AG32 MCU

Reference Manual

1.1

contents

1 Device overview

1.1 Introduction

The AG32 family of 32-bit microcontrollers is designed to offer new degrees of freedom and rich compatible peripherals, and compatible pin and features to MCU users. AG32 product series offers supreme quality, stability, and exceptional pricing value.

1.1.1 System Overview

- RISC-V core with RV32IMAFC support
- Up to 1 Mbyte of Flash memory
- 128KB SRAM
- 16KB instruction cache

1.1.2 Clock, reset and supply management

- 3.0 V to 3.6 V application supply and I/Os
- POR, PDR
- 4-to-26 MHz crystal oscillator
- Internal 20MHz oscillator
- 32 kHz oscillator for RTC
- Internal 40 kHz oscillator

1.1.3 Low-power operation

- Sleep, Stop and Standby modes
- VBAT supply for RTC

1.1.4 ADC/DAC/CMP/DMA/Timers/GPIO

- 3×12-bit, 1.0 MSPS A/D converters: up to 16 channels and 3 MSPS in triple interleaved mode
- 2×10-bit D/A converters

- Two rail-to-rail analog comparators
- General-purpose DMA
- Advanced-control timers
- Up to 78 user I/O ports

1.1.5 Communication interfaces

- I2C interfaces
- UART interfaces
- SPI interfaces
- CAN interfaces

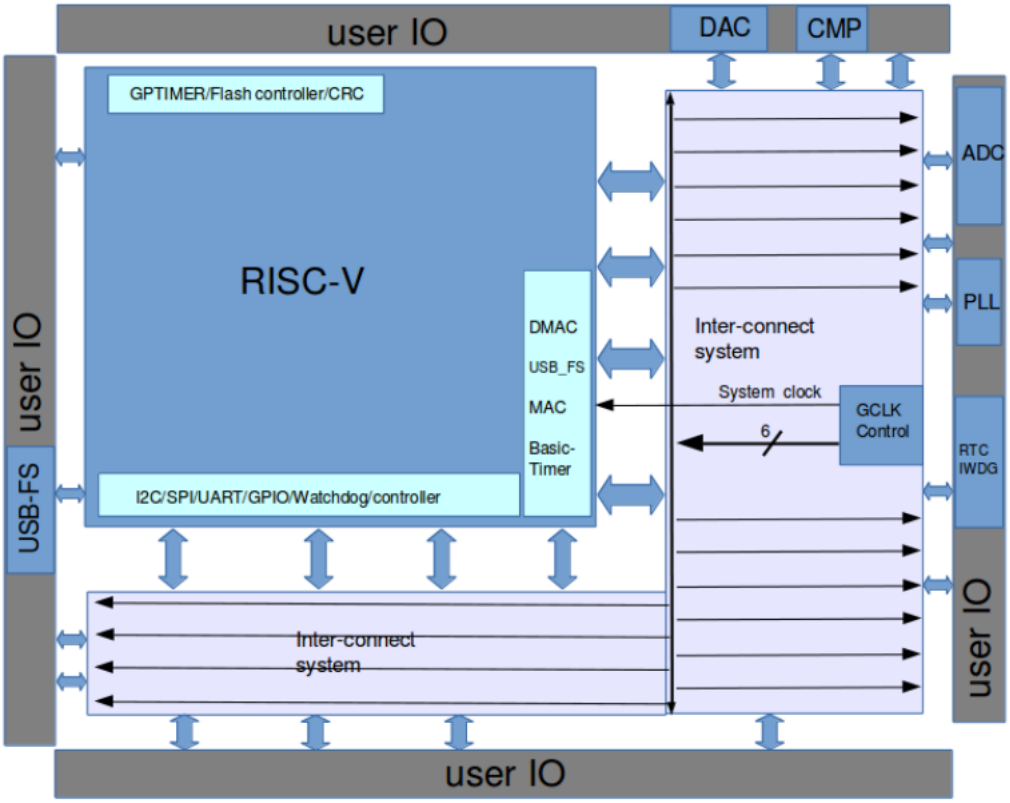
1.1.6 Others

- Debug mode – Serial wire debug (SWD) & JTAG interfaces
- USB 2.0 full-speed device/host controller with on-chip PHY
- 10/100 Ethernet MAC with dedicated DMA supports MII/RMII
- RTC: subsecond accuracy
- 128-bit unique ID

1.2 Features and peripheral counts

Peripherals	AG32VF303KCU6	AG32VF303CCT6	AG32VF303VCT6	AG32VF407RGT6	AG32VF407VGT6
Flash memory in Kbytes	256			1024	
SRAM in Kbytes	128				
Ethernet	yes				
Timers	Advanced-control timers				
SPI/I²C	yes				
UART	yes				
USB FS	yes				
CAN	yes				
12-bit ADC	3				
Number of channels	16				
10-bit DAC	2				
Number of channels					
rail-to-rail analog comparators	2				
Maximum CPU	200Mhz				
frequency					
Operating voltage	3.0 to 3.6 V				
Package	QFN32	LQFP48	LQFP100	LQFP64	LQFP100

1.3 Chip architecture



1.4 Memory Map

	Address	
ROM	0x0001 0000 - 0x0001 1FFF	
System Control	0x0300 0000 - 0x0300 0FFF	
PLIC	0x0C00 0000 - 0x0C20 FFFF	
SRAM	0x2000 0000 - 0x2001 FFFF	
FLASH (XIP)	0x8000 0000 - 0x80FF FFFF	
Option bytes	0x8100 0000 - 0x8100 003F	
RTC	0x4000 0000 - 0x4000 007F	
FLASH control	0x4000 1000 - 0x4000 1FFF	
APB Peripherals	0x4001 0000 - 0x40FF FFFF	
AHB Peripherals	0x4100 0000 - 0x41FF FFFF	
External AHB	0x6000 0000 - 0x7FFF FFFF	

1.5 System Control

Device boot mode (BOOT_MODE)

- Address offset: 0x00

31 - 2	1	0
Reserved	BOOT_MODE	
	RO	RO

- Bit [1:0]: Device boot mode

The values of BOOT0 and BOOT1 pins are latched on the 4th rising edge of SYSCLK after a reset

Reset control (RST_CNTL)

- Address offset: 0x04
- Bit 31 RSTF_LPWR: Reset flag by low power
  - 0: No reset detected

- 1: Low power reset detected

- Bit 30 RSTF_WDOG: Reset flag by watch dog
  - 0: No reset detected
  - 1: Watch dog reset detected
- Bit 29 RSTF_IWDG: Reset flag by independent watch dog
  - 0: No reset detected
  - 1: Independent watch dog reset detected
- Bit 28 RSTF_SFT: Reset flag by softare
  - 0: No reset detected
  - 1: Softare reset detected
- Bit 27 RSTF_POR: Reset flag by power on reset
  - 0: No reset detected
  - 1: Power on reset detected
- Bit 26 RSTF_PIN: Reset flag by NRST pin
  - 0: No reset detected
  - 1: NRST pin reset detected
- Bit 25 RSTF_EXT: Reset flag by external logic
  - 0: No reset detected
  - 1: External logic reset detected
- Bit 24 RST_REMOVE: Reset flag removal
  - Write 1 to clear all reset flags
- Bit 1 RST_EXT_EN: External logic reset enable

- 0: Exernal logic reset disabled
- 1: External logic reset enabled

- Bit 0 RST_SFT: Reset by software
  - Write 1 to trigger software reset

Power control (PWR_CNTL)

- Address offset: 0x08
- Bit [1:0] LPWR_MODE: Low power mode
  - 00: Enter sleep mode with WFI (wait for interrupt) instruction
  - 01: Enter stop mode with WFI instruction
  - 11: Enter standby mode with WFI instruction

Clock control (CLK_CNTL)

- Address offset: 0x0C
- Bit [15:12] SCLK_DIV_HIGH: Flash SPI clock divider high
  - Flash SPI clock is divided by (SCLK_DIV_HIGH + 1) from SYS_CLK, valid range is from 0 (divided by 1) to 15 (divided by 16)
- Bit [11:8] SCLK_DIV_LOW: Flash SPI clock divider low
  - Must be set to the same value as SCLK_DIV_HIGH
- Bit 6 PLL_RDY: PLL ready
  - 0: PLL is not ready
  - 1: PLL is ready
- Bit 5 PLL_ON: PLL on
  - 0: PLL is turned off
  - 1: PLL is turned on

- Bit 4 HSE_RDY: HSE ready
  - 0: HSE is not ready
  - 1: HSE is ready
- Bit 3 HSE_BYP: HSE bypass
  - 0: HSE oscillator is not bypassed
  - 1: HSE oscillator is bypassed
- Bit 2 HSE_ON: HSE on
  - 0: HSE oscillator is turned off
  - 1: HSE oscillator is turned on

JTAG control (SWJ_CNTL)

- Address offset: 0x14
- Bit 4: NJTRST: Configuration for pin NJTRST
  - 0: NJTRST is used as a dedicated pin
  - 1: NJTRST is used as a user pin
- Bit 3: JTDO: Configuration for pin JTDO
  - 0: JTDO is used as a dedicated pin
  - 1: JTDO is used as a user pin
- Bit 2: JTDI: Configuration for pin JTDI
  - 0: JTDI is used as a dedicated pin
  - 1: JTDI is used as a user pin
- Bit 1: JTMS: Configuration for pin JTMS
  - 0: JTMS is used as a dedicated pin
  - 1: JTMS is used as a user pin

- Bit 0: JTCK: Configuration for pin JTCK
  - 0: JTCK is used as a dedicated pin
  - 1: JTCK is used as a user pin

Debug control (DBG_CNTL)

- Address offset: 0x1C
- Bit 4 DBG_RTC_STOP: Stop RTC during debug
- Bit 3 DBG_IWDG_STOP: Stop IWDG during debug

Wake up rise triggers (WKP_RISE_TRG)

- Address offset: 0x20
- Bit [7:0] EXT_INT0-7: Wake up device from stop mode using EXT_INT0-7, rising edge triggered
- Bit 8 ALARM: Wake up device from stop mode using RTC alaram

Wake up fall triggers (WKP_FALL_TRG)

- Address offset: 0x24
- Bit [7:0] EXT_INT0-7: Wake up device from stop mode using EXT_INT0-7, falling edge triggered
- Bit 8 ALARM: Wake up device from stop mode using RTC alaram

Wake up pending register (WKP_PENDING)

- Address offset: 0x28
- Bits [8:0]: Correspoding bits are set when the selected triggering event occurs

PBUS clock divider (PBUS_DIVIDER)

- Address offset: 0x38
- Bits [3:0] PBUS_DIV: APB clock is divided by (PBUS_DIV + 1) from SYS_CLK, valid range is from 0 (divided by 1) to 15 (divided by 16)

APB peripheral reset (APB_RESET)

- Address offset: 0x40
- Each APB peripheral can be reset with the corresponding bit
  - 0: Reset is deasserted
  - 1: Reset is asserted
- Bit [28]: I2C1
- Bit [27]: I2C0
- Bit [26]: CAN0
- Bit [25]: UART4
- Bit [24]: UART3
- Bit [23]: UART2
- Bit [22]: UART1
- Bit [21]: UART0
- Bit [20]: GPTIMER4
- Bit [19]: GPTIMER3
- Bit [18]: GPTIMER2
- Bit [17]: GPTIMER1
- Bit [16]: GPTIMER0
- Bit [15]: TIMER1
- Bit [14]: TIMER0
- Bit [13]: GPIO9
- Bit [12]: GPIO8
- Bit [11]: GPIO7

- Bit [10]: GPIO6
- Bit [9]: GPIO5
- Bit [8]: GPIO4
- Bit [7]: GPIO3
- Bit [6]: GPIO2
- Bit [5]: GPIO1
- Bit [4]: GPIO0
- Bit [3]: SPI1
- Bit [2]: SPI0
- Bit [1]: WATCHDOG0
- Bit [0]: FCB0

AHB peripheral reset (AHB_RESET)

- Address offset: 0x50
- Each AHB peripheral can be reset with the corresponding bit
  - 0: Reset is deasserted
  - 1: Reset is asserted
- Bit [3]: MAC0
- Bit [2]: CRC0
- Bit [1]: USB0
- Bit [0]: DMAC0

APB peripheral clock enable (APB_CLKENABLE)

- Address offset: 0x60

- Clock must be enabled before any APB peripheral is accessed. Bit asssignment is the same as APB_RESET register
  - 0: Peripheral clock is disabled
  - 1: Peripheral clock is enabled

AHB peripheral clock enable (AHB_CLKENABLE)

- Address offset: 0x70
- Clock must be enabled before any AHB peripheral is accessed. Bit asssignment is the same as AHB_RESET register
  - 0: Peripheral clock is disabled
  - 1: Peripheral clock is enabled

APB peripheral clock stop during debug (APB_CLKSTOP)

- Address offset: 0x80
- Clock can be automatically stopped during debug for the following APB peripherals:
  - WATCHDOG
  - TIMER
  - GPTIMER
  - CAN
- Bit asssignment is the same as APB_RESET register
  - 0: Clock is not stopped during debug
  - 1: Clock is stopped during debug

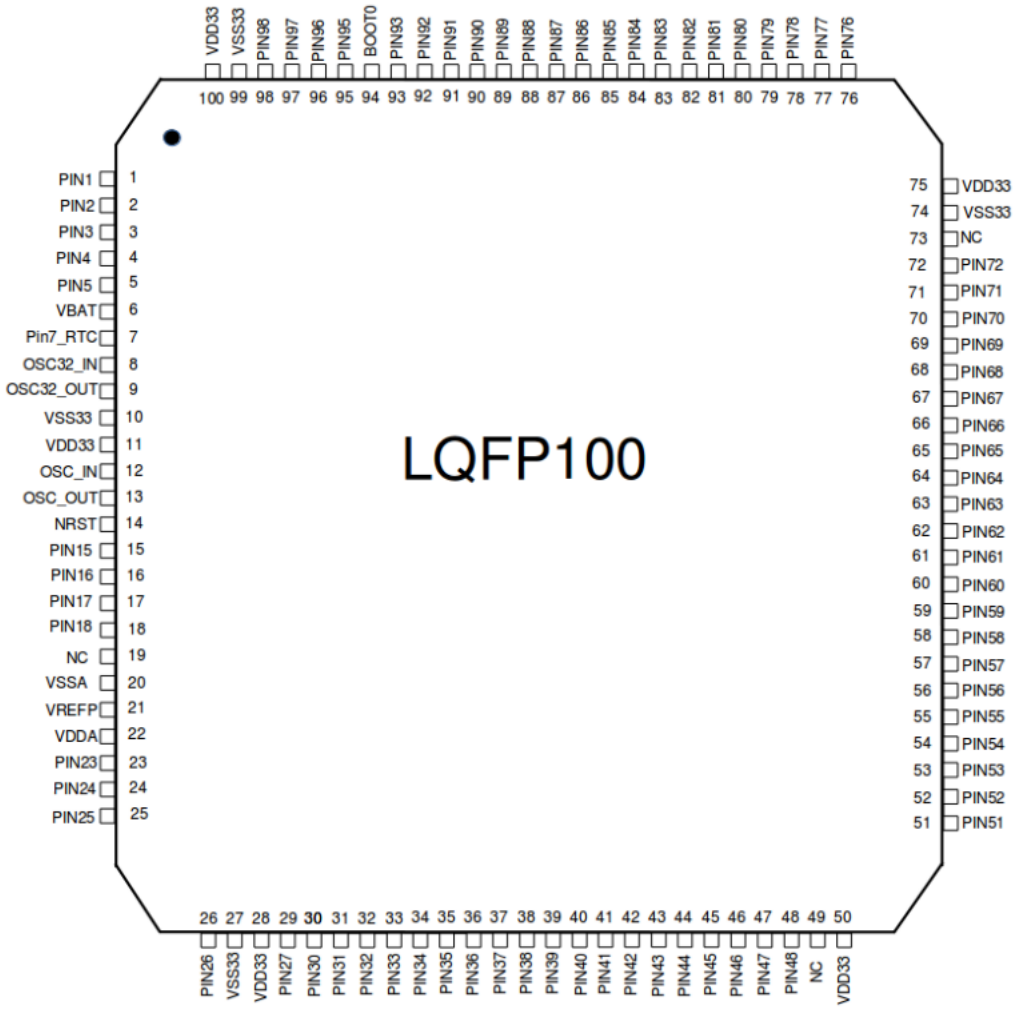
Device ID code (DEVICE_ID)

- Address offset: 0x100
- Bit [31:0]: Returns the chip device ID: 0x40200001. Read only

2 Pin Definition

LQFP-100

Figure 2. AG32VF103VCT6/AG32VF407VGT6 LQFP100 pinout

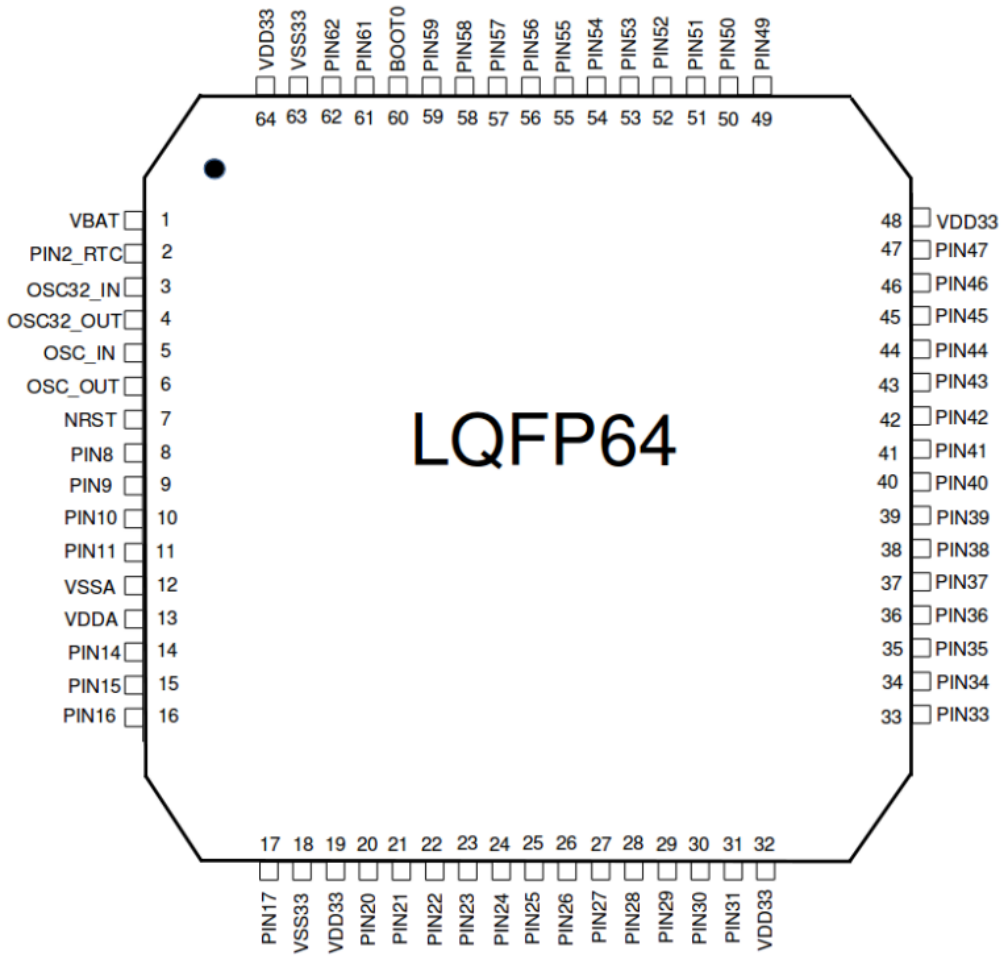


Pin	Pin name	Function		Pin	Pin name	Function
1	PIN_1	IO		26	PIN_26	IO/ADC_IN3/CMP_PA3
2	PIN_2	IO		27	VSS33	GND
3	PIN_3	IO		28	VDD33	VDD33
4	PIN_4	IO		29	PIN_29	IO/ADC_IN4/CMP_PA4/DAC0
5	PIN_5	IO		30	PIN_30	IO/ADC_IN5/CMP_PA5/DAC1
6	VBAT	VBAT		31	PIN_31	IO/ADC_IN6
7	PIN_7	IO/RTC		32	PIN_32	IO/ADC_IN7
8	OSC32_IN	OSC32_IN		33	PIN_33	IO/ADC_IN14
9	OSC32_OUT	OSC32_OUT		34	PIN_34	IO/ADC_IN15
10	VSS33	GND		35	PIN_35	IO/ADC_IN8
11	VDD33	VDD33		36	PIN_36	IO/ADC_IN9
12	OSC_IN	OSC_IN		37	PIN_37	IO/BOOT1
13	OSC_OUT	OSC_OUT		38	PIN_38	IO
14	NRST	NRST		39	PIN_39	IO
15	PIN_15	IO/ADC_IN10		40	PIN_40	IO
16	PIN_16	IO/ADC_IN11		41	PIN_41	IO
17	PIN_17	IO/ADC_IN12		42	PIN_42	IO
18	PIN_18	IO/ADC_IN13		43	PIN_43	IO
19	NC	NC		44	PIN_44	IO
20	VSSA	GNDA		45	PIN_45	IO
21	VREFP	VREFP		46	PIN_46	IO
22	VDDA	VDDA		47	PIN_47	IO
23	PIN_23	IO/WKUP/ADC_IN0/CMP_PA0		48	PIN_48	IO
24	PIN_24	IO/ADC_IN1/CMP_PA1		49	NC	NC
25	PIN_25	IO/ADC_IN2/CMP_PA2		50	VDD33	VDD33

Pin	Pin name	Function		Pin	Pin name	Function
51	PIN_51	IO		76	PIN_76	IO/JTCK
52	PIN_52	IO		77	PIN_77	IO/JTDI
53	PIN_53	IO		78	PIN_78	IO
54	PIN_54	IO		79	PIN_79	IO
55	PIN_55	IO		80	PIN_80	IO
56	PIN_56	IO		81	PIN_81	IO
57	PIN_57	IO		82	PIN_82	IO
58	PIN_58	IO		83	PIN_83	IO
59	PIN_59	IO		84	PIN_84	IO
60	PIN_60	IO		85	PIN_85	IO
61	PIN_61	IO		86	PIN_86	IO
62	PIN_62	IO		87	PIN_87	IO
63	PIN_63	IO		88	PIN_88	IO
64	PIN_64	IO		89	PIN_89	IO/JTDO
65	PIN_65	IO		90	PIN_90	IO/JNTRST
66	PIN_66	IO		91	PIN_91	IO
67	PIN_67	IO		92	PIN_92	IO
68	PIN_68	IO/UART0_TX		93	PIN_93	IO
69	PIN_69	IO/UART0_RX		94	BOOT0	BOOT0
70	PIN_70	IO/USBDM		95	PIN_95	IO
71	PIN_71	IO/USBDP		96	PIN_96	IO
72	PIN_72	IO/JTMS		97	PIN_97	IO
73	NC	NC		98	PIN_98	IO
74	VSS33	GND		99	VSS33	GND
75	VDD33	VDD33		100	VDD33	VDD33

