. Updated parameters and values in *Table 135: General operating conditions on page 195*.
22-Sep-2010	5	Added *Section 13.1: STM32W108 JTAG TAP connection on page 192* and *Section 6.4.4: Slow timers (Watchdog and Sleeptimer) control and status registers on page 45*. Added Total Tx current values for 24 MHz in *Table 150: DC electrical characteristics on page 204* and updated *Figure 56: Transmit power consumption on page 207*.
20-Oct-2010	6	Updated *Table 142: ADC module key parameters for 1 MHz sampling* and *Table 143: ADC module key parameters for input buffer disabled and 6 MHz sampling*. Added *Table 144: ADC module key parameters for input buffer enabled and 6MHz sampling*. Modified system wakeup time from deep sleep in *Table 152: Non-RF system electrical characteristics*.
09-Dec-2010	7	Updated datasheet status to full datasheet.
29-Mar-2011	8	Added INT_SLEEPTMRFLAG, INT_SLEEPTMRCFG, SLEEPTMR_CLKEN.
27-Jul-2011	9	Update for STM32W108CC and STM32W108CZ

Index of registers

W