LQFP-64

Figure 1. AG32VF103RCT6/AG32VF407RGT6 LQFP64 pinout



Pin	Pin name	Function		Pin	Pin name	Function
1	VBAT	VBAT		33	PIN_33	IO
2	PIN_2	IO/RTC		34	PIN_34	IO
3	OSC32_IN	OSC32_IN		35	PIN_35	IO
4	OSC32_OUT	OSC32_OUT		36	PIN_36	IO
5	OSC_IN	OSC_IN		37	PIN_37	IO
6	OSC_OUT	OSC_OUT		38	PIN_38	IO
7	NRST	NRST		39	PIN_39	IO
8	PIN_8	IO/ADC_IN10		40	PIN_40	IO
9	PIN_9	IO/ADC_IN11		41	PIN_41	IO
10	PIN_10	IO/ADC_IN12		42	PIN_42	IO/UART0_TX
11	PIN_11	IO/ADC_IN13		43	PIN_43	IO/UART0_RX
12	VSSA	GNDA		44	PIN_44	IO/USBDM
13	VDDA	VDDA		45	PIN_45	IO/USBDP
14	PIN_14	IO/WKUP/ADC_IN0/CMP_PA0		46	PIN_46	IO/JTMS
15	PIN_15	IO/ADC_IN1/CMP_PA1		47	PIN_47	IO
16	PIN_16	IO/ADC_IN2/CMP_PA2		48	VDD33	VDD33
17	PIN_17	IO/ADC_IN3/CMP_PA3		49	PIN_49	IO/JTCK
18	VSS33	GND		50	PIN_50	IO/JTDI
19	VDD33	VDD33		51	PIN_51	IO
20	PIN_20	IO/ADC_IN4/CMP_PA4/DAC0		52	PIN_52	IO
21	PIN_21	IO/ADC_IN5/CMP_PA5/DAC1		53	PIN_53	IO
22	PIN_22	IO/ADC_IN6		54	PIN_54	IO
23	PIN_23	IO/ADC_IN7		55	PIN_55	IO/JTDO
24	PIN_24	IO/ADC_IN14		56	PIN_56	IO/JNTRST
25	PIN_25	IO/ADC_IN15		57	PIN_57	IO
26	PIN_26	IO/ADC_IN8		58	PIN_58	IO
27	PIN_27	IO/ADC_IN9		59	PIN_59	IO
28	PIN_28	IO/BOOT1		60	BOOT0	BOOT0
29	PIN_29	IO		61	PIN_61	IO
30	PIN_30	IO		62	PIN_62	IO
31	PIN_31	IO		63	VSS33	GND
32	VDD33	VDD33		64	VDD33	VDD33

LQFP-48

Pin	Pin name	Function		Pin	Pin name	Function
1	VBAT	VBAT		25	PIN_25	IO
2	PIN_2	IO/RTC		26	PIN_26	IO
3	OSC32_IN	OSC32_IN		27	PIN_27	IO
4	OSC32_OUT	OSC32_OUT		28	PIN_28	IO
5	OSC_IN	OSC_IN		29	PIN_29	IO
6	OSC_OUT	OSC_OUT		30	PIN_30	IO/UART0_TX
7	NRST	NRST		31	PIN_31	IO/UART0_RX
8	VSSA	GNDA		32	PIN_32	IO/USBDM
9	VDDA	VDDA		33	PIN_33	IO/USBDP
10	PIN_10	IO/WKUP/ADC_IN0/CMP_PA0		34	PIN_34	IO/JTMS
11	PIN_11	IO/ADC_IN1/CMP_PA1		35	PIN_35	IO
12	PIN_12	IO/ADC_IN2/CMP_PA2		36	VDD33	VDD33
13	PIN_13	IO/ADC_IN3/CMP_PA3		37	PIN_37	IO/JTCK
14	PIN_14	IO/ADC_IN4/CMP_PA4/DAC0		38	PIN_38	IO/JTDI
15	PIN_15	IO/ADC_IN5/CMP_PA5/DAC1		39	PIN_39	IO/JTDO
16	PIN_16	IO/ADC_IN6		40	PIN_40	IO/JNTRST
17	PIN_17	IO/ADC_IN7		41	PIN_41	IO
18	PIN_18	IO/ADC_IN8		42	PIN_42	IO
19	PIN_19	IO/ADC_IN9		43	PIN_43	IO
20	PIN_20	IO/BOOT1		44	BOOT0	BOOT0
21	PIN_21	IO		45	PIN_45	IO
22	PIN_22	IO		46	PIN_46	IO
23	VSS33	GND		47	VSS33	GND
24	VDD33	VDD33		48	VDD33	VDD33

3 Clock

3.1 Clock sources

Three different clock sources can be used to drive the system clock (SYSCLK):

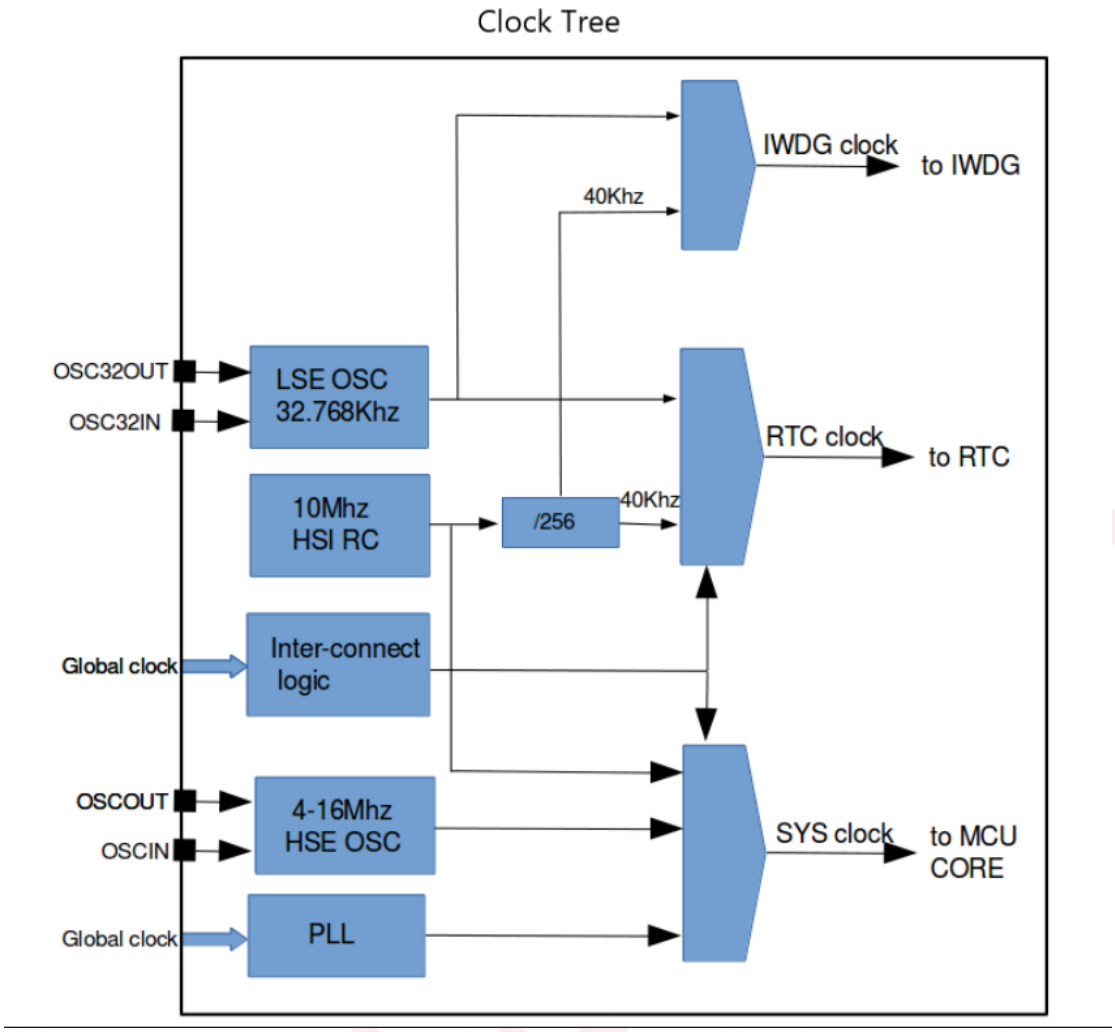
(1) HSI oscillator clock

(2) HSE oscillator clock

(3) PLL clock

(4) Interconnect global clocks(FPGA Core)

The devices have the following two secondary clock sources:
(1) 40 kHz low speed internal RC (LSI), which drives the independent watchdog and optionally the RTC used for Auto-wakeup from Stop/Standby mode.
(2) 32.768 kHz low speed external crystal (LSE crystal), which optionally drives the real-time clock (RTCCLK)
Each clock source can be switched on or off independently when it is not used, to optimize power consumption.



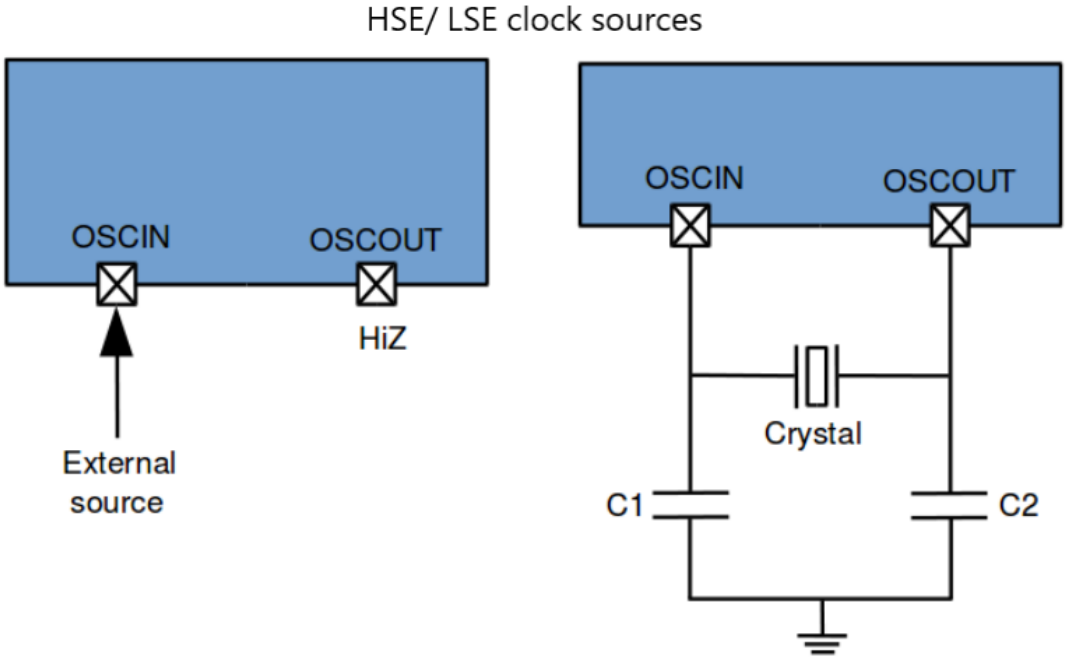
3.2 HSE clock

The high speed external clock signal (HSE) can be generated from two possible clock sources:

(1) HSE external crystal/ceramic resonator

(2) HSE user external clock

The resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected oscillator.



External source (HSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 100 MHz. You select this mode by setting the HSEBYP and HSEON bits in the Clock control register (RCC_CR). The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC_IN pin while the OSC_OUT pin should be left hi-Z.

External crystal/ceramic resonator (HSE crystal)

The 4 to 24 MHz external oscillator has the advantage of producing a very accurate rate on the main clock. The HSERDY flag indicates if the high-speed external oscillator is stable or not. At startup, the clock is not released until this bit is set by hardware. The HSE Crystal can be switched on and off using the HSEON bit.

3.3 HSI clock

The HSI clock signal is generated from an internal Oscillator and can be used directly as a system clock. The HSI internal oscillator has the advantage of providing a clock source at low cost (no external components). It also has a faster startup time than the HSE crystal oscillator.

3.4 PLL clock

The internal PLL can be used to multiply HSE crystal output clock frequency.

If the USB interface is used in the application, the PLL must be programmed to output 48 MHz. This is needed to provide a 48 MHz USBCLK.

3.5 LSE clock

The LSE crystal is a 32.768 kHz Low Speed External crystal or ceramic resonator. It has the advantage providing a low-power but highly accurate clock source to the real-time clock peripheral (RTC) for clock/calendar or other timing functions.

The LSE crystal is switched on and off using the LSEON bit in Backup domain control register.

The LSERDY flag in the Backup domain control register indicates if the LSE crystal is stable or not. At startup, the LSE crystal output clock signal is not released until this bit is set by hardware.

External source (LSE bypass)

In this mode, an external clock source must be provided. It can have a frequency of up to 1MHz. The external clock signal (square, sinus or triangle) with ~50% duty cycle has to drive the OSC32_IN pin while the OSC32_OUT pin should be left Hi-Z.

3.6 LSI clock

The LSI clock is HSI divided by 256. It can be kept running in Stop mode for the independent watchdog (IWDG) and Auto-wakeup unit. The clock frequency is around 40 kHz (between 30 kHz and 60 kHz).

3.7 System clock (SYSCLK) selection

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or through the PLL as system clock, it is not possible to stop it. A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked).

A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked).

3.8 RTC clock

The RTCCLK clock source can be either the CLKLOCAL(from fpga core logic), LSE or LSI clocks. This is selected by programming the RTCSEL[1:0] bits in the Backup domain control register (RCC_BDCR). This selection cannot be modified without resetting the Backup domain.
The LSE clock is in the Backup domain, whereas the HSE and LSI clocks are not.

Consequently:

(1) If LSE is selected as RTC clock:

The RTC continues to work even if the VDD supply is switched off, provided the VBAT supply is maintained.

(2) If LSI is selected as Auto-Wakeup unit (AWU) clock:

The AWU state is not guaranteed if the VDD supply is powered off.

(3) If the CLKLOCAL is used as the RTC clock:

The RTC state is not guaranteed if the VDD supply is powered off or if the internal voltage regulator is powered off (removing power from the 1.2 V domain).

The DPB bit (disable backup domain write protection) in the Power controller register must be set to 1.

3.9 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access,

(1) Under run or stop mode

Select LSE or LSI clock source by setting the IWDG_STOP_CLKSEL bit in the Backup domain control register (RCC_BDCR).

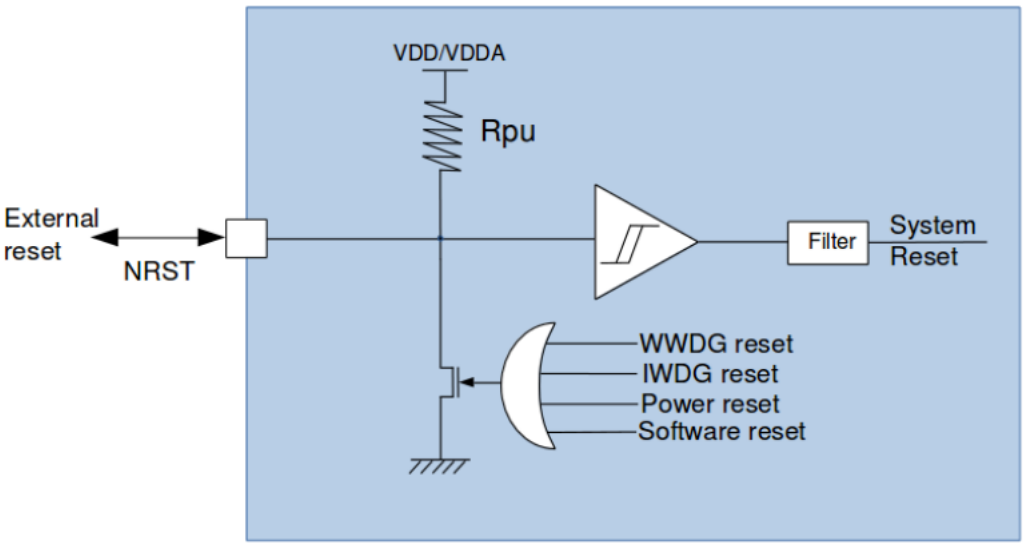
(2) Under Standby mode

HW will select LSE as clock source for IWDG.

4 Reset

There are three types of reset: system reset, power reset and backup domain reset.

Simplified diagram of the reset circuit



4.1 System reset

A system reset is generated when one of the following events occurs:

(1) A low level on the NRST pin (external reset)

(2) Window watchdog end of count condition (WWDG reset)

(3) Independent watchdog end of count condition (IWDG reset)

(4) A software reset (SW reset)

(5) Low-power management reset

The reset source can be identified by checking the reset flags in the Control/Status register, RCC_CSR.

Software reset

The SYSRESETREQ bit in MCU Application Interrupt and Reset Control Register must be set to force a software reset on the device.

Low-power management reset

There are two ways to generate a low-power management reset:

(1) Reset generated when entering Standby mode:

This type of reset is enabled by resetting nRST_STDBY bit in User Option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.

(2) Reset when entering Stop mode:

This type of reset is enabled by resetting nRST_STOP bit in User Option Bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

4.2 Power reset

A power reset is generated when one of the following events occurs:

(1) Power-on/power-down reset (POR/PDR reset)

(2) When exiting Standby mode

4.3 Backup domain reset

The backup domain has two specific resets that affect only the backup domain.

A backup domain reset is generated when one of the following events occurs:

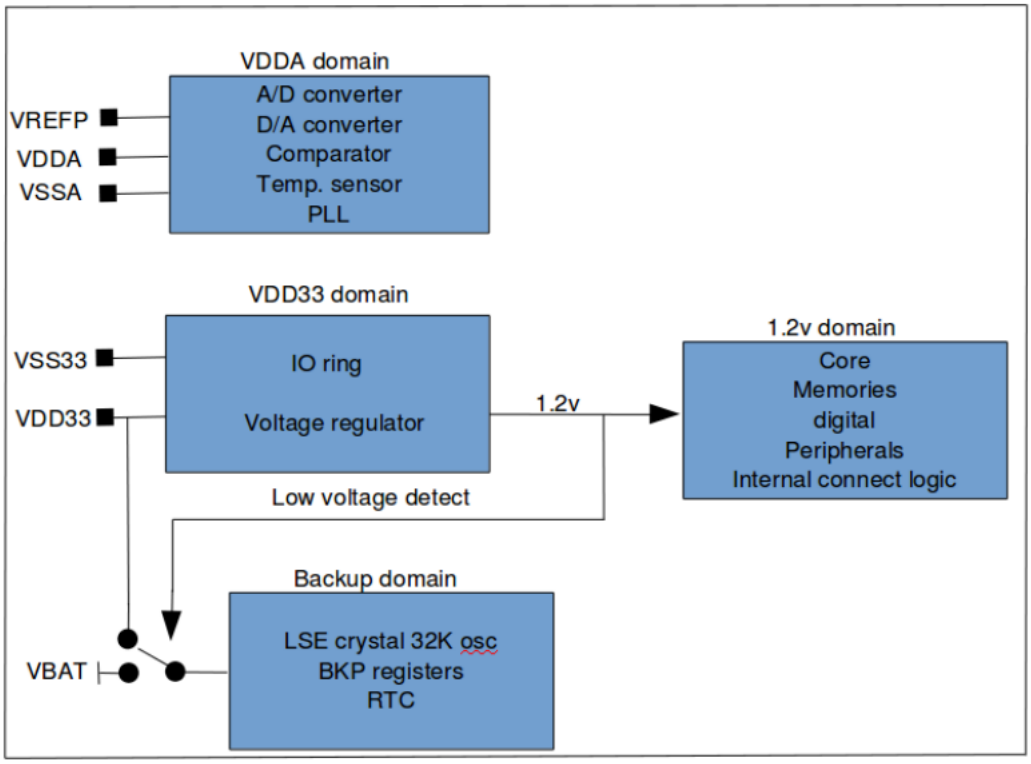
(1)Software reset.

(2) VDD33 or VBAT power on, if both supplies have previously been powered off.

5 Power control

5.1 Power supplies

The AG32VF requires a 3.0-to-3.6V operating voltage supply (VDD33). An embedded regulator is used to supply the internal 1.2V digital power. The real-time clock (RTC) and backup registers can be powered from the VBAT voltage when the main VDD33 supply is powered off.



5.2 Independent ADC and DAC converter supply and reference voltage

To improve conversion accuracy, the ADC and the DAC have an independent power supply which can be separately filtered and shielded from noise on the PCB.

(1) The ADC and DAC voltage supply input is available on a separate VDDA pin.

(2) An isolated supply ground connection is provided on pin VSSA.

To ensure a better accuracy on low-voltage inputs and outputs, the user can connect a separate external reference voltage on VREFP. VREFP is the highest voltage, represented by the full scale

value, for an analog input (ADC) or output (DAC) signal. The voltage on VREFP can range from 3.0V to VDDA.

5.3 Battery backup domain

To retain the content of the Backup registers and supply the RTC function when VDD33 is turned off, VBAT pin can be connected to an optional standby voltage supplied by a battery or by another source.

The VBAT pin powers the RTC unit, the LSE oscillator and the OSC32_IN and OSC32_OUT Pins, allowing the RTC to operate even when the main digital supply (VDD33) is turned off.

If no external battery is used in the application, it is recommended to connect VBAT externally to VDD33 with a 100nF external ceramic decoupling capacitor.

When the backup domain is supplied by VDD33 (analog switch connected to VDD33).

5.4 Voltage regulator

The voltage regulator is always enabled after Reset. It works in two different modes depending on the application modes.

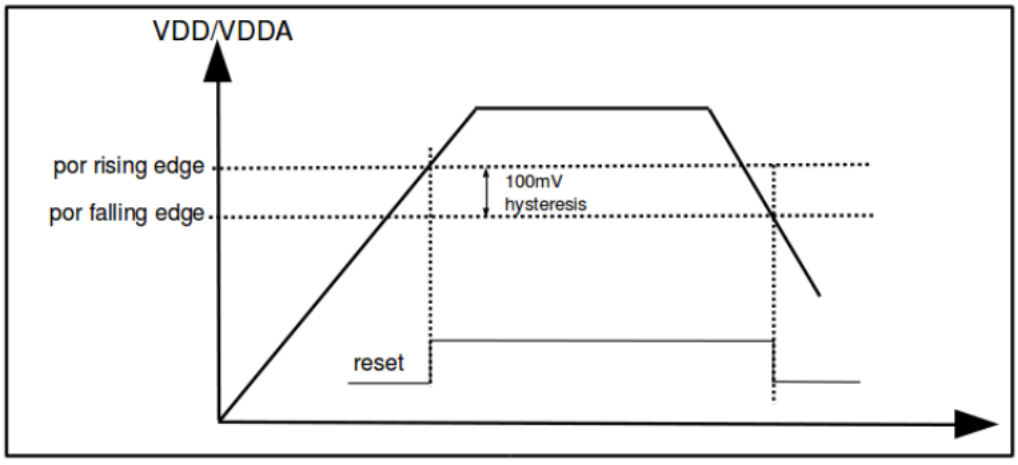
(1) In Run and Stop modes, the regulator supplies full power to the 1.2V domain (core, memories, digital peripherals and interconnect logic).

(2) In Standby Mode, the regulator is powered off. The contents of the registers and SRAM are lost except for the Standby circuitry and the Backup Domain.

5.5 Power on reset (POR)/power down reset (PDR)

The device has an integrated POR/PDR circuitry that allows proper operation starting from/down to 2.2V. The device remains in Reset mode when VDD33/VDDA is below a specified threshold, VPOR/PDR, without the need for an external reset circuit.

Power on reset/power down reset waveform



5.6 Low-power modes

By default, the micro-controller is in Run mode after a system or a power Reset. Several low-power modes are available to save power when the CPU does not need to be kept running. It is up to the user to select the mode that gives the best compromise between low-power consumption, short startup time and available wakeup sources.

The AG32VF devices feature three low-power modes:

(1) Sleep mode (CPU clock off, all peripherals including core peripherals are kept running)

(2) Stop mode (all clocks are stopped)

(3) Standby mode (1.2V domain powered-off)

In addition, the power consumption in Run mode can be reduce by one of the following means:

(1) Slowing down the system clocks.

(2) Gating the clocks to the APB and AHB peripherals when they are unused.

5.6.1 Slowing down system clocks

In Run mode the speed of the system clocks can be reduced. And also slow down peripherals before entering Sleep mode.

5.6.2 Peripheral clock gating

In Run mode, the clocks for individual peripherals and memories can be stopped at any time to reduce power consumption.

To further reduce power consumption in Sleep mode the peripheral clocks can be disabled prior to executing the WFI or WFE instructions.

5.6.3 Sleep mode

Entering Sleep mode

The Sleep mode is entered by executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions. Two options are available to select the Sleep mode entry mechanism, depending on the SLEEPONEXIT bit in the System Control register:

(1) Sleep-now: if the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.

(2) Sleep-on-exit: if the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

In the Sleep mode, all I/O pins keep the same state as in the Run mode.

Exiting Sleep mode

If the WFI instruction is used to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode. If the WFE instruction is used to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated either by:

(1) enabling an interrupt in the peripheral control register but not in the NVIC, and enabling the SEVONPEND bit in the System Control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

(2) or configuring an external or internal EXTI line in event mode. When the CPU resumes from WFE, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bit corresponding to the event line is not set. This mode offers the lowest wakeup time as no time is wasted in interrupt entry/exit.

Table 1. Sleep-now

Sleep-now mode	**Description**
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 0 Refer to the System Control register.
Mode exit	If WFI was used for entry: Interrupt: Refer to : Interrupt and exception vectors If WFE was used for entry Wakeup event: Refer to : Wakeup event management
Wakeup latency	None

Table 2. Sleep-on-exit

Sleep-on-exit	**Description**
Mode entry	WFI (wait for interrupt) while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 Refer to the System Control register.
Mode exit	Interrupt: refer to: Interrupt and exception vectors.
Wakeup latency	None

5.6.4 Stop mode

The Stop mode is based on the MCU deep-sleep mode combined with peripheral clock gating.

In Stop mode, all clocks in the 1.2V domain are stopped, the PLL, the HSI and the HSE oscillators are disabled. SRAM and register contents are preserved.

In the Stop mode, all I/O pins keep the same state as in the Run mode.

- **Entering Stop mode**

Refer to Table 3 for details on how to enter the Stop mode.

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the APB domain is ongoing, The Stop mode entry is delayed until the APB access is finished.

In Stop mode, the following features can be selected by programming individual control bits:

(1) Independent watchdog (IWDG): the IWDG is started by writing to its enable register or by hardware option. Once started it cannot be stopped except by a Reset.

(2) Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR).

(3) External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR).

The ADC or DAC can also consume power during the Stop mode, unless they are disabled before entering it.

- **Exiting Stop mode**

Refer to Table 3 for more details on how to exit Stop mode.

When exiting Stop mode by issuing an interrupt or a wakeup event, the HSI RC oscillator is selected as system clock.

Table 3. Stop mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP bit in System Control register – Clear PDDS bit in Power Control register (PWR_CR) Note: To enter Stop mode, all EXTI Line pending bits (in Pending register (EXTI_PR)), all peripheral interrupt pending bits, and RTC Alarm flag must be reset. Otherwise, the Stop mode entry procedure is ignored and program execution continues.
Mode exit	If WFI was used for entry: Any EXTI Line configured in Interrupt mode (the corresponding EXTI Interrupt vector must be enabled in the NVIC). Refer to: Interrupt and exception vectors. If WFE was used for entry: Any EXTI Line configured in event mode. Refer to: Wakeup event management
Wakeup latency	HSI RC wakeup time

5.6.5 Standby mode

The Standby mode allows to achieve the lowest power consumption. It is based on the deep-sleep mode, with the voltage regulator disabled. The 1.2V domain is consequently powered off. The PLL, the HSI oscillator and the HSE oscillator are also switched off. SRAM and register contents are lost except for registers in the Backup domain and Standby circuitry.

- **Entering Standby mode**

Refer to Table 4 for more details on how to enter Standby mode.

In Standby mode, the following features can be selected by programming individual control bits:

(1) Independent watchdog (IWDG): the IWDG is started by writing to its enable register or by hardware option. Once started it cannot be stopped except by a reset.

(2) Real-time clock (RTC): this is configured by the RTCEN bit in the Backup domain control register (RCC_BDCR).

(3) External 32.768 kHz oscillator (LSE OSC): this is configured by the LSEON bit in the Backup domain control register (RCC_BDCR)

- **Exiting Standby mode**

The micro-controller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge or falling edge on the WKUP pin or the rising edge of an RTC alarm occurs. All registers are reset after wakeup from Standby.

After waking up from Standby mode, program execution restarts in the same way as after a Reset. The SBF status flag in the Power control/status register (PWR_CSR) indicates that the MCU was in Standby mode.

Refer to Table 4 for more details on how to exit Standby mode.

Table 4. Standby mode

Stop mode	Description
Mode entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: – Set SLEEPDEEP in System Control register – Set PDDS bit in Power Control register (PWR_CR) – Clear WUF bit in Power Control/Status register (PWR_CSR) – No interrupt (for WFI) or event (for WFI) is pending
Mode exit	WKUP pin rising edge, RTC alarm event's rising edge, external Reset in NRST pin, IWDG Reset.
Wakeup latency	Reset phase

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance except:

(1) Reset pin (still available)

(2) CLKRTCOUT pin if configured for calibration out

(3) WKUP pin, if enabled

5.6.6 Auto-wakeup (AWU) from low-power mode

The RTC can be used to wakeup the MCU from low-power mode without depending on an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. For this purpose, two of the three alternative RTC clock sources can be selected by programming the RTCSEL[1:0] bits in the Backup domain control register (RCC_BDCR):

(1) Low-power 32.768 kHz external crystal oscillator (LSE OSC).

This clock source provides a precise time base with very low-power consumption.

(2) Low-power internal RC Oscillator (LSI RC)

This clock source has the advantage of saving the cost of the 32.768 kHz crystal.

6 Interrupt Controller

The AG32 device embed a nested vectored interrupt controller able to manage 16 priority levels, and handle up to 44 maskable interrupt channels plus the 16 interrupt lines of the RISC-V core.

6.1 Local interrupts

4 local interrupts (LOCAL_INT0-3) are connected directly to the core and have lower latencies. They have fixed priorities.

6.2 External interrupts

External interrupts are routed through the Platform-Level Interrupt Controller (PLIC). They have programmable priority levels and a threshold. The interrupt numbers are listed below:

Interrupt Name	Interrupt Number	Comment
FLASH	1	
RTC	2	
FCB0	3	
WATCHDOG0	4	
SPI0	5	
SPI1	6	
GPIO0	7	
GPIO1	8	
GPIO2	9	
GPIO3	10	
GPIO4	11	
GPIO5	12	
GPIO6	13	

GPIO7	14	
GPIO8	15	
GPIO9	16	
TIMER0	17	
TIMER1	18	
GPTIMER0	19	
GPTIMER1	20	
GPTIMER2	21	
GPTIMER3	22	
GPTIMER4	23	
UART0	24	
UART1	25	
UART2	26	
UART3	27	
UART4	28	
CAN0	29	
I2C0	30	
I2C1	31	
DMAC0_INTR	32	DMA combined interrupt
DMAC0_INTTC	33	DMA terminal count interrupt
DMAC0_INTERR	34	DMA error interrupt
USB0	35	
MAC0	36	
EXT_INT0	37	
EXT_INT1	38	
EXT_INT2	39	
EXT_INT3	40	
EXT_INT4	41	
EXT_INT5	42	
EXT_INT6	43	

EXT_INT7	44	

6.3 Overall priority

From highest priority to lowest:

- LOCAL_INT3
- LOCAL_INT2
- LOCAL_INT1
- LOCAL_INT0
- External interrupts from PLIC
- Machine software interrupt
- Machine timer interrupt

6.4 Interrupt enable

- The machine interrupt enable (MIE) bit of the RISC-V machine status register (mstatus) must be set as a global enable for all interrupts.
- Corresponding bits in the RISC-V machine interrupt enable register (mie) must be set for each type of interrupt to work:
  - The machine external interrupt enable (MEIE) bit for external interrupts.
  - The machine software interrupt enable (MSIE) bit for machine software interrupt.
  - The machine timer interrupt enable (MTIE) bit for machine timer interrupt.
  - Bits 16-19 for LOCAL_INT0-3, respectively.

6.5 Interrupt registers

- Machine software interrupt pending (MSIP)
  - Address: 0x2000000
  - Bit 0:
    - Write 1 to trigger machine software interrupt
    - Write 0 to clear the pending status
- Machine timer compare low (MTIMECMP_LO)
  - Address: 0x2004000
  - Bit [31:0]: Lower 32 bits of the machine timer compare register
- Machine timer compare high (MTIMECMP_HI)
  - Address: 0x2004004
  - Bit [31:0]: Higher 32 bits of the machine timer compare register

- Machine timer low (MTIME_LO)
  - Address: 0x200bFF8
  - Bit [31:0] Lower 32 bits of the machine timer register
- Machine timer high (MTIME_HI)
  - Address: 0x200bFFC
  - Bit [31:0] Higher 32 bits of the machine timer register
- External interrupt priority (PRIORITY)
  - Address: 0xC000000 + (interrupt number * 4)
  - Each priority registers holds the priority level of the corresponding interrupt
  - The valid range of priority level is from 0 (lowest, interrupt disabled) to 15 (highest).
- External interrupt pending (PENDING)
  - Address: 0xC001000
  - Each interrupt has 1 bit pending status. The bit offset is decided by the interrupt number.
  - The bit is set automatically by hardware when the corresponding interrupt is triggered and is cleared automatically by reading the CLAIM_COMPLETE register when the corresponding interrupt has the highest priority.
- External interrupt enable (ENABLE)
  - Address: 0xC002000
  - Each interrupt has 1 bit enable. The bit offset is decided by the interrupt number.
  - Each bit can be set or cleared by software.
- External interrupt threshold (THRESHOLD)
  - Address: 0xC200000
  - Bit [3:0]: Can be set by software to determine the external interrupt threshold. Only those external interrupts that have higher priority than THRESHOLD will trigger an interrupt to the CPU core.
- External interrupt claim and complete (CLAIM_COMPLETE)
  - Address: 0xC200004
  - Reading this register will return the highest priority pending interrupt number and clear the corresponding pending bit (only for enabled interrupts with above threshold priority). Since Interrupts are numbered starting from 1, a read value of 0 means no active interrupt. A write to this register will complete the interrupt and make the written interrupt number ready to respond again

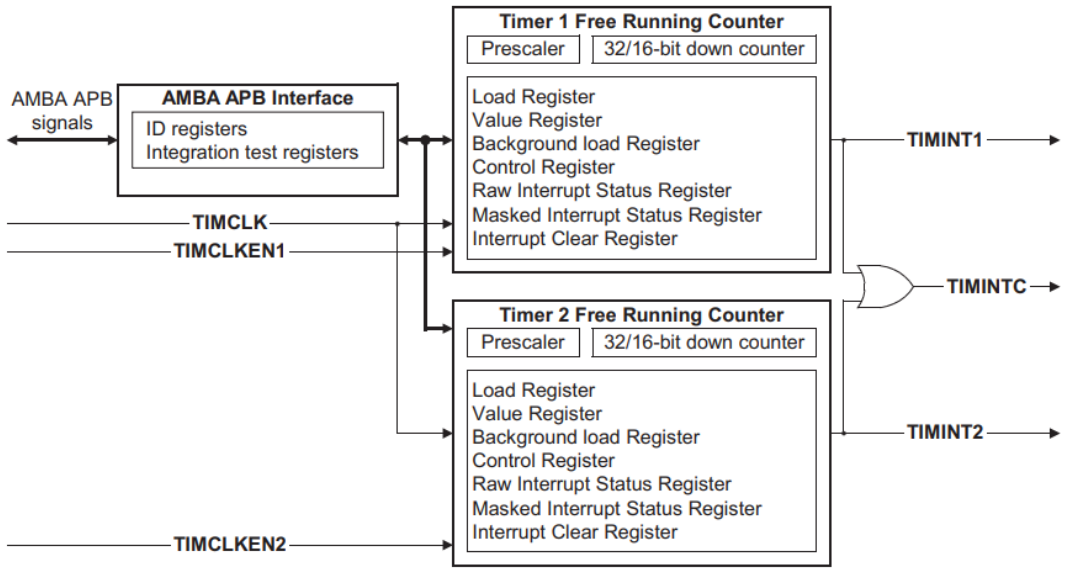
7 Dual Timer(Basic Timers)

7.1 Introduction

The Dual-Timer module consists of two programmable 32/16-bit down counters that can generate interrupts on reaching zero.

- Two 32/16-bit down counters with free-running, periodic and one-shot modes.
- Common clock with separate clock-enables for each timer gives flexible control of the timer intervals.
- Interrupt output generation on timer count reaching zero.
- Identification registers that uniquely identify the Dual-Timer module. These can be used by software to automatically configure itself.

Figure below shows a simplified block diagram of the module.



7.2 Functional Overview

7.2.1 Overview

The Dual-Timer module consists of two identical programmable Free Running Counters (FRCs) that can be configured for 32-bit or 16-bit operation and one of three timer modes;

• free-running

• periodic

• one-shot.

The FRCs operate from a common timer clock, **TIMCLK** with each FRC having its own clock enable input, **TIMCLKEN1** and **TIMCLKEN2**. Each FRC also has a prescaler that can divide down the enabled **TIMCLK** rate by 1, 16, or 256. This enables the count rate for each FRC to be controlled independently using their individual clock enables and prescalers.

TIMCLK can be equal to or be a submultiple of the **PCLK** frequency. However, the positive edges of **TIMCLK** and **PCLK** must be synchronous and balanced.

The operation of each Timer module is identical. A Timer module can be programmed for a 32-bit or 16-bit counter size and one of three timer modes using the Control Register. The three timer modes are:

Free-running The counter operates continuously and wraps around to its maximum value each time that it reaches zero.

Periodic The counter operates continuously by reloading from the Load Register each time that the counter reaches zero.

One-shot The counter is loaded with a new value by writing to the Load Register. The counter decrements to zero and then halts until it is reprogrammed.

The timer count is loaded by writing to the Load Register and, if enabled, the timer count decrements at a rate determined by **TIMCLK**, **TIMCLKENX**, and the prescaler setting. When the Timer counter is already running, writing to the Load Register causes the counter to immediately restart from the new value.

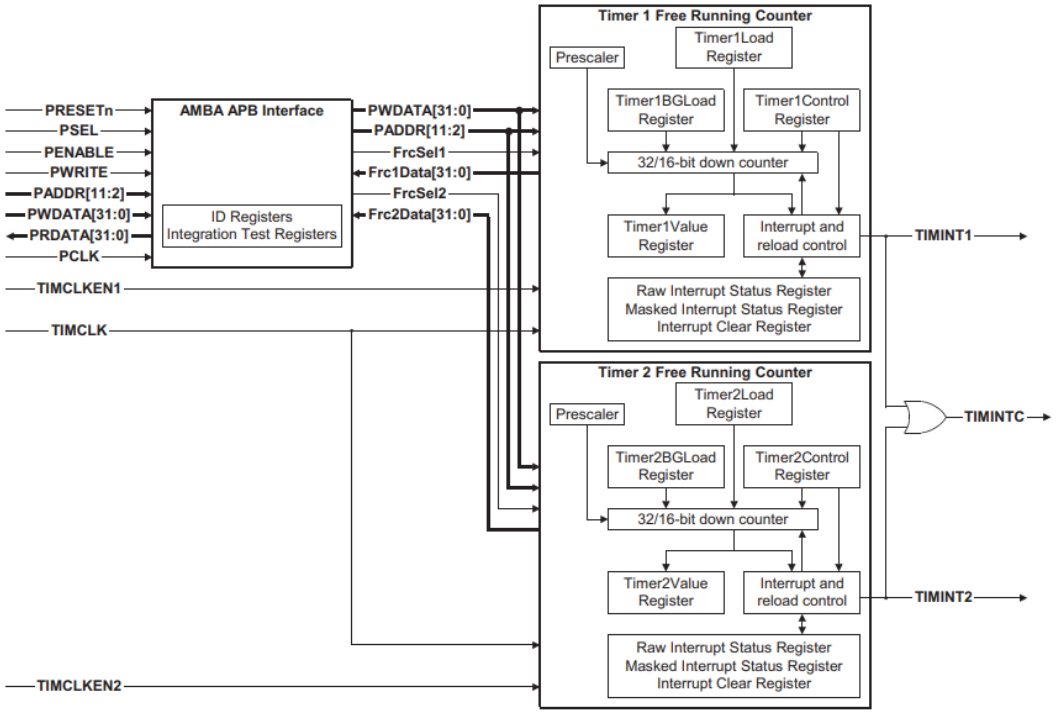
An alternative way of loading the Timer count is by writing to the Background Load Register. This has no immediate effect on the current count but the counter continues to decrement. On reaching zero, the Timer count is reloaded from the new load value if it is in periodic mode.

When the Timer count reaches zero an interrupt is generated. The interrupt is cleared by writing to the Interrupt Clear Register. The external interrupt signals can be masked off by the Interrupt Mask Registers.

The current counter value can be read from the Value Register at any time.

7.2.2 Functional description

The Dual-Timer module block diagram is shown in Figure below.



7.2.2.1 AMBA APB Interface

The AMBA APB slave interface generates read and write decodes for accesses to all registers in the Dual-Timer module.

7.2.2.2 Free-running counter blocks

The two FRCs are identical and contain the 32/16-bit down counter and interrupt functionality. The counter logic is clocked independently of **PCLK** by **TIMCLK** in conjunction with a clock enable **TIMCLKENX** although there are constraints on the relationship between **PCLK** and **TIMCLK**.
Although the two FRCs are driven from a common clock, **TIMCLK**, each timer count rate can be independently controlled by their respective clock enables, **TIMCLKEN1** and **TIMCLKEN2**. The prescaler in each FRC gives a further independent control of the count rate of each FRC.

7.2.2.3 Interface reset

The Dual-Timer module is reset by the global reset signal **PRESETn**.
PRESETn can be asserted asynchronously to PCLK but must be deasserted synchronously to the rising edge of PCLK. **PRESETn** is used to reset the state of the Dual-Timer module registers. The Dual-Timer module requires that **PRESETn** is asserted LOW for at least one period of **PCLK**. In summary, the Timer is initialized to the following state after reset:
• the counter is disabled
• free-running mode is selected
• 16-bit counter mode is selected
• prescalers are set to divide by 1
• interrupts are cleared but enabled
• the Load Register is set to zero
• the counter Value is set to 0xFFFFFFFF.

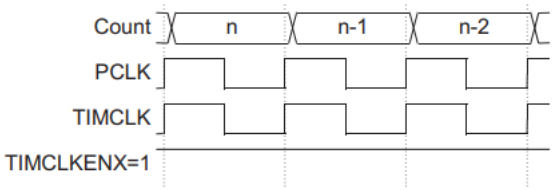
7.2.2.4 Clock signals and clock enables

The Dual-Timer module uses two input clocks:
• **PCLK** is used to time all APB accesses to the Dual-Timer module registers.
• **TIMCLK** is qualified by the clock enables, **TIMCLKEN1** and **TIMCLKEN2**, and used to clock the prescalers, counters and their associated interrupt logic. This qualified **TIMCLK** rate is referred to as the effective timer clock rate. The prescaler counter only decrements on a rising edge of TIMCLK when **TIMCLKENX** is HIGH. The Timer counter only decrements on a rising edge of TIMCLK when **TIMCLKENX** is HIGH and the prescaler counter generates an enable.

The relationship between **TIMCLK** and **PCLK** must observe the following constraints:
• the rising edges of **TIMCLK** must be synchronous and balanced with a rising edge of **PCLK**
• **TIMCLK** frequency cannot be greater than **PCLK** frequency.

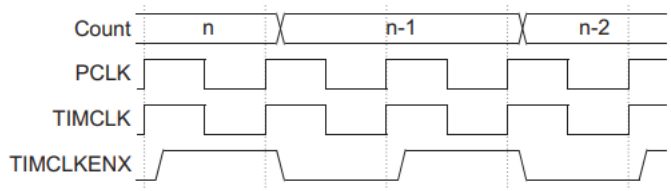
TIMCLK, **TIMCLKEN1**, and **TIMCLKEN2** can be used in the ways described in the following sections:
• TIMCLK equals PCLK and TIMCLKENX equals one
• TIMCLK equals PCLK and TIMCLKENX is pulsed
• TIMCLK is less than PCLK and TIMCLKENX equals
• TIMCLK is less than PCLK and TIMCLKENX is pulsed.
Note:
Unless otherwise stated these examples use a prescale setting of divide by 1. The examples apply to either Timer1 or Timer2 in the module. TIMCLKENX refers to either TIMCLKEN1 or TIMCLKEN2.

TIMCLK equals PCLK and TIMCLKENX equals one
Figure below shows the case where TIMCLK is identical to **PCLK** and **TIMCLKENX** is permanently enabled. In this case, the counter is decremented on every **TIMCLK** edge.



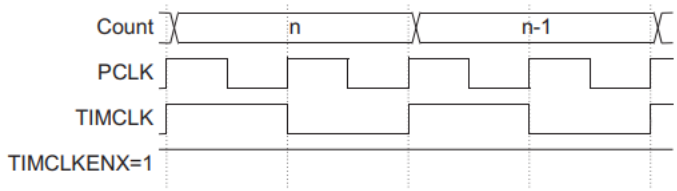
TIMCLK equals PCLK and TIMCLKENX is pulsed

Figure below shows the case where **TIMCLK** is identical to **PCLK** but **TIMCLKENX** only enables every second **TIMCLK** edge. In this case, the counter is decremented on every second **TIMCLK** rising edge.



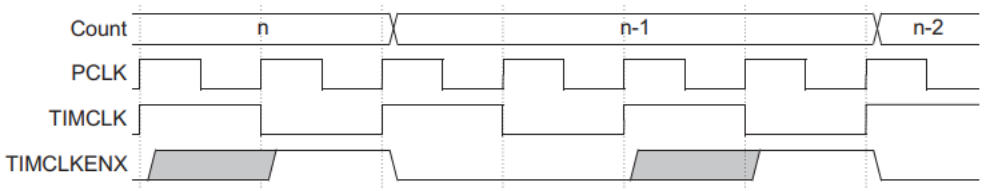
TIMCLK is less than PCLK and TIMCLKENX equals one

Figure below shows the case where **TIMCLK** frequency is a submultiple of the **PCLK** frequency but the rising edges of **TIMCLK** are synchronous and balanced with **PCLK** edges. **TIMCLKENX** is permanently enabled. In this case, the counter is decremented on every **TIMCLK** rising edge.



TIMCLK is less than PCLK and TIMCLKENX is pulsed

Figure below shows the case where **TIMCLK** frequency is a submultiple of the **PCLK** frequency but the rising edges of **TIMCLK** are synchronous and balanced with **PCLK** edges. **TIMCLKENX** only enables every second **TIMCLK** edge. In this case, the counter is decremented on every second **TIMCLK** rising edge.



7.2.2.5 Prescaler operation

The prescaler generates a timer clock enable that is used to enable the decrementing of the timer counter at one of the following rates:

• the effective timer clock rate where **TIMCLK** is qualified by **TIMCLKENX**

• the effective timer clock rate divided by 16

• the effective timer clock rate divided by 256.

Figure below shows how the timer clock enable is generated by the prescaler

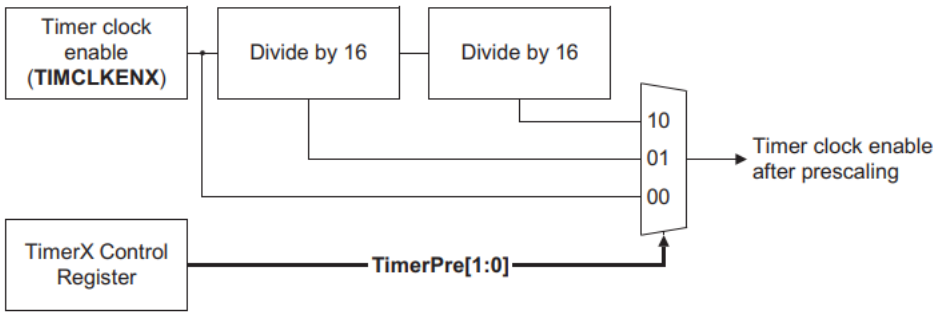
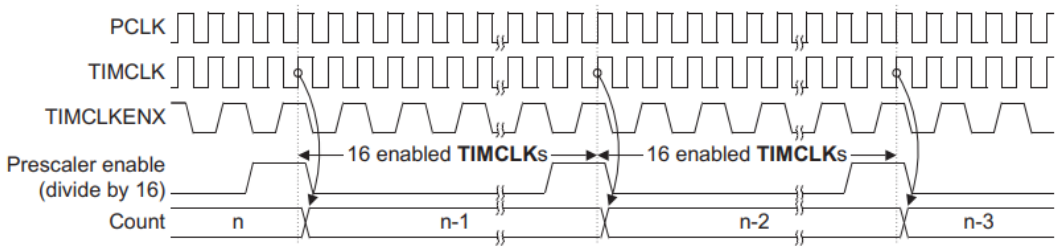


Figure below shows an example of how the prescaler generates the timer clock enable for a prescaler setting of divide by 16.



7.2.2.6 Timer operation

After the initial application and release of PRESETn, the Timer state is initialized as follows:

• the counter is disabled, TimerEn=0

• free-running mode is selected, TimerMode=0 and OneShot=0

• 16-bit counter mode is selected, TimerSize=0

• prescalers are set to divide by 1, TimerPre=0x0

• interrupts are cleared but enabled, IntEnable=1

• the Load Register is set to zero

• the counter Value is set to 0xFFFFFFFF.

The operation in each of the three Timer modes is described in:

• Free-running mode
• Periodic mode
• One-shot mode

Free-running mode

Free-running mode is selected by setting the following bits in the TimerControl Register:
• set TimerMode bit to 1
• set OneShot bit to 0.
The 32-bit or 16-bit counter operation is selected by setting the TimerSize bit appropriately in the TimerControl Register.

On reset the timer value is initialized to 0xFFFFFFFF and if the counter is enabled then the count decrements by one for each **TIMCLK** positive edge when **TIMCLKENX** is HIGH and the prescaler generates an enable pulse. Alternatively, a new initial counter value can be loaded by writing to the TimerXLoad Register and the counter starts decrementing from this value if the counter is enabled.
In 32-bit mode, when the count reaches zero, 0x00000000, an interrupt is generated and the counter wraps around to 0xFFFFFFFF irrespective of the value in the TimerXLoad Register. The counter starts to decrement again and this whole cycle repeats for as long as the counter is enabled.
In 16-bit mode, only the least significant 16-bits of the counter are decremented and when the count reaches 0x0000, an interrupt is generated and the counter wraps round to 0xFFFF irrespective of the value in the TimerXLoad Register.
If the counter is disabled by clearing the TimerEn bit in the TimerControl Register, the counter halts and holds its current value. If the counter is re-enabled again then the counter continues decrementing from the current value.
The counter value can be read at any time by reading the TimerXValue Register.

Periodic mode

Periodic mode is selected by setting the following bits in the TimerControl Register:
• set TimerMode bit to 0
• set OneShot bit to 0.
The 32-bit or 16-bit counter operation is selected by setting the TimerSize bit appropriately in the TimerControl Register.
An initial counter value can be loaded by writing to the TimerXLoad Register and the counter starts decrementing from this value if the counter is enabled.
In 32-bit mode, the full 32 bits of the counter are decremented and when the count reaches zero, 0x00000000, an interrupt is generated and the counter reloads with the value in the TimerXLoad Register. The counter starts to decrement again and this whole cycle repeats for as long as the counter is enabled.
In 16-bit mode, only the least significant 16-bits of the counter are decremented and when the count reaches 0x0000, an interrupt is generated and the counter reloads with the value in the TimerXLoad Register. The counter starts to decrement again and this whole cycle repeats for as long as the counter is enabled.
If a new value is loaded into the counter by writing to the TimerXLoad Register while the counter is running then the counter values changes to the new load value on the next **TIMCLK** when **TIMCLKENX**

is HIGH.

If a new value is written to the Background Load Register, TimerXBGLoad, while the counter is running then the TimerXLoad Register is also updated with the same load value but the counter continues to decrement to zero. When it reaches zero, the counter reloads with the new load value and uses this new load value for each subsequent reload for as long as the timer is enabled in periodic mode.

If the counter is disabled by clearing the TimerEn bit in the TimerControl Register, the counter halts and holds its current value. If the counter is re-enabled again then the counter continues decrementing from the current value.

One-shot mode

One-shot timer mode is selected by setting the OneShot bit in the TimerControl Register to 1. The TimerMode bit has no effect in one-shot mode.

The 32-bit or 16-bit counter operation is selected by setting the TimerSize bit appropriately in the TimerControl Register.

To initiate a count down sequence in one-shot mode, write a new load value to the TimerXLoad Register and the counter starts decrementing from this value if enabled.

In 32-bit mode, the full 32-bits of the counter are decremented and when the count reaches zero, 0x00000000, an interrupt is generated and the counter halts.

In 16-bit mode, only the least significant 16-bits of the counter are decremented and when the count reaches 0x0000, an interrupt is generated and the counter halts.

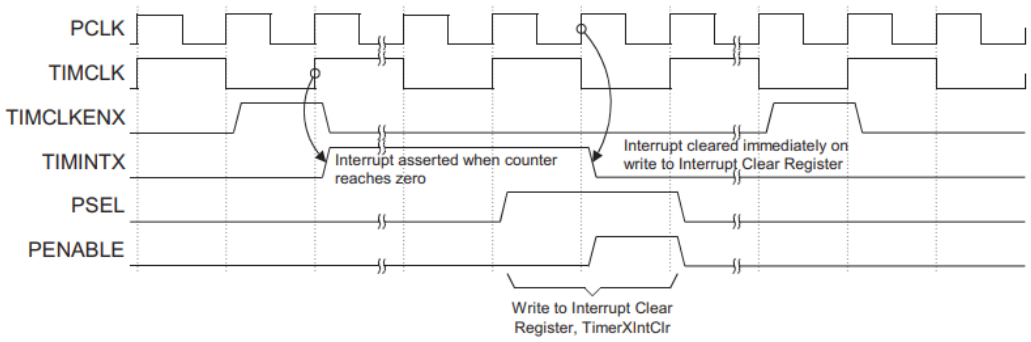
One-shot mode can be retriggered by writing a new value to the TimerXLoad Register. The counter values changes to the new load value on the next **TIMCLK** when **TIMCLKENX** is HIGH.

7.2.2.7 Interrupt behavior

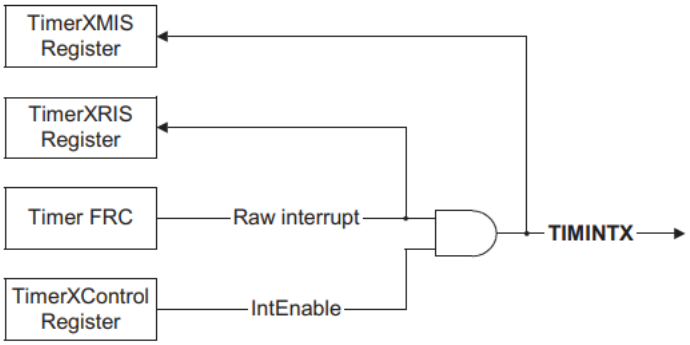
An interrupt is generated if IntEnable=1 and the counter reaches 0x00000000 in 32-bit mode or 0xXXXX0000 in 16-bit mode. The most significant 16 bits of the counter are ignored in 16-bit mode.

When the Timer module raises an interrupt by asserting **TIMINTX**, the timing of this signal is generated from a rising clock edge of **TIMCLK** enabled by **TIMCLKENX**. When the interrupt is cleared by a write to the Interrupt Clear Register, TimerXIntClr, the **TIMINTX** signal is deasserted immediately in the **PCLK** domain rather than waiting for the next enabled **TIMCLK** rising edge.

Figure below illustrates an example of the timing for an interrupt being raised and cleared.



The interrupt signals generated by the Timer module, **TIMINT1** and **TIMINT2**, can be masked by setting the IntEnable bit to 0 in the TimerXControl Register. The raw interrupt status prior to masking can be read from the TimerXRIS Register and the masked interrupt status can be read from the TimerXMIS Register. Figure below shows how the raw and masked interrupt status is accessed.



7.2.2.8 Programming the timer interval

Table below shows the equations that are used to calculate the timer interval generated for each timer mode in terms of:

• **TIMCLK$_{FREQ}$** is the frequency of **TIMCLK**.

• **TIMCLKENX$_{DIV}$** is the effective division of the **TIMCLK** rate by the clock enable, **TIMCLKENX**. For example, if **TIMCLKENX** enables every fourth TIMCLK edge then **TIMCLKENX$_{DIV}$**=4.

• PRESCALEDIV is the prescaler division factor of 1, 16, or 256. Derived from Control Register bits[3:2].

• TimerXLoad is the value in the Load Register.

Mode	Interval
Free-running 32-bit	$\left[\frac{\text{TIMCLKENX}_{DIV} \times \text{PRESCALE}_{DIV}}{\text{TIMCLK}_{FREQ}}\right] \times 2^{32}$
Free-running 16-bit	$\left[\frac{\text{TIMCLKENX}_{DIV} \times \text{PRESCALE}_{DIV}}{\text{TIMCLK}_{FREQ}}\right] \times 2^{16}$
Periodic and One-shot	$\left[\frac{\text{TIMCLKENX}_{DIV} \times \text{PRESCALE}_{DIV}}{\text{TIMCLK}_{FREQ}}\right] \times \text{TimerXLoad}$

For example, the TimerXLoad value required for a 1ms periodic interval with **TIMCLK**=100MHz, **TIMCLKENX**$_{DIV}$=1, and PRESCALE$_{DIV}$=1 is calculated as shown in Example below.

$$\text{TimerXLoad} = \left[\frac{\text{Interval} \times \textbf{TIMCLK}_{FREQ}}{\textbf{TIMCLKENX}_{DIV} \times \text{PRESCALE}_{DIV}} \right]$$

$$\text{TimerXLoad} = \left[\frac{1\text{ms} \times 100\text{MHz}}{1 \times 1} \right] = 10^5 = \texttt{0x000186A0}$$

Note:
The minimum valid value for TimerXLoad is 1. If TimerXload is set to 0 then an interrupt is generated immediately.

7.3 Programmer's Model

7.3.1 ummary of registers

Address	Type	Width	Reset value	Name	Description
Base+0x00	Read/write	32	0x00000000	Timer1Load	See *Load Register, TimerXLoad* on Chapter 3.2.1
Base+0x04	Read	32	0xFFFFFFFF	Timer1Value	See Current Value Register, TimerXValue on Chapter 3.2.2
Base+0x08	Read/write	8	0x20	Timer1Control	See Control Register, TimerXControl on Chapter 3.2.3
Base+0x0C	Write	-	-	Timer1IntClr	See Interrupt Clear Register. TimerXIntClr on Chapter 3.2.4
Base+0x10	Read	1	0x0	Timer1RIS	See Raw Interrupt Status Register, TimerXRIS on Chapter 3.2.5
Base+0x14	Read	1	0x0	Timer1MIS	See Masked Interrupt Status Register, TimerXMIS on Chapter 3.2.6
Base+0x18	Read/write	32	0x00000000	Timer1BGLoad	See Background Load Register, TimerXBGLoad on Chapter 3.2.7
Base+0x20	Read/write	32	0x00000000	Timer2Load	See Load Register, TimerXLoad on Chapter 3.2.1
Base+0x24	Read	32	0xFFFFFFFF	Timer2Value	See Current Value Register, TimerXValue on Chapter 3.2.2
Base+0x28	Read/write	8	0x20	Timer2Control	See Control Register, TimerXControl on Chapter 3.2.3

Base+0x2C	Write	-	-	Timer2IntClr	See Interrupt Clear Register. TimerXIntClr on Chapter 3.2.4
Base+0x30	Read	1	0x0	Timer2RIS	See Raw Interrupt Status Register, TimerXRIS on Chapter 3.2.5
Base+0x34	Read	1	0x0	Timer2MIS	See Masked Interrupt Status Register, TimerXMIS on Chapter 3.2.6
Base+0x38	Read/write	32	0x00000000	Timer2BGLoad	See Background Load Register, TimerXBGLoad on Chapter 3.2.7

7.4 Register descriptions

7.4.1.1 Load Register, TimerXLoad

The TimerXLoad Register is a 32-bit register that contains the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.
When this register is written to directly, the current count immediately resets to the new value at the next rising edge of **TIMCLK** which is enabled by **TIMCLKENX**.

Note:
The minimum valid value for **TimerXLoad** is 1. If **TimerXload** is set to 0 then an interrupt is generated immediately

The value in this register is also over-written if the TimerXBGLoad Register is written to, but the current count is not immediately affected.
If values are written to both the TimerXLoad and TimerXBGLoad Registers before an enabled rising edge on **TIMCLK**, then on the next enabled **TIMCLK** edge the value written to the TimerXLoad value replaces the current count value. After that, each time the counter reaches zero the current count value resets to the value written to TimerXBGLoad.

Reading from the TimerXLoad Register at any time after the two writes have occurred retrieves the value written to TimerXBGLoad. That is, the value read from TimerXLoad is always the value that takes effect for Periodic mode after the next time the counter reaches zero.

7.4.1.2 Current Value Register, TimerXValue

The TimerXValue Register is a 32-bit read-only register that gives the current value of the decrementing counter.
After a load operation has taken place by writing a new load value to TimerXLoad, the TimerXValue

register reflects the new load value immediately in the **PCLK** clock domain without waiting for the next **TIMCLK** edge qualified by **TIMCLKENX**.

Note:

The most significant 16 bits of the 32-bit TimerXValue Register are not automatically set to 0 when in 16-bit timer mode. If the timer is in 16-bit mode then the most significant 16 bits of the TimerXValue Register might have a non-zero value if the timer was previously in 32-bit mode and a write to the TimerXLoad Register has not occurred since the change to 16-bit mode.

7.4.1.3 Control Register, TimerXControl

The bit assignments of the Control Register are listed in Table below.

Table 3-2 Control Register bit assignments

Bits	Name	Type	Function
[31:8]	-	-	Reserved bits, do not modify, and ignore on read
[7]	TimerEn	Read/write	Enable bit: 0 = Timer module disabled (default) 1 = Timer module enabled.
[6]	TimerMode	Read/write	Mode bit: 0 = Timer module is in free-running mode (default) 1 = Timer module is in periodic mode.
[5]	IntEnable	Read/write	Interrupt Enable bit: 0 = Timer module Interrupt disabled 1 = Timer module Interrupt enabled (default).
[4]	-	-	Reserved bit, do not modify, and ignore on read
[3:2]	TimerPre	Read/write	Prescale bits: 00 = 0 stages of prescale, clock is divided by 1 (default) 01 = 4 stages of prescale, clock is divided by 16 10 = 8 stages of prescale, clock is divided by 256 11 = Undefined, do not use.
[1]	TimerSize	Read/write	Selects 16/32 bit counter operation: 0 = 16-bit counter (default) 1 = 32-bit counter.
[0]	OneShot	Read/write	Selects one-shot or wrapping counter mode: 0 = wrapping mode (default) 1 = one-shot mode.

Caution:

The counter mode, size or prescale settings must not be changed while the Timer module is running. If a new configuration is required then the Timer module must be disabled and then the new configuration

values written to the appropriate registers. The Timer module must then be re-enabled after the configuration changes are complete. Failure to follow this procedure can result in unpredictable behavior of the device.

7.4.1.4 Interrupt Clear Register. TimerXIntClr

Any write to this register, clears the interrupt output from the counter.

7.4.1.5 Raw Interrupt Status Register, TimerXRIS

The TimerXRIS Register indicates the raw interrupt status from the counter. The bit assignment is listed in Table below.

Bits	Name	Type	Function
[31:1]	-	-	Reserved bits, do not modify, and ignore on read
[0]	TimerXRIS	Read	Raw interrupt status from the counter

7.4.1.6 Masked Interrupt Status Register, TimerXMIS

The TimerXMIS Register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the Timer Interrupt Enable bit from the control register, and is the same value which is passed to the interrupt output pin, **TIMINTX**. The bit assignment is listed in Table below.

Bits	Name	Type	Function
[31:1]	-	-	Reserved bits, do not modify, and ignore on read
[0]	TimerXMIS	Read	Enabled interrupt status from the counter

7.4.1.7 Background Load Register, TimerXBGLoad

The TimerXBGLoad Register is a 32-bit register that contains the value from which the counter is to decrement. This is the value used to reload the counter when Periodic mode is enabled, and the current count reaches zero.

This provides an alternative method of accessing the TimerXLoad Register. The difference is that writes to TimerXBGLoad do not cause the counter to restart from the new value immediately.

Reading from this register returns the same value returned from TimerXLoad. See Load Register, TimerXLoad on page before for more information.

8 Advanced-control timers

8.1 Introduction

The advanced-control timers consist of a 32-bit auto-reload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control and general-purpose timers are completely independent, and do not share any resources.

8.2 Main features

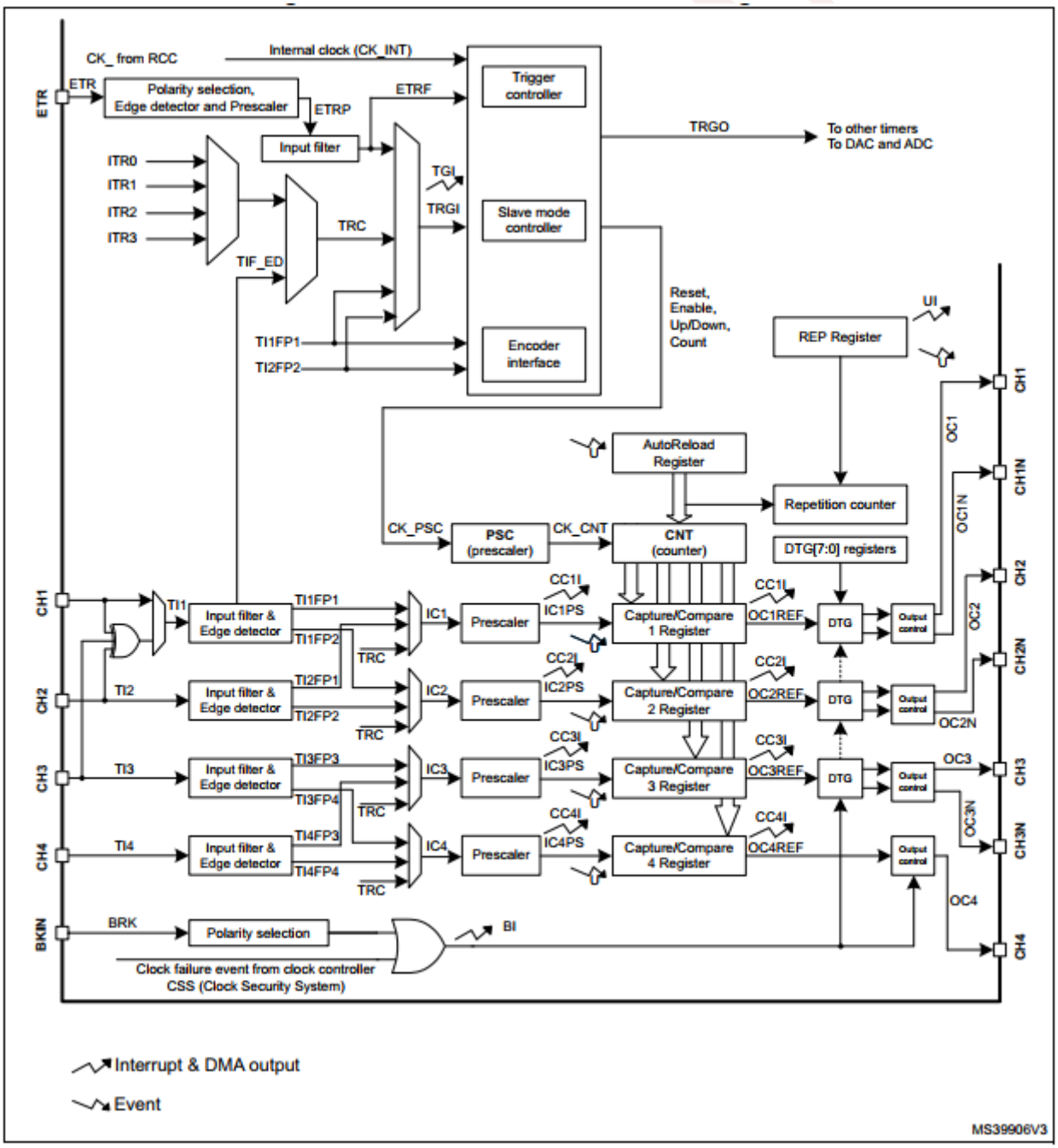
Timer features include:

- 32-bit up, down, up/down auto-reload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency either by any factor between 1 and 65536.
- Up to 4 independent channels for:
  - Input capture
  - Output compare
  - PWM generation (Edge and Center-aligned Mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- Break input to put the timer’s output signals in reset state or in a known state.
  Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)

– Trigger event (counter start, stop, initialization or count by internal/external trigger)
– Input capture
– Output compare
– Break input

- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 1. Advanced-control timer block diagram



8.3 Functional description

8.3.1 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related auto-reload register. The counter can count up, down or both 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 or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TCNT)
- Prescaler register (PSC)
- Auto-reload register (ARR)
- Repetition counter register (RCR)

The auto-reload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the auto-reload preload enable bit (ARPE) in CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output CK_CNT, which is enabled only when the counter enable bit (CEN) in CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting 1 clock cycle after setting the CEN bit in the CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 2 and *Figure 3* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 2. Counter timing diagram with prescaler division change from 1 to 2

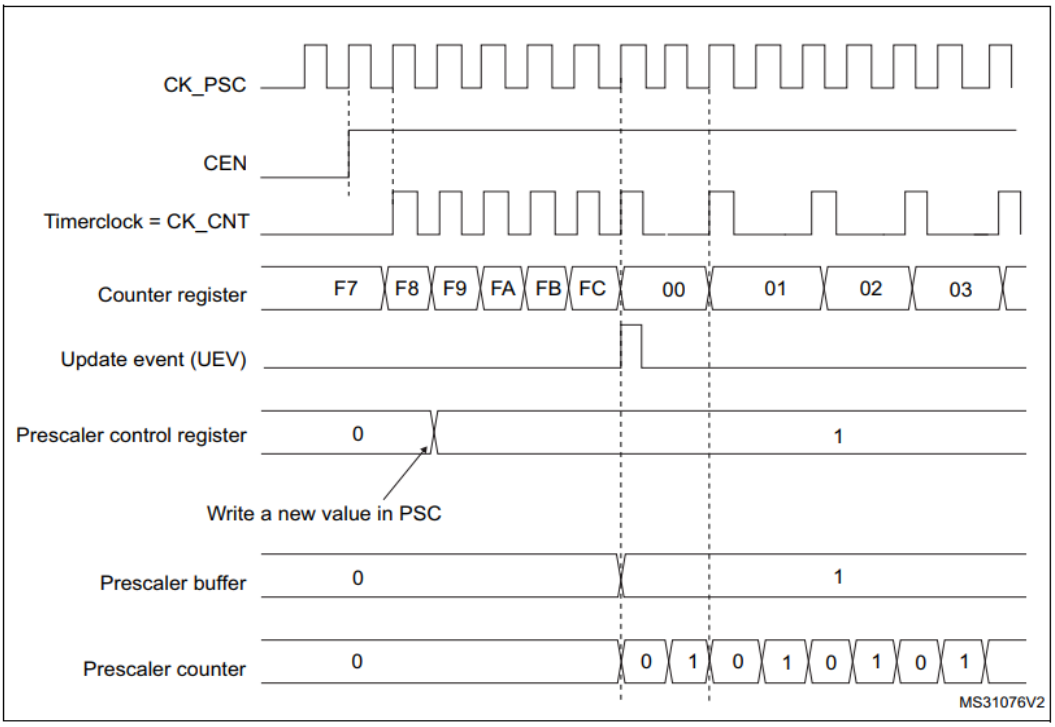
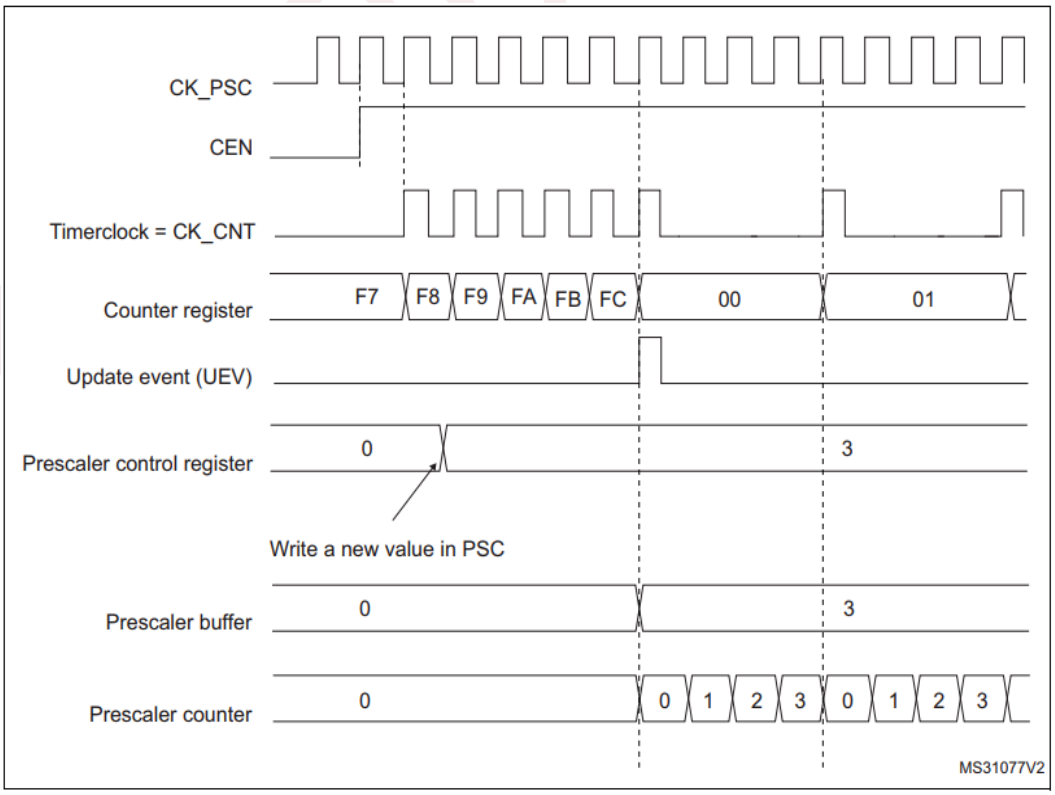


Figure 3. Counter timing diagram with prescaler division change from 1 to 4



8.3.2 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the auto-reload value (content of the ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register plus one (RCR+1). Else the update event is generated at each counter overflow.

Setting the UG bit in the EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of RCR register,
- The auto-reload shadow register is updated with the preload value (ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when ARR=0x36.

Figure 4. Counter timing diagram, internal clock divided by 1

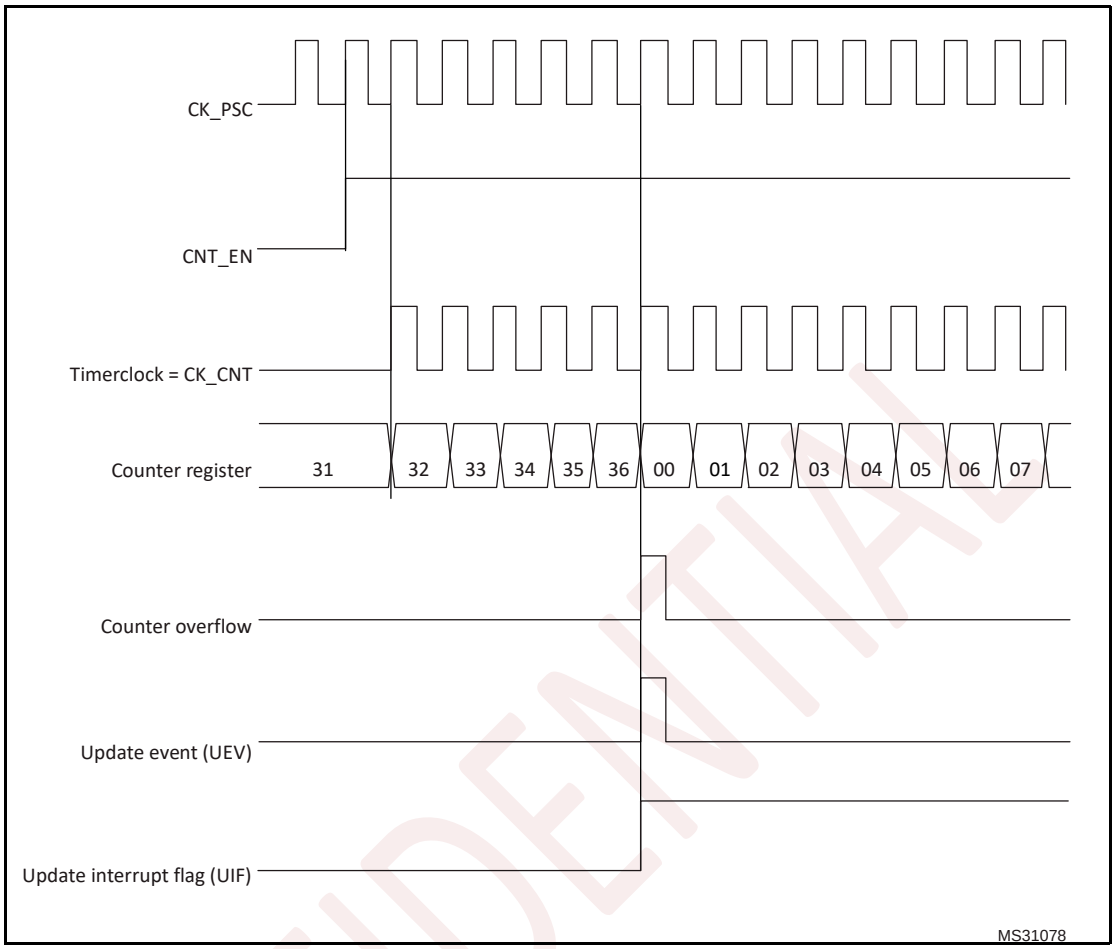


Figure 5. Counter timing diagram, internal clock divided by 2

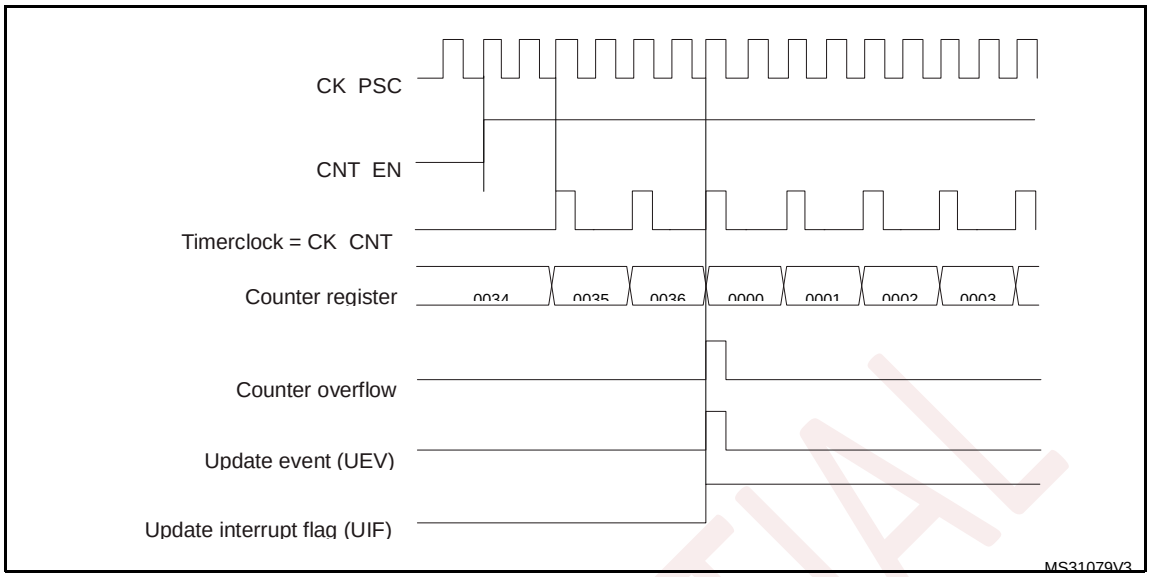


Figure 6. Counter timing diagram, internal clock divided by 4

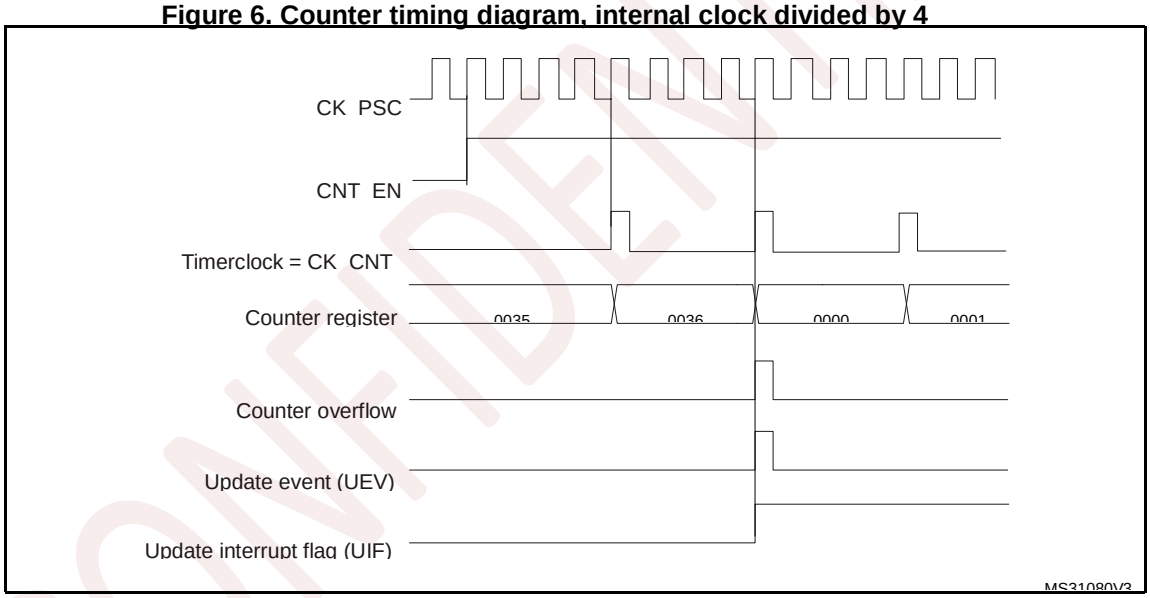


Figure 7. Counter timing diagram, internal clock divided by N

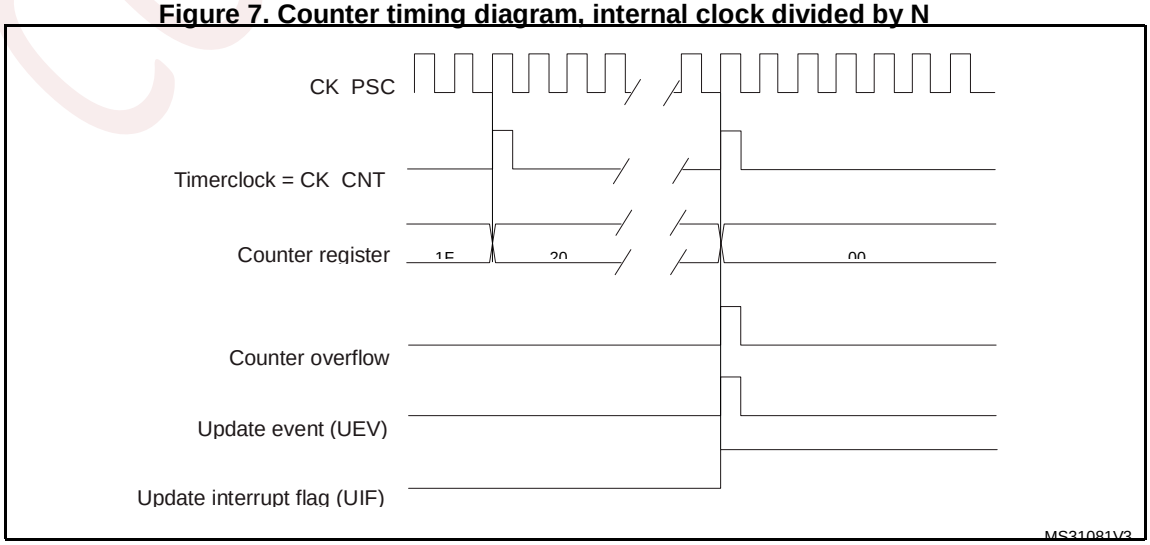


Figure 8. Counter timing diagram, update event when ARPE=0 (ARR not preloaded)

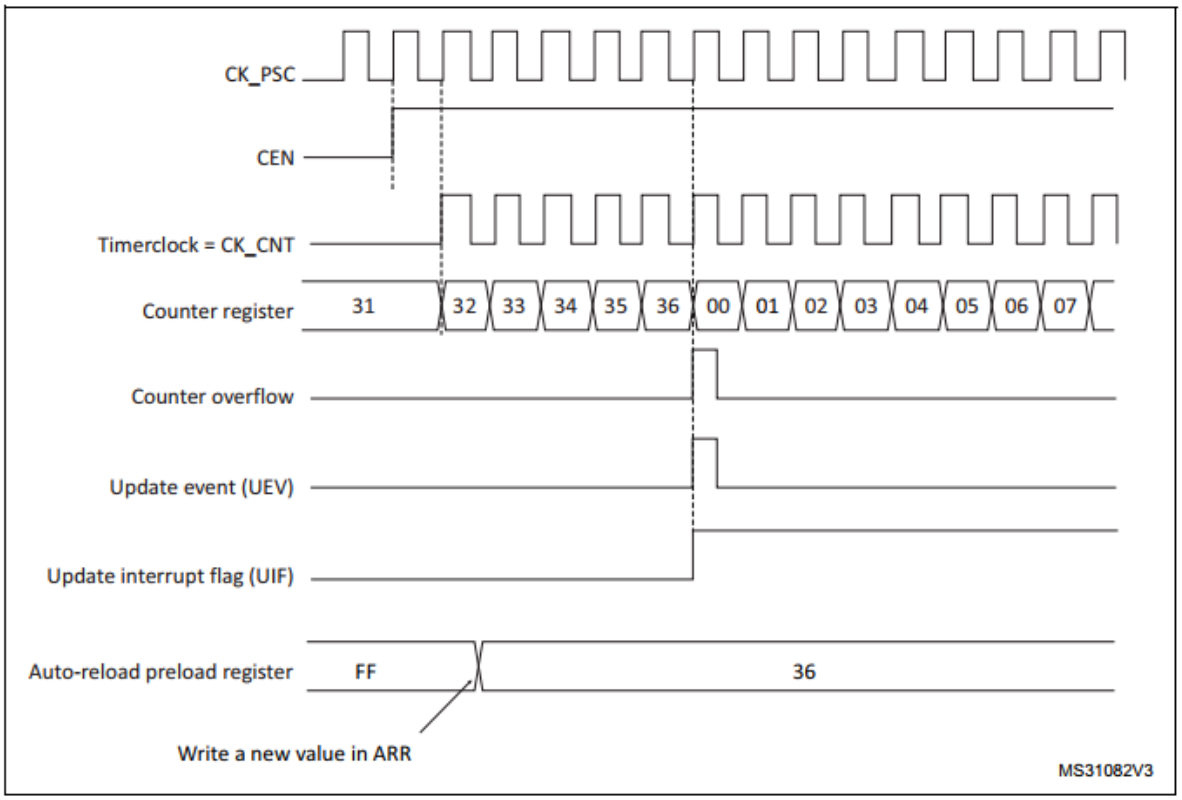
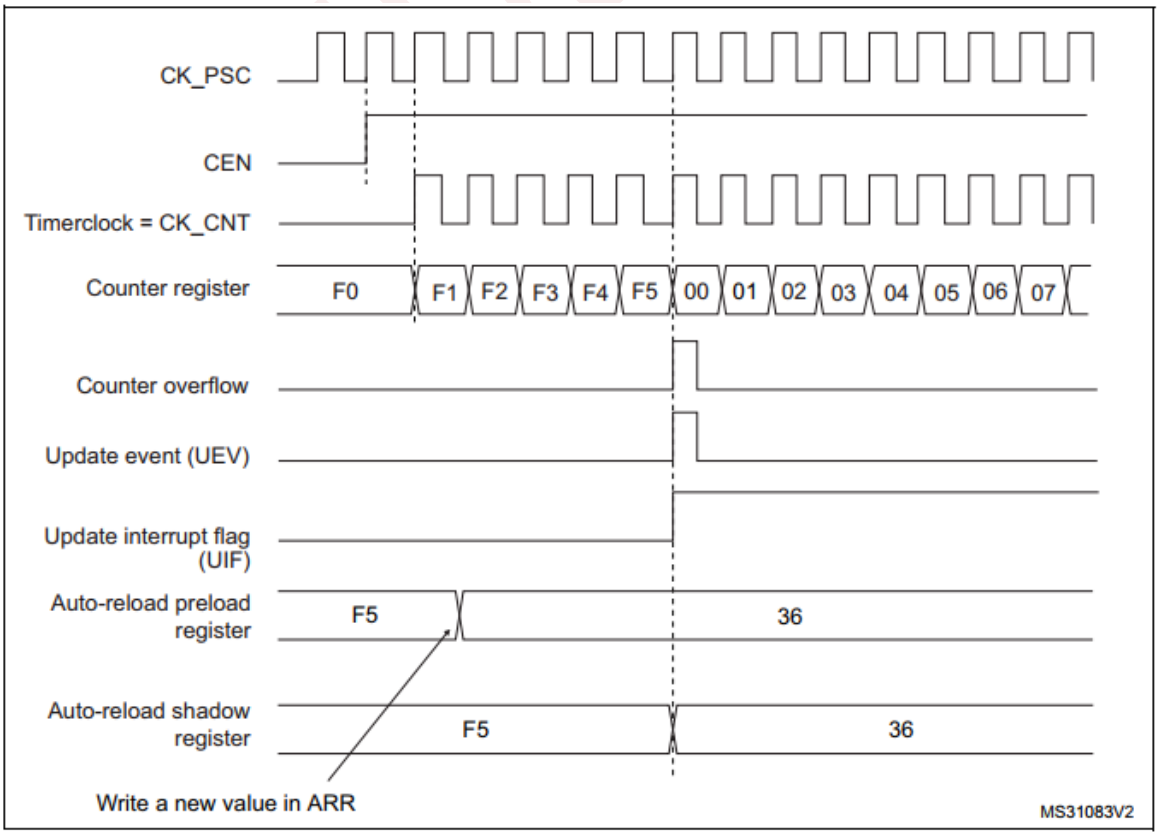


Figure 9. Counter timing diagram, update event when ARPE=1 (ARR preloaded)



Downcounting mode

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

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register plus one (RCR+1). Else the update event is generated at each counter underflow.

Setting the UG bit in the EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current auto-reload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the PSC register)
- The auto-reload active register is updated with the preload value (content of the ARR register). Note that the auto-reload is updated before the counter is reloaded, so that the next period is the expected one

The following figures show some examples of the counter behavior for different clock frequencies when ARR=0x36.

Figure 10. Counter timing diagram, internal clock divided by 1

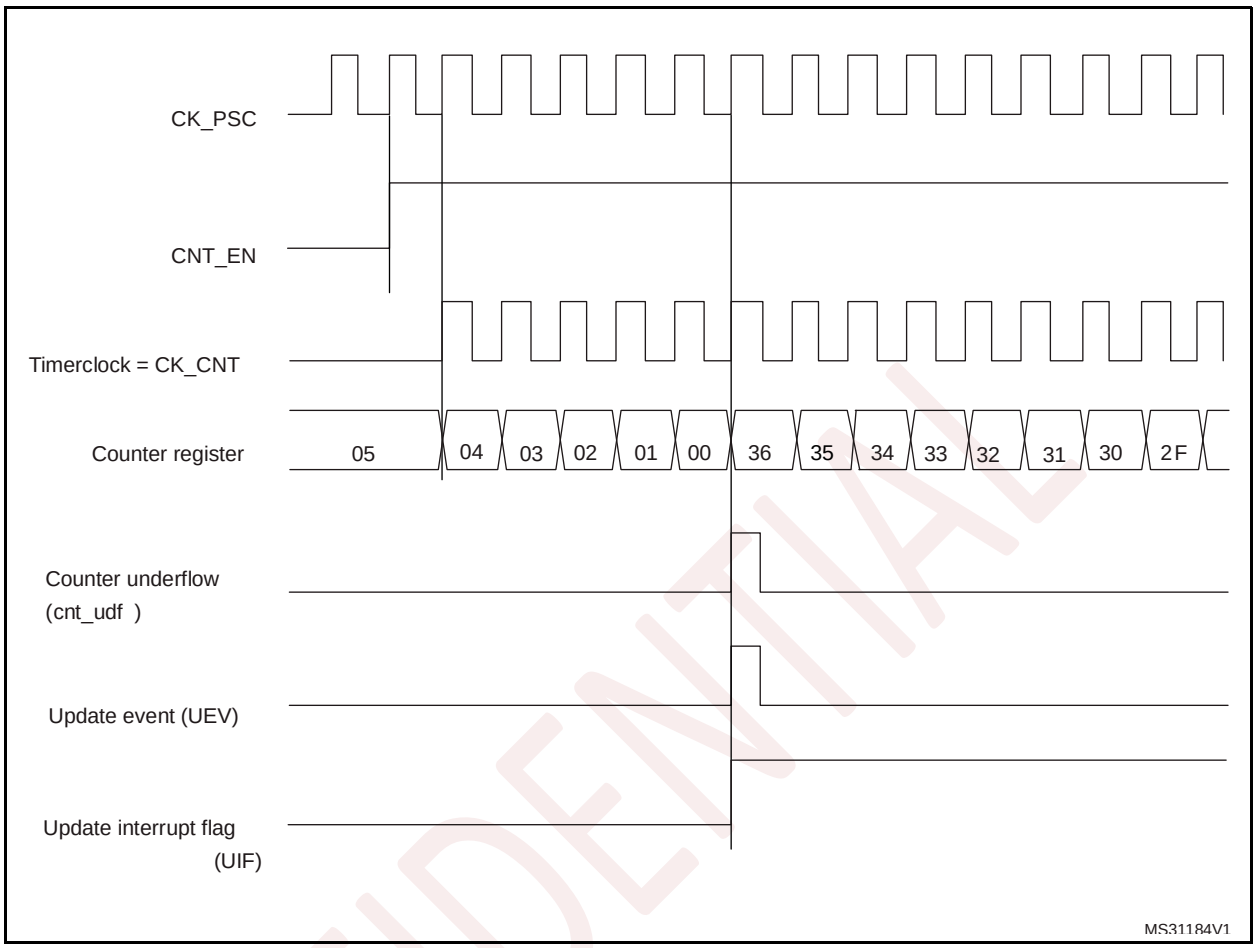


Figure 11. Counter timing diagram, internal clock divided by 2

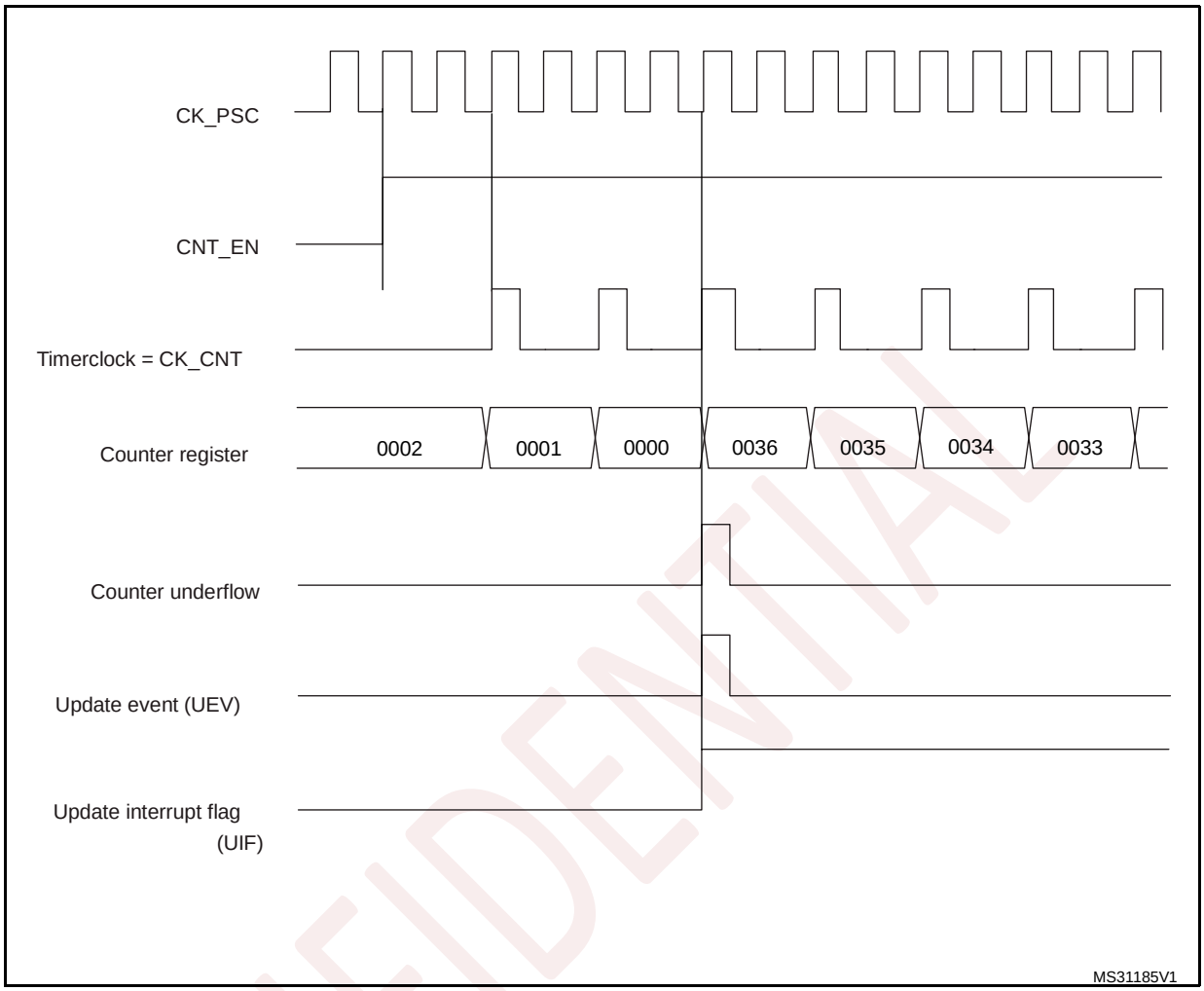


Figure 12. Counter timing diagram, internal clock divided by 4

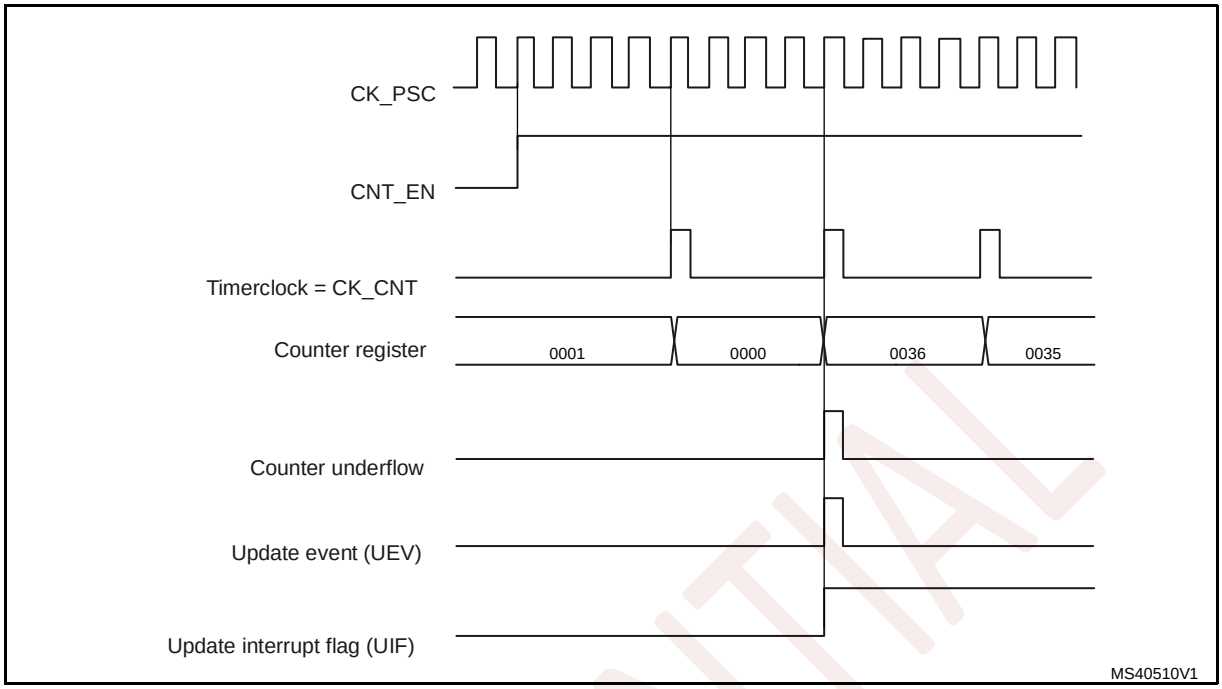


Figure 13. Counter timing diagram, internal clock divided by N

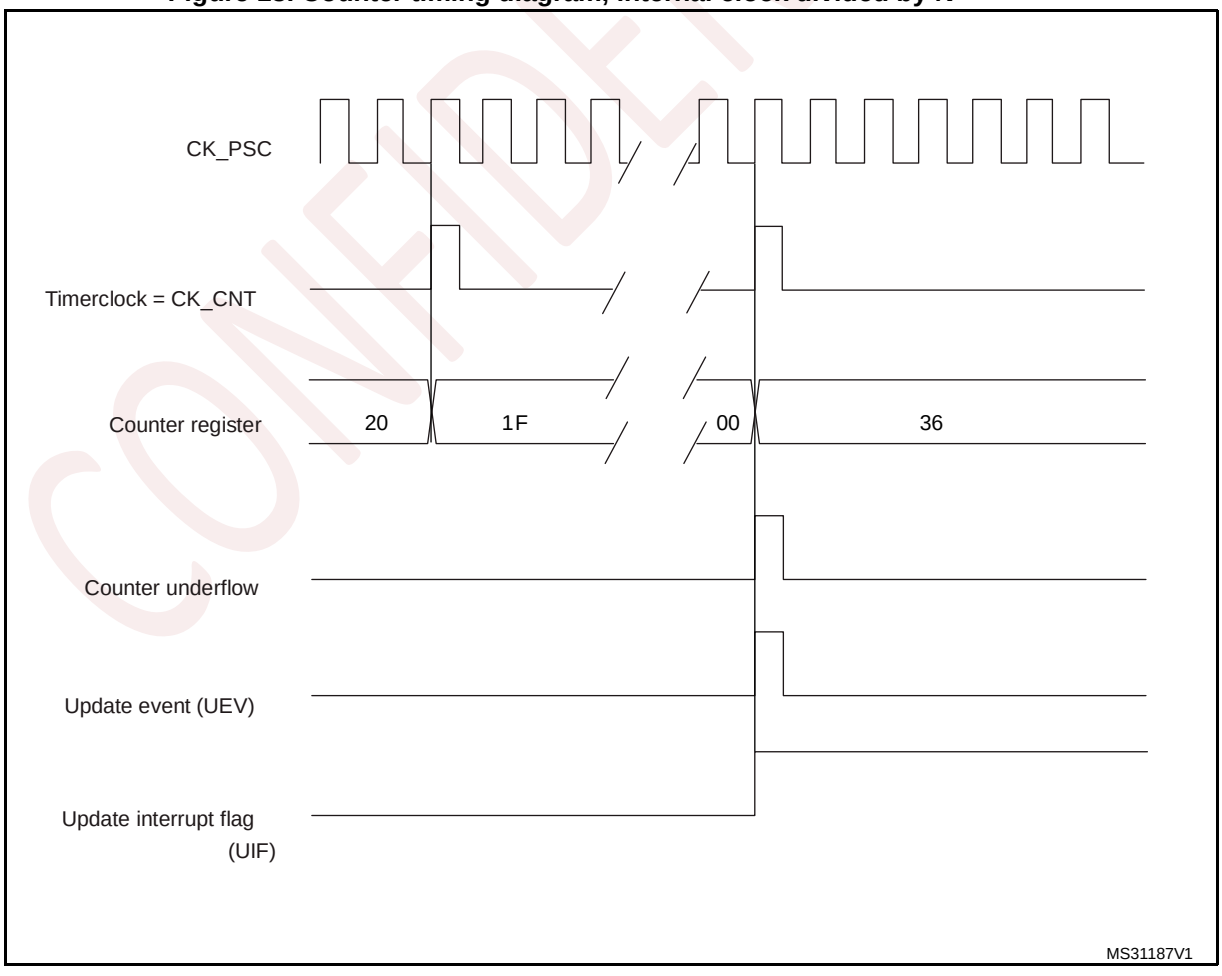
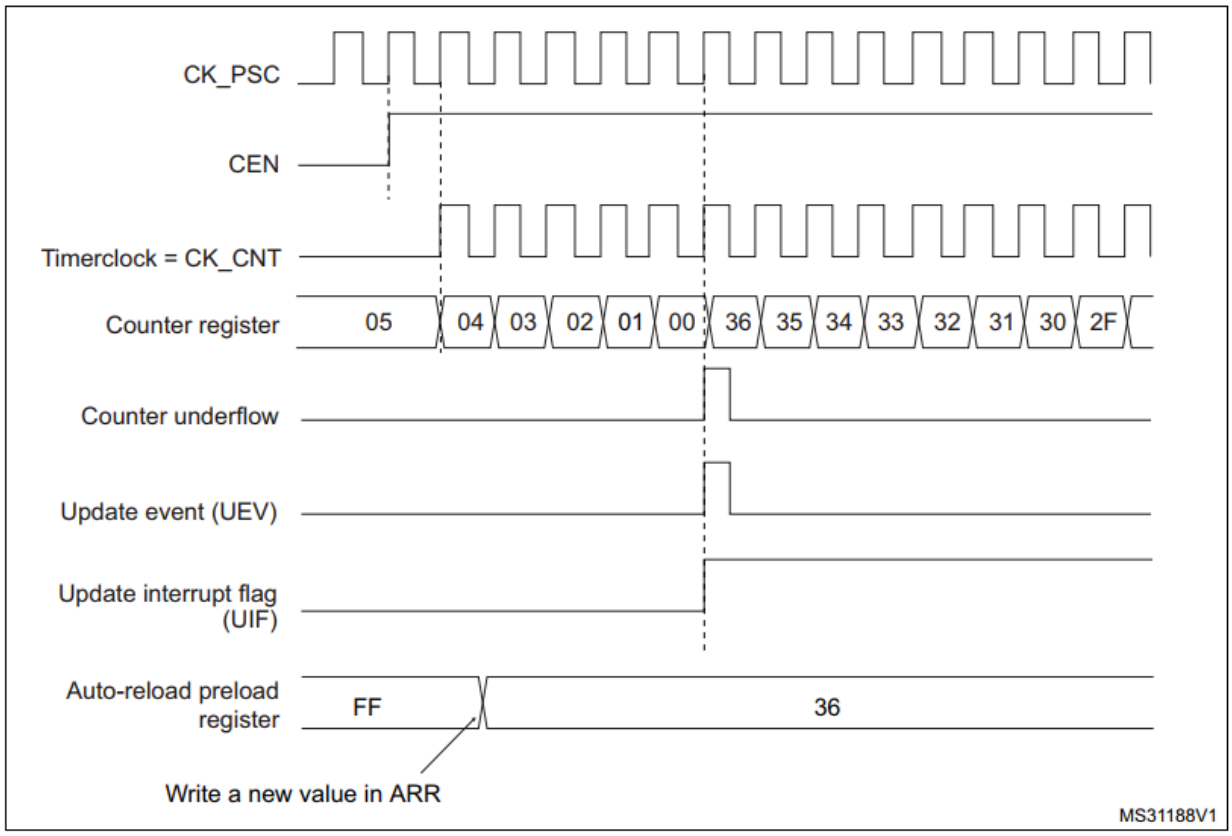


Figure 14. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the ARR register) – 1, 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.

Center-aligned mode is active when the CMS bits in CR1 register are not equal to '00'. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = "01"), the counter counts up (Center aligned mode 2, CMS = "10") the counter counts up and down (Center aligned mode 3, CMS = "11").

In this mode, the DIR direction bit in the 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 or by setting the UG bit in the EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until 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 URS bit (update request selection) in CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the PSC register)
- The auto-reload active register is updated with the preload value (content of the ARR register). Note that if the update source is a counter overflow, the autoreload 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.

Figure15. Counter timing diagram, internal clock divided by 1, ARR = 0x6

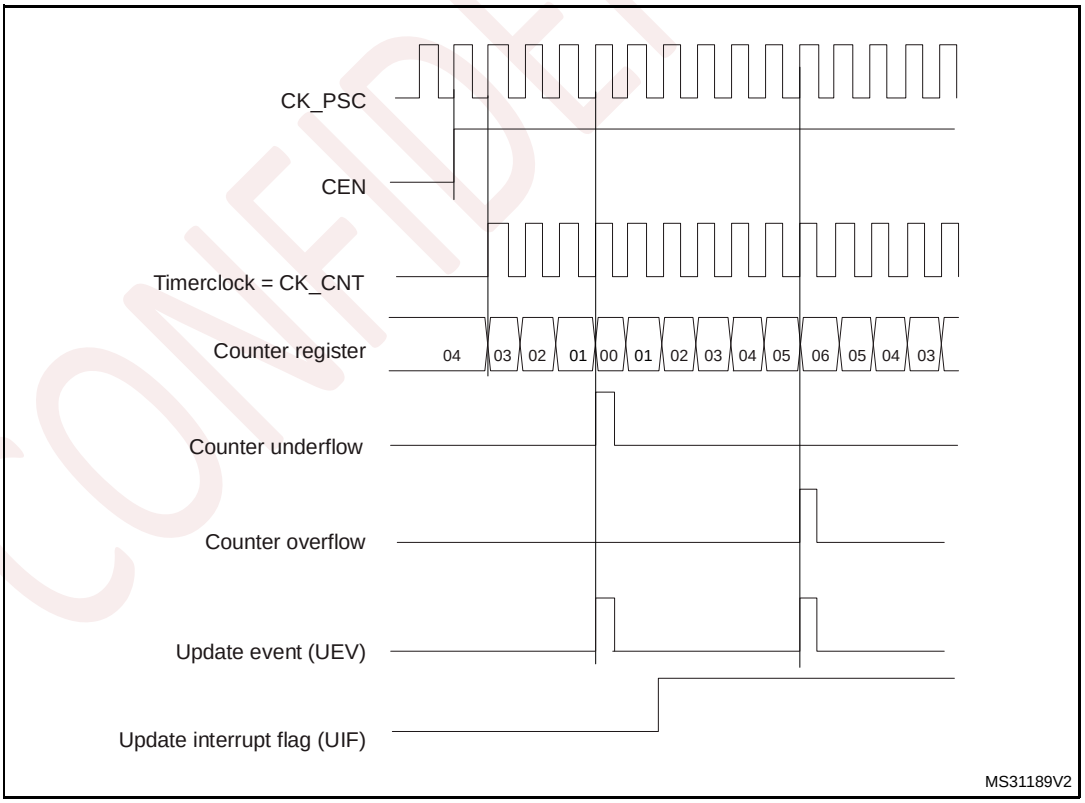


Figure16. Counter timing diagram, internal clock divided by 2

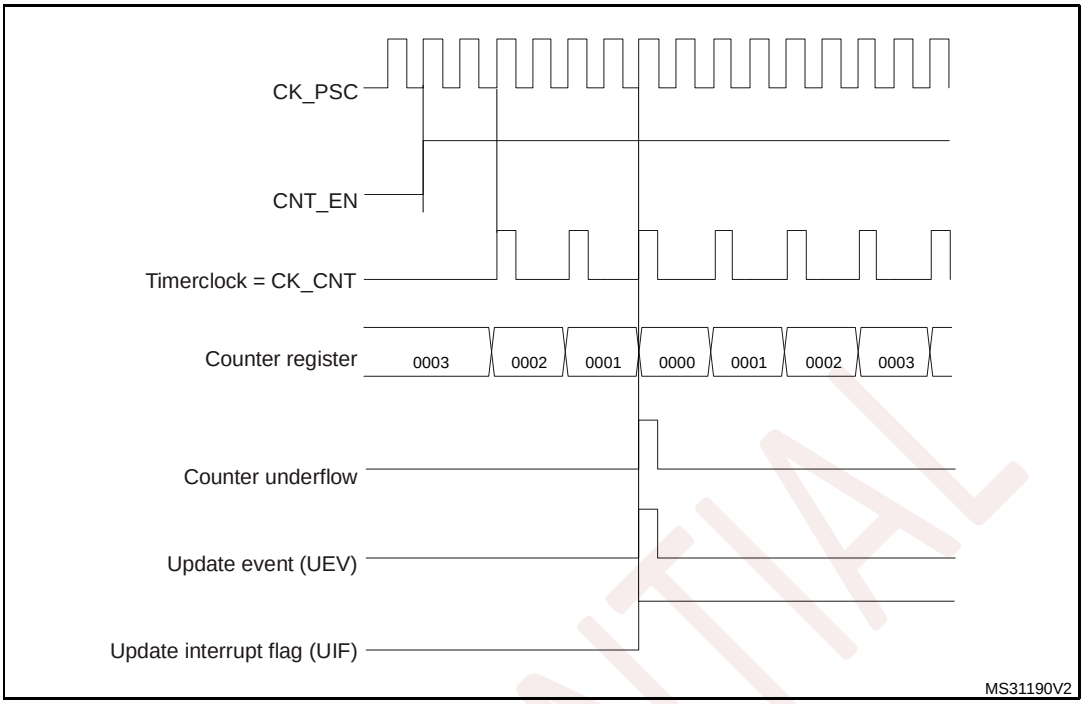
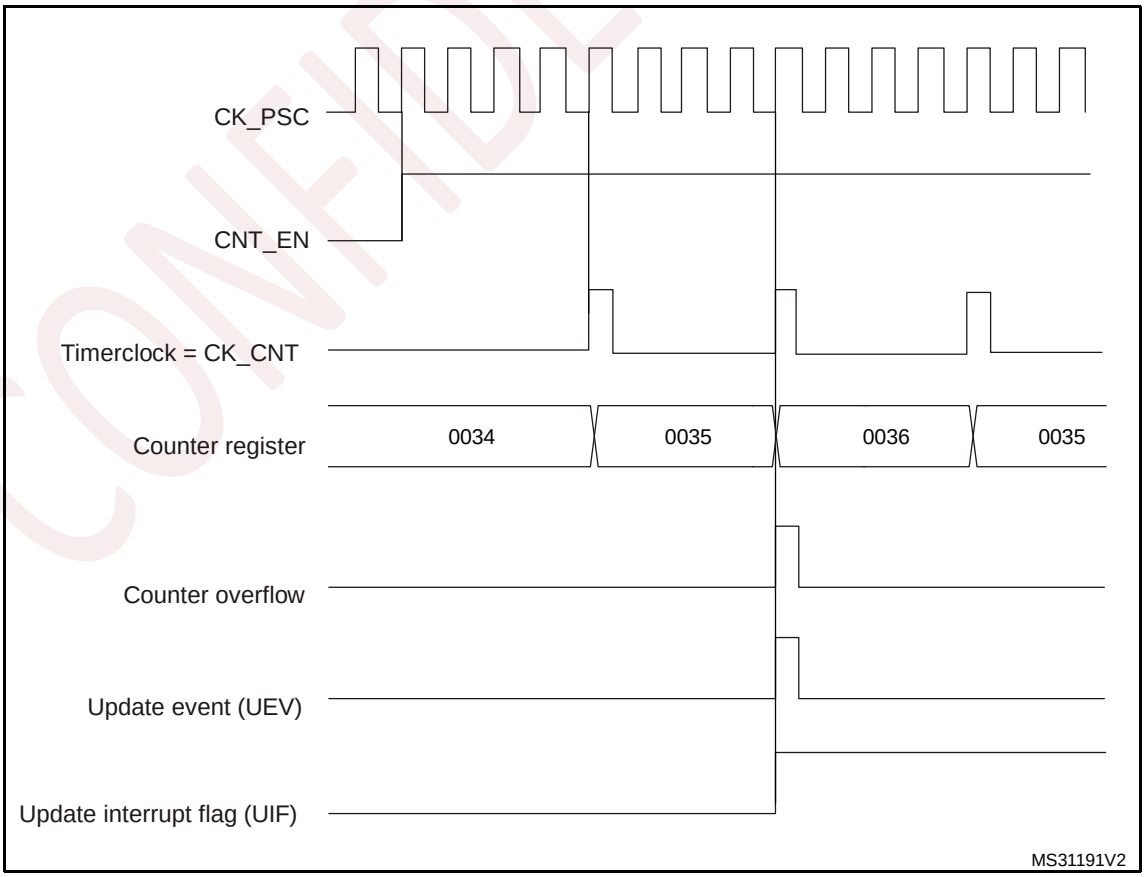


Figure17. Counter timing diagram, internal clock divided by 4, ARR=0x36



1. Center-aligned mode 2 or 3 is used with an UIF on overflow.

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

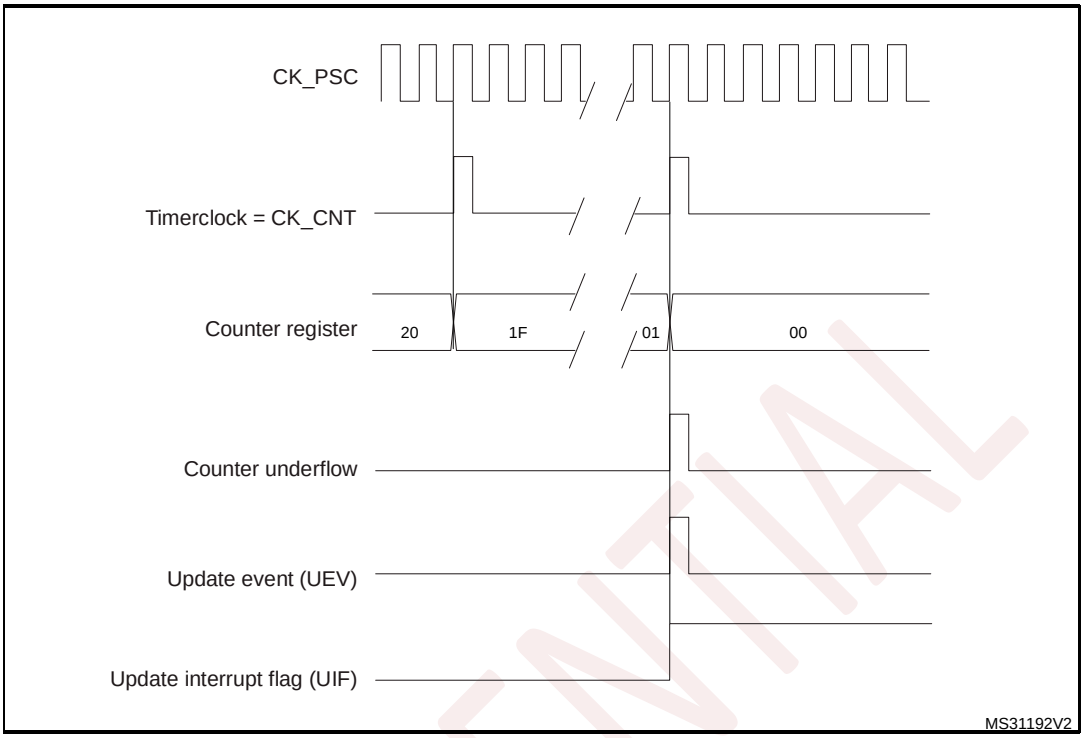


Figure 19. Counter timing diagram, update event with ARPE=1 (counter underflow)

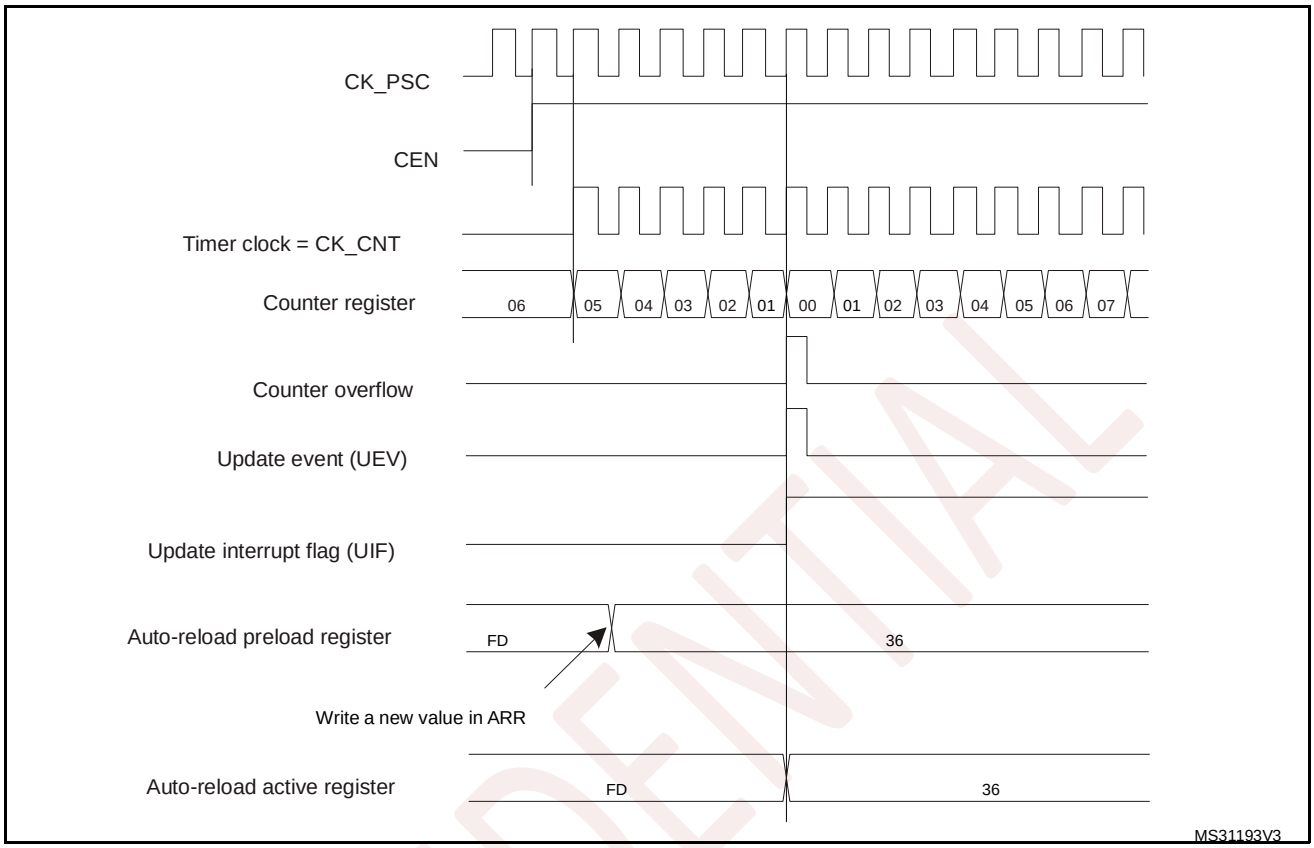
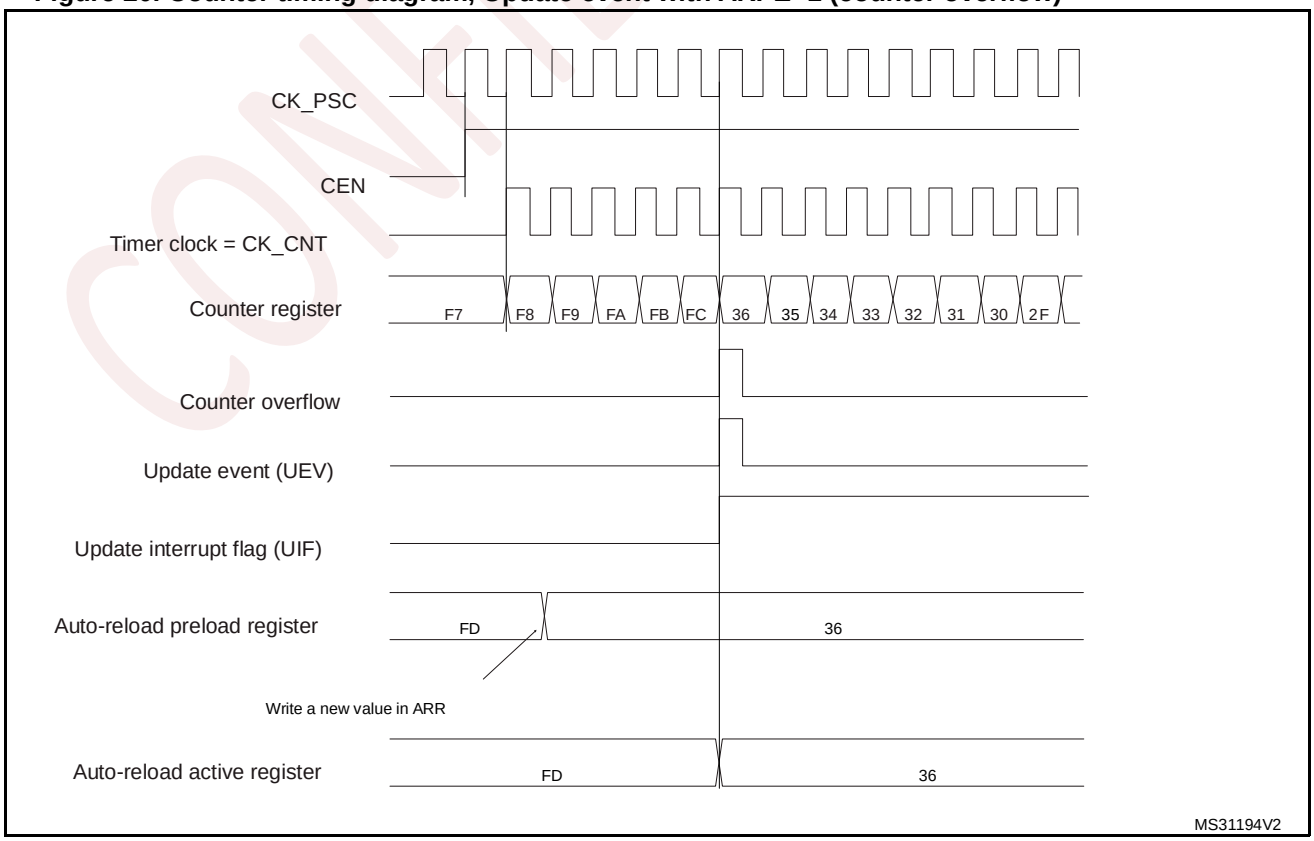


Figure 20. Counter timing diagram, Update event with ARPE=1 (counter overflow)



8.3.3 Repetition counter

Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (ARR auto-reload register, PSC prescaler register, but also CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the RCR repetition counter register.

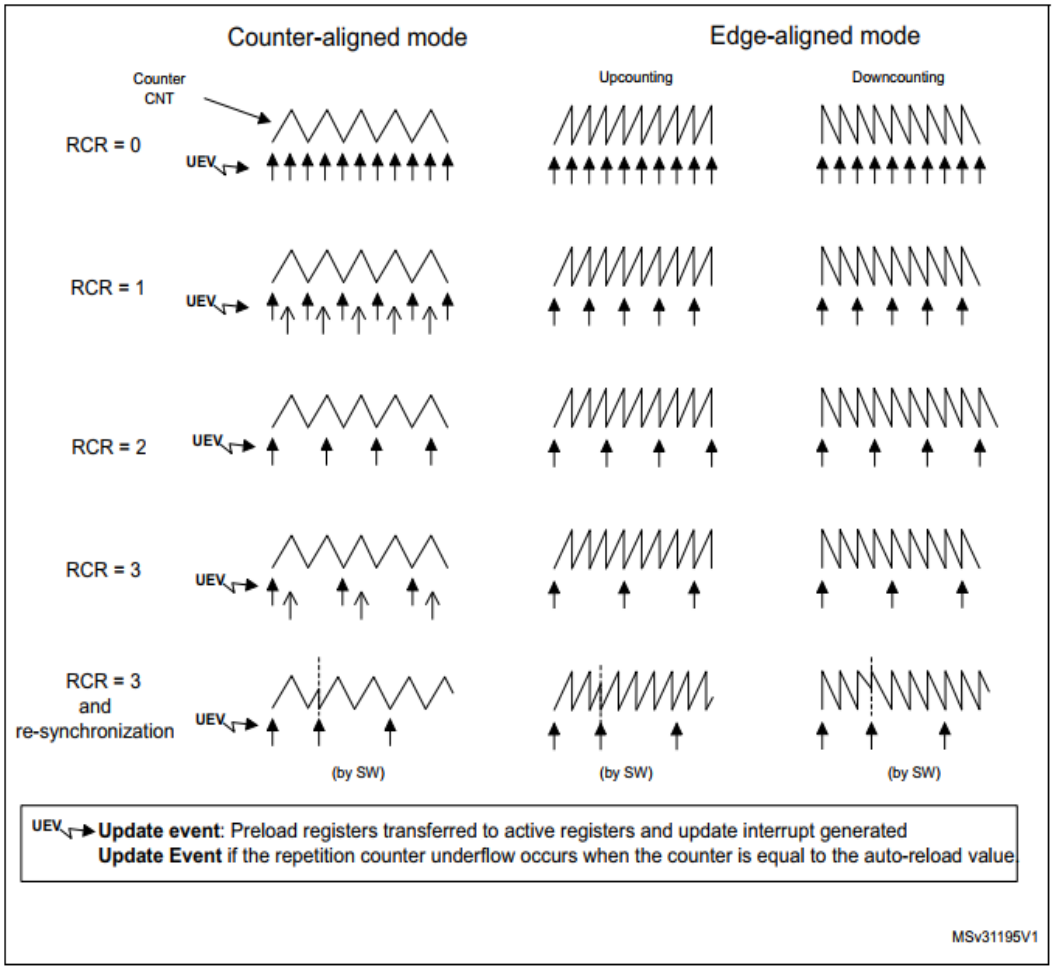
The repetition counter is decremented:

- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 128 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an auto-reload type; the repetition rate is maintained as defined by the RCR register value (refer to *Figure 21*). When the update event is generated by software (by setting the UG bit in EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the RCR register.

In center-aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was started. If the RCR was written before starting the counter, the UEV occurs on the overflow. If the RCR was written after starting the counter, the UEV occurs on the underflow. For example for RCR = 3, the UEV is generated on each 4th overflow or underflow event depending on when RCR was written.

Figure 21. Update rate examples depending on mode and RCR register settings



8.3.4 Clock selection

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

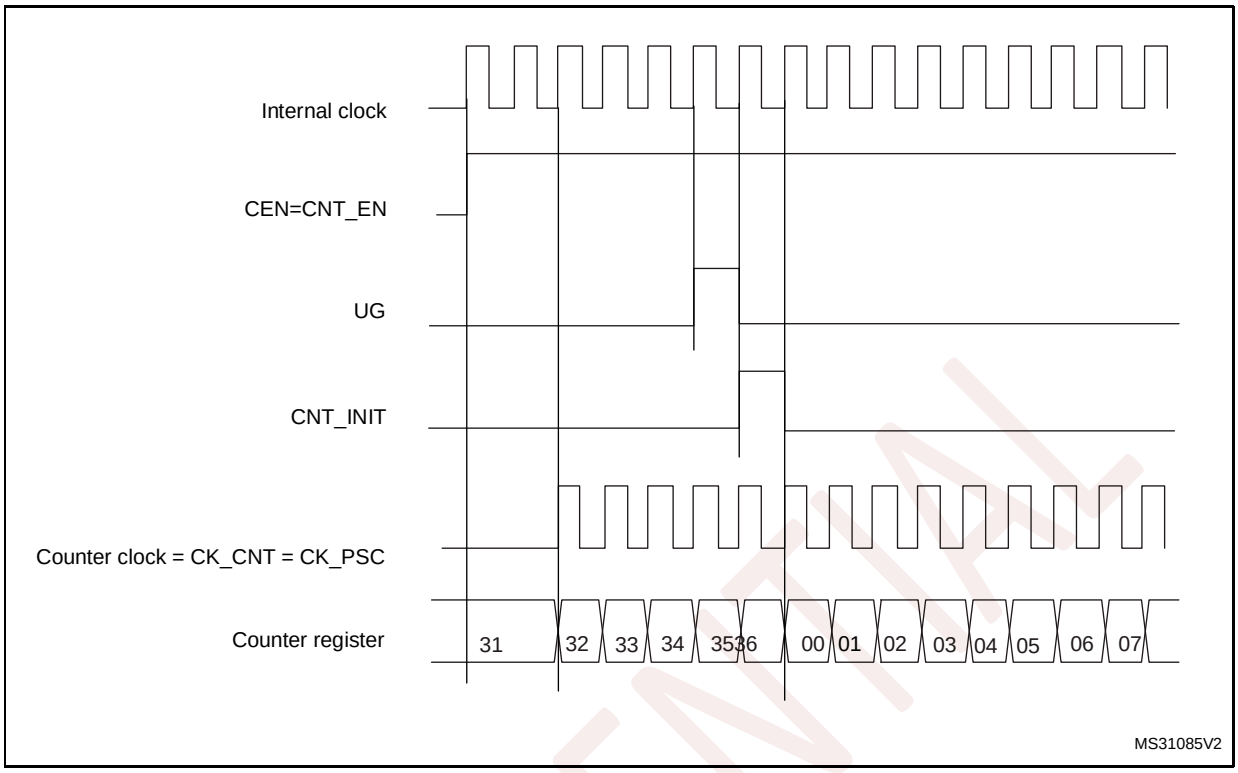
- Internal clock (CK_INT)
- External clock mode1: external input pin
- External clock mode2: external trigger input ETR
- Internal trigger inputs (ITRx): using one timer as prescaler for another timer, for example, the user can configure Timer 1 to act as a prescaler for Timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMS=000), then the CEN, DIR (in the CR1 register) and UG bits (in the EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock CK_INT.

*Figure 22*shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

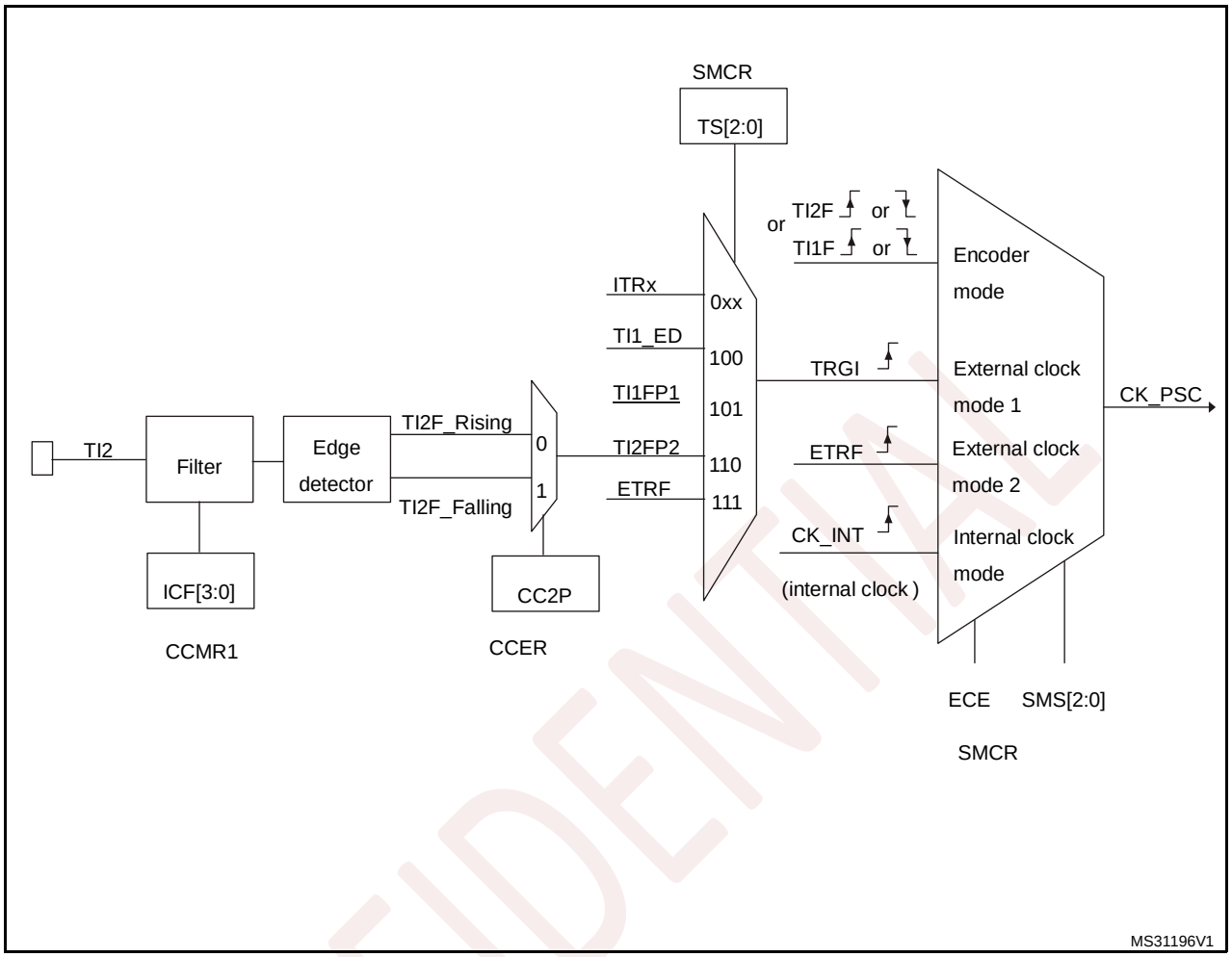
Figure 22. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

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

Figure 23. TI2 external clock connection example

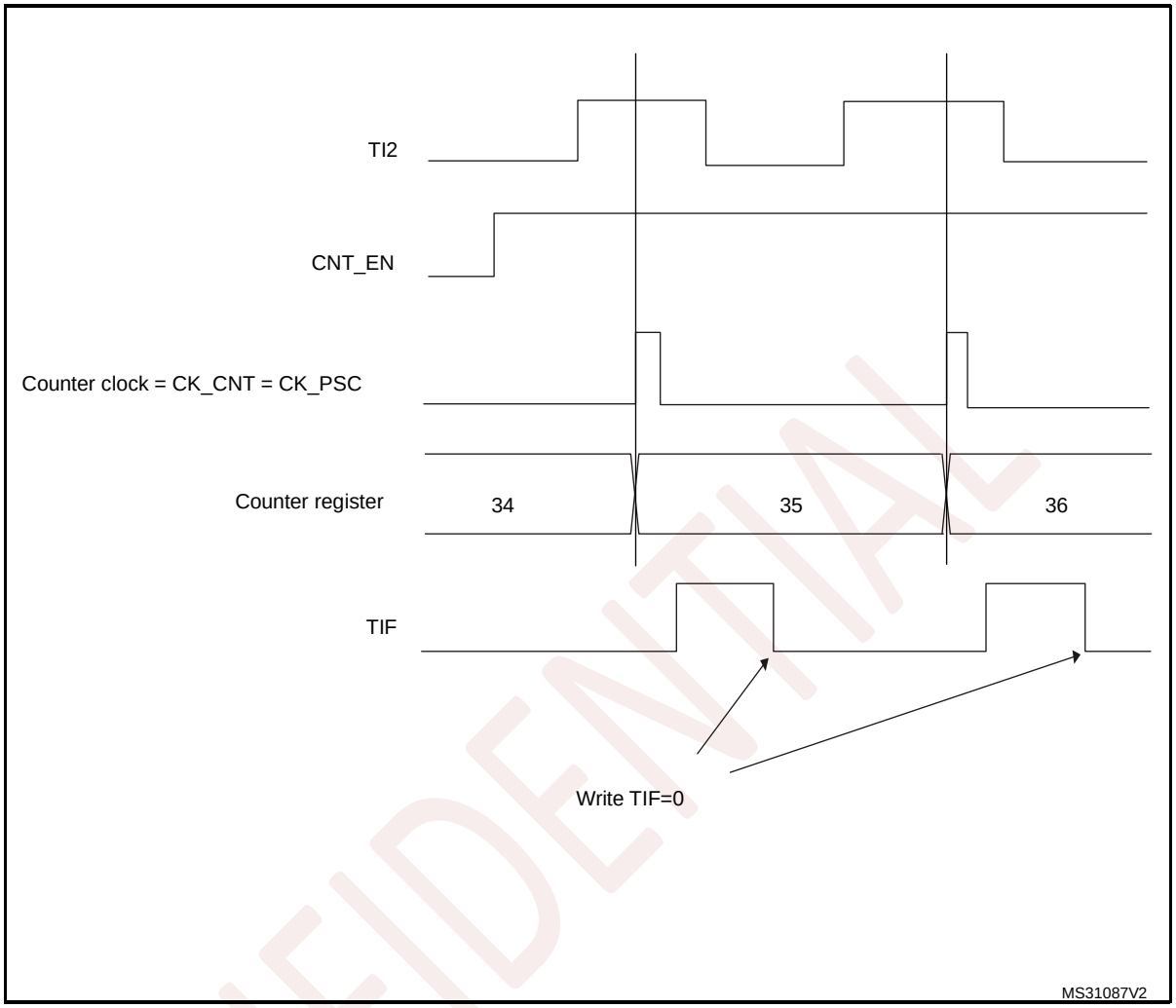


For example, to configure the upcounter to count in response to a rising edge on the TI2 input, use the following procedure:

1. Configure channel 1 to detect rising edges on the TI2 input by writing CC2S = ‘01’ in the CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the CCMR1 register (if no filter is needed, keep IC2F=0000).
3. Select rising edge polarity by writing CC2P=0 in the CCER register.
4. Configure the timer in external clock mode 1 by writing SMS=111 in the SMCR register.
5. Select TI2 as the trigger input source by writing TS=110 in the SMCR register.
6. Enable the counter by writing CEN=1 in the CR1 register.

Note: *The capture prescaler is not used for triggering, so the user does not need to configure it.*
When a rising edge occurs on TI2, the counter counts once and the TIF 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 TI2 input.

Figure 24. Control circuit in external clock mode 1



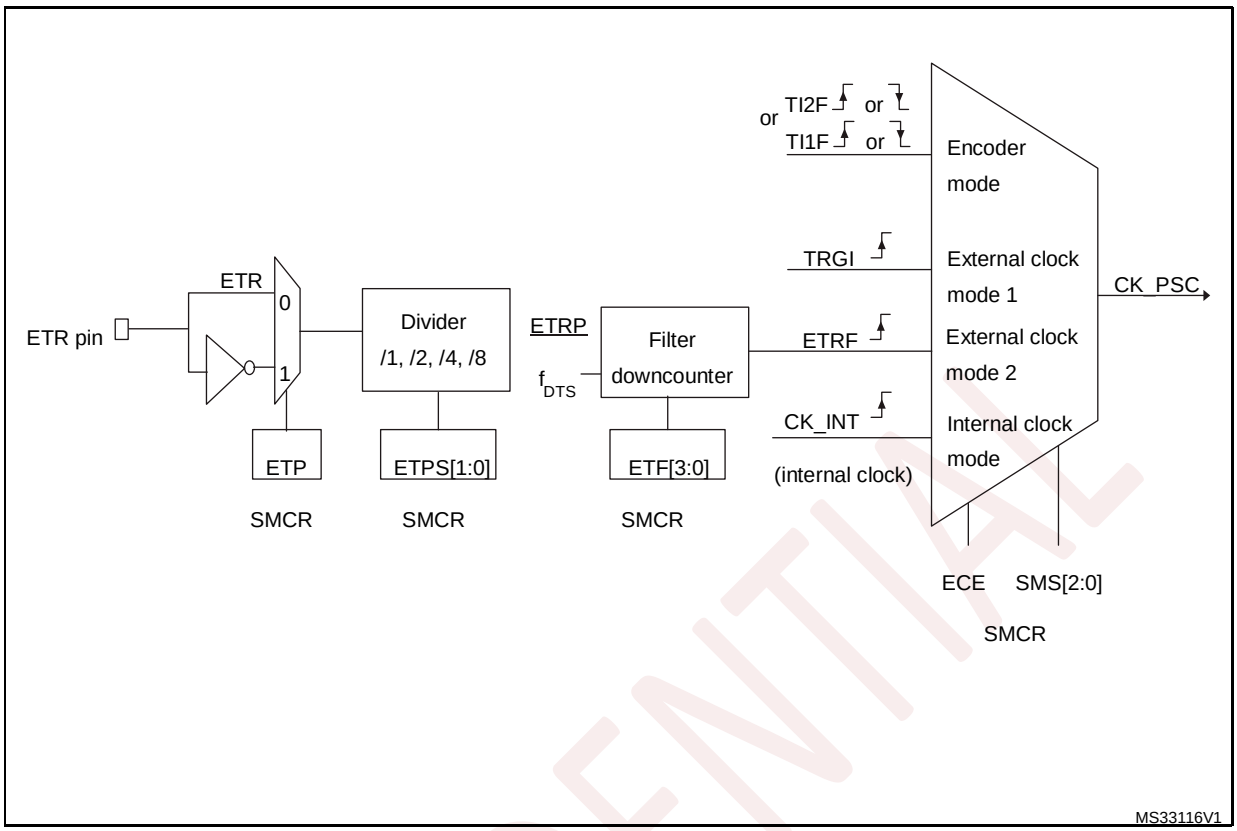
External clock source mode 2

This mode is selected by writing ECE=1 in the SMCR register.

The counter can count at each rising or falling edge on the external trigger input ETR.

Figure 25 gives an overview of the external trigger input block.

Figure 25. External trigger input block



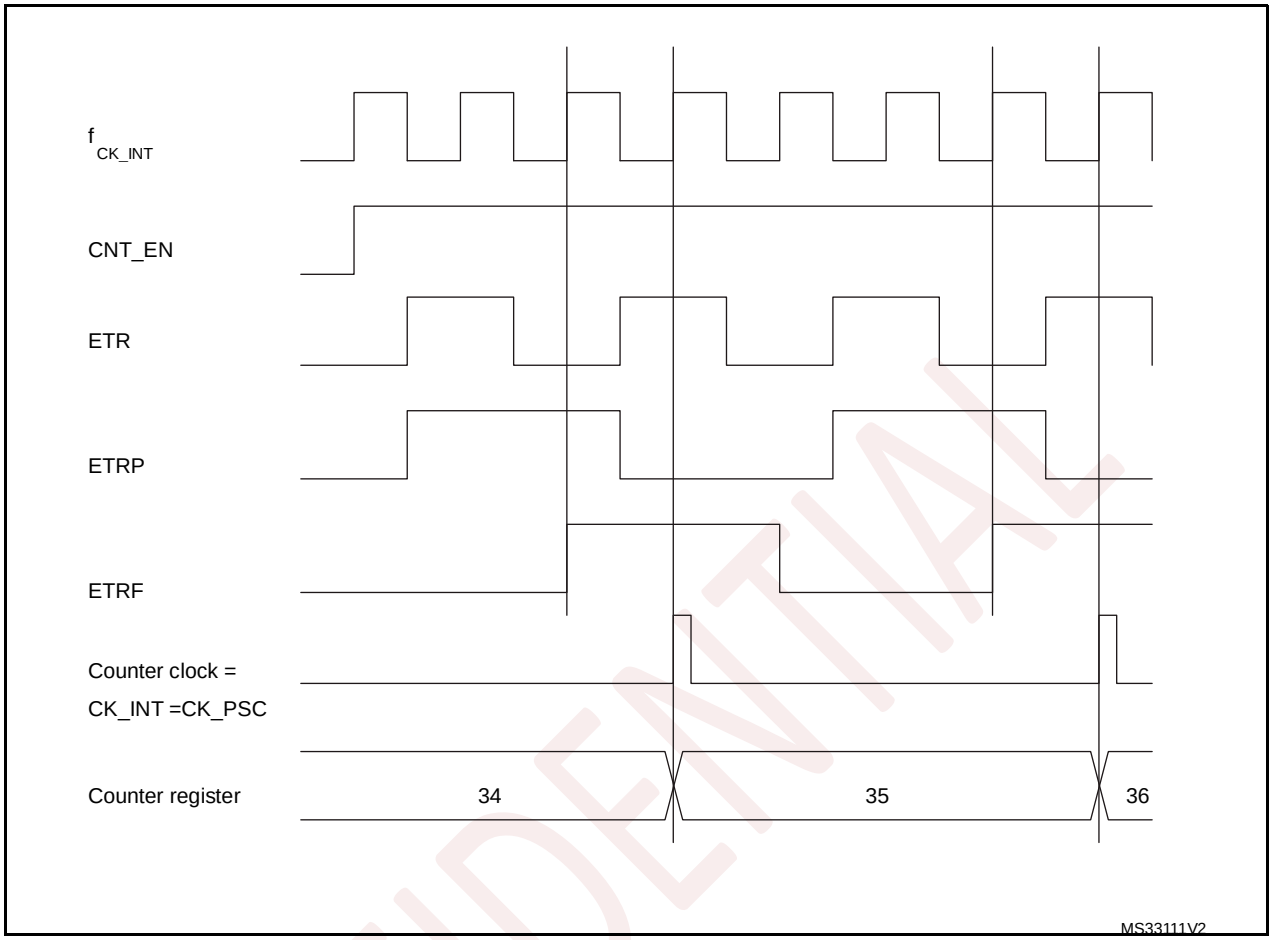
For example, to configure the upcounter to count each 2 rising edges on ETR, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0]=0000 in the SMCR register.
2. Set the prescaler by writing ETPS[1:0]=01 in the SMCR register
3. Select rising edge detection on the ETR pin by writing ETP=0 in the SMCR register
4. Enable external clock mode 2 by writing ECE=1 in the SMCR register.
5. Enable the counter by writing CEN=1 in the 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 26. Control circuit in external clock mode 2



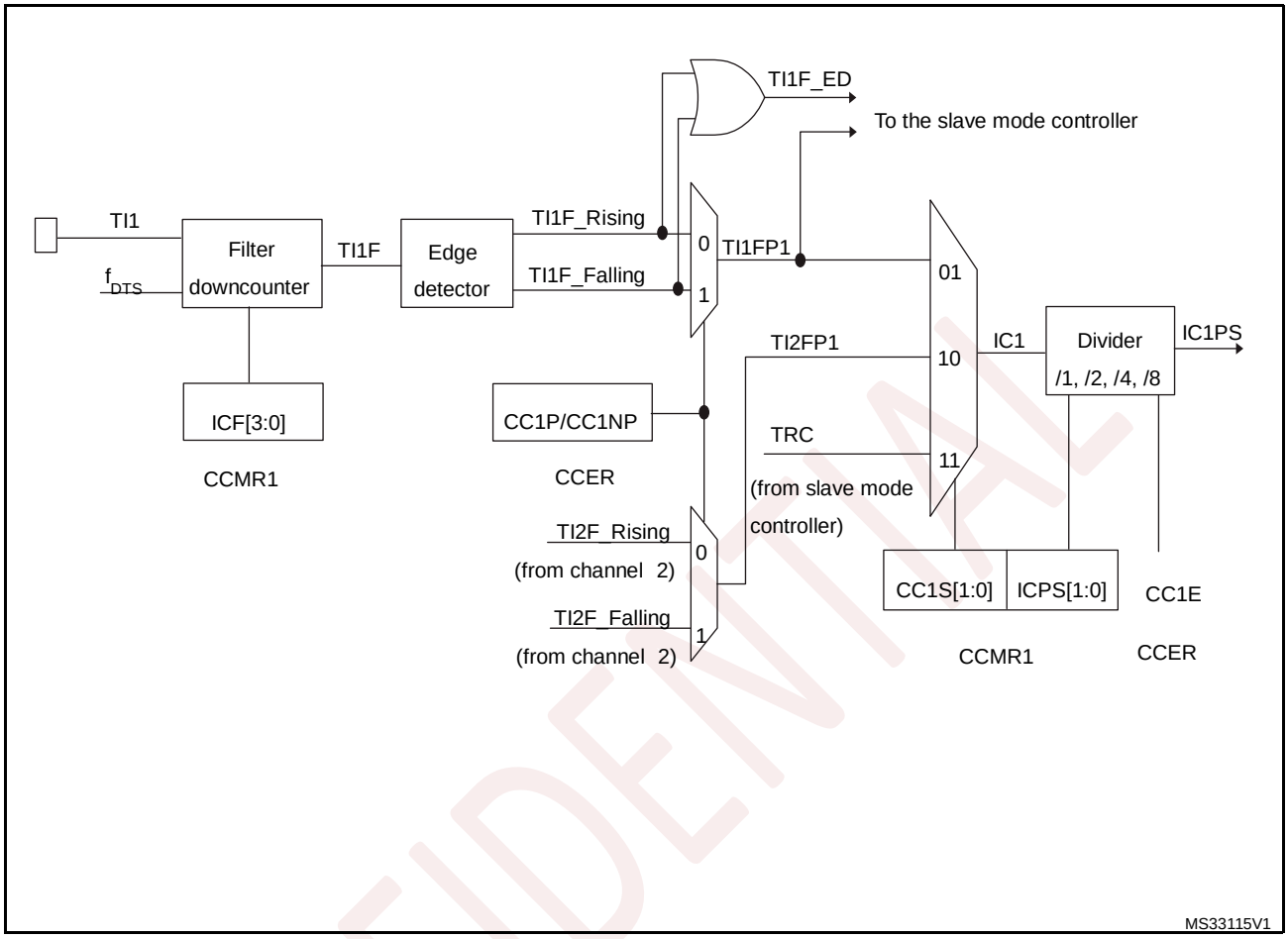
8.3.5 Capture/compare channels

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

Figure 27 to *Figure 30* give an overview of one Capture/Compare channel.

The input stage samples the corresponding TIx input to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 27. Capture/compare channel (example: channel 0 input stage)



The output stage generates an intermediate waveform that is then used for reference: OCxRef (active high). The polarity acts at the end of the chain.

Figure 28. Capture/compare channel 0 main circuit

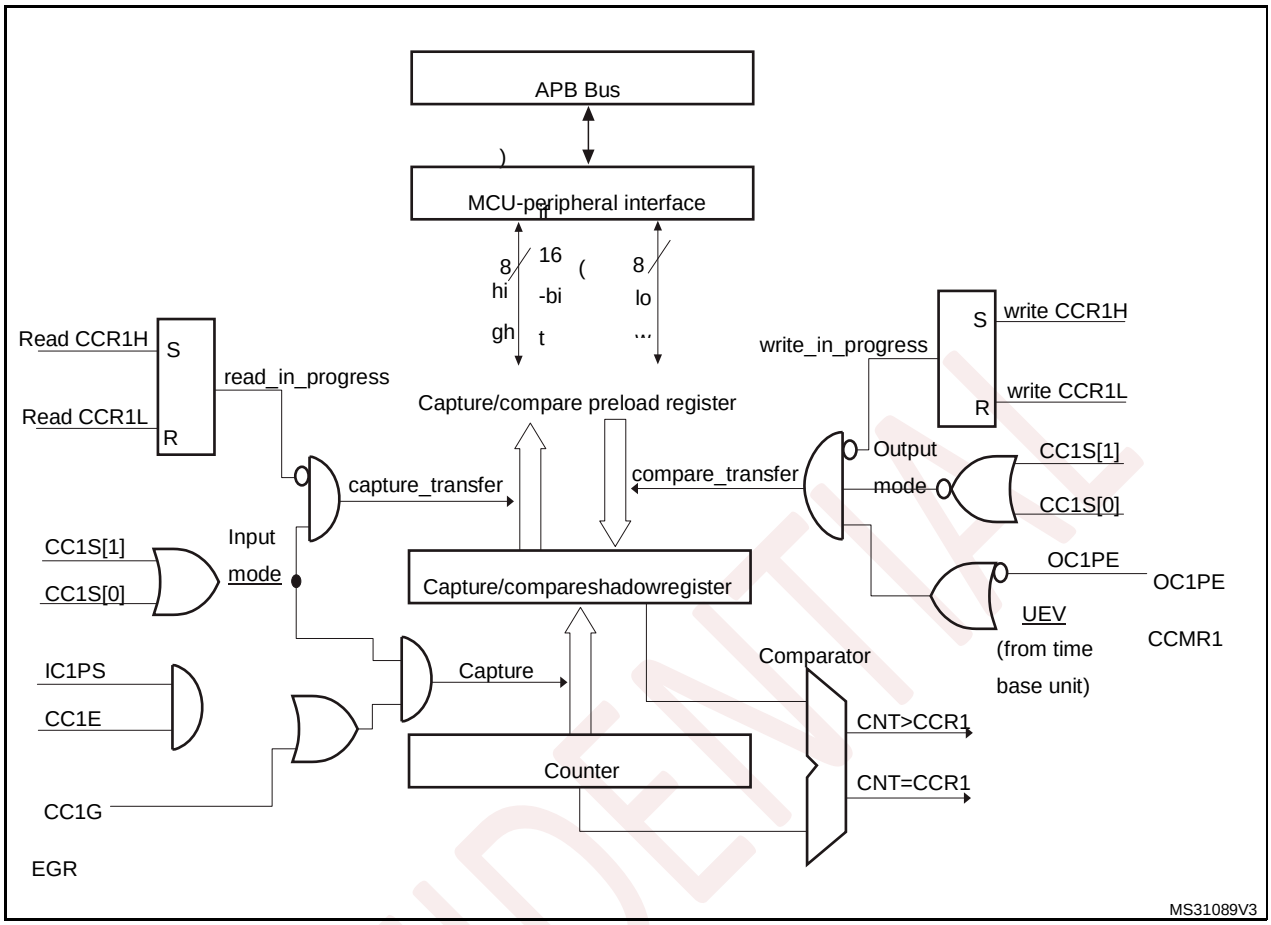


Figure 29. Output stage of capture/compare channel (channel 1 to3)

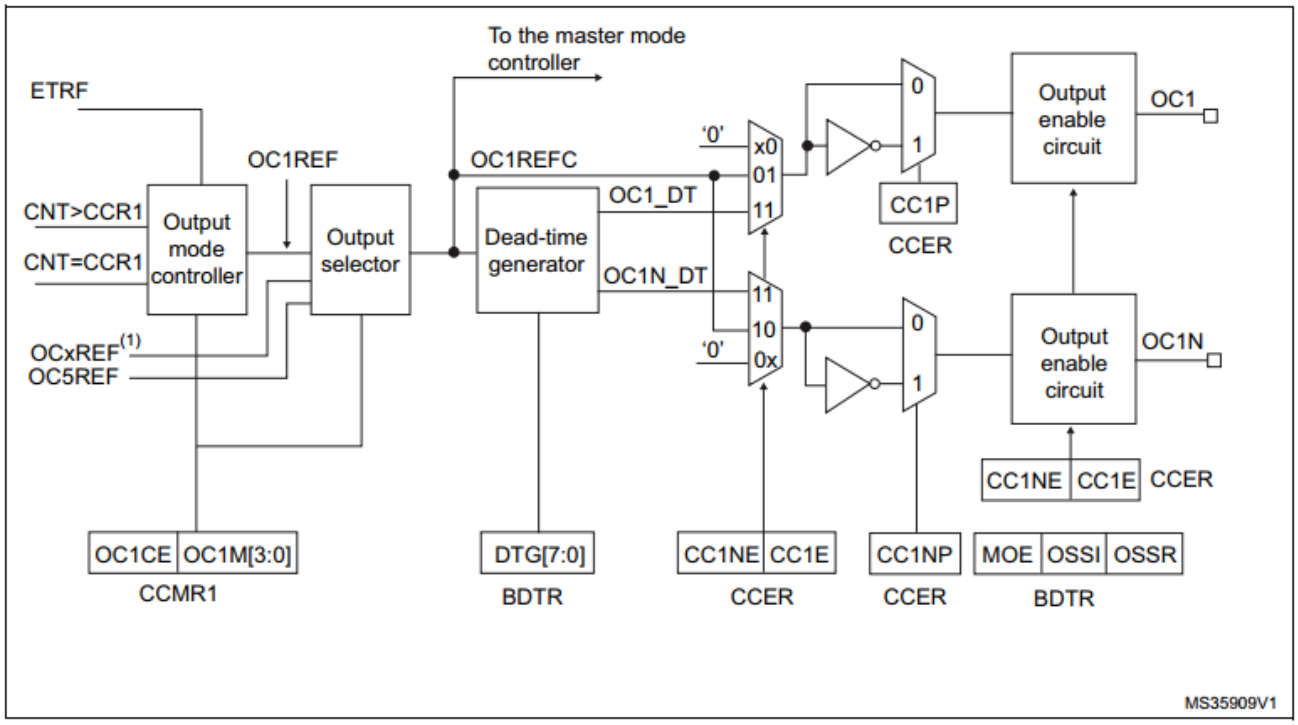
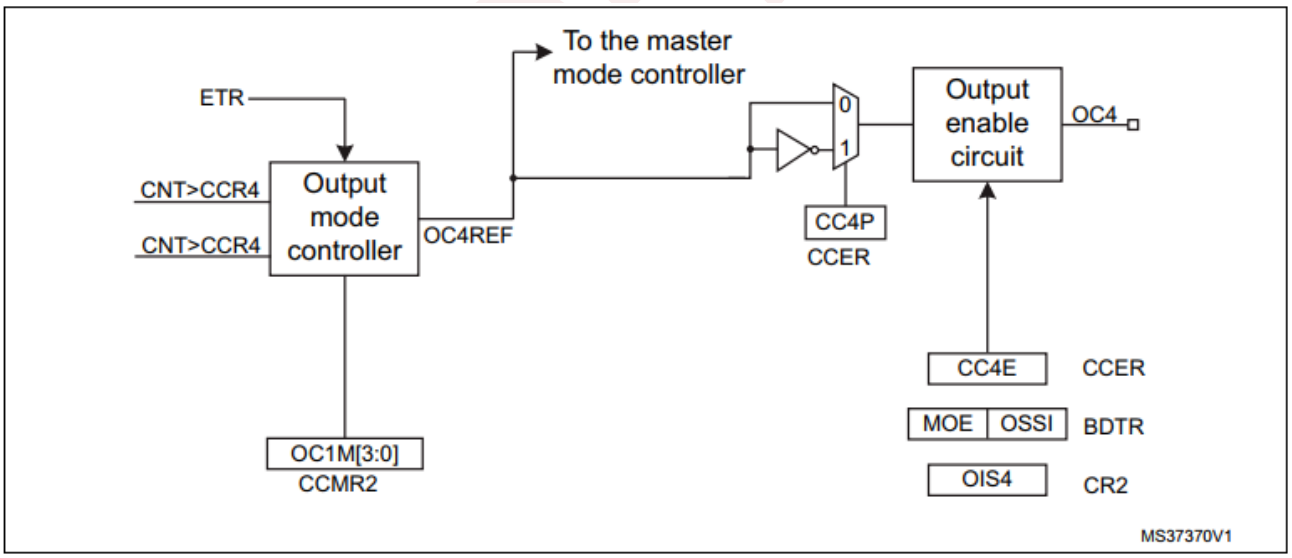


Figure 30. Output stage of capture/compare channel (channel 4)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

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

8.3.6 Input capture mode

In Input capture mode, the Capture/Compare registers (CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (SR register) is set. CCxIF can be cleared by software by writing it to '0' or by reading the captured data stored in the CCRx register. CCxOF is cleared when written to '0'.

The following example shows how to capture the counter value in CCR1 when TI1 input rises. To do this, use the following procedure:

- Select the active input: CCR1 must be linked to the TI1 input, so write the CC1S bits to 01 in the CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the CCR1 register becomes read-only.
- Program the needed input filter duration with respect to the signal connected to the timer (by programming ICxF bits in the CCMRx register if the input is a TIx input). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on TI1 when 8 consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the CCMR1 register.
- Select the edge of the active transition on the TI1 channel by writing CC1P bit to 0 in the CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to '00' in the CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the DIER register, and/or the DMA request by setting the CC1DE bit in the DIER register.

When an input capture occurs:

- The CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which could happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the

corresponding CCxG bit in the EGR register.

8.3.7 PWM input mode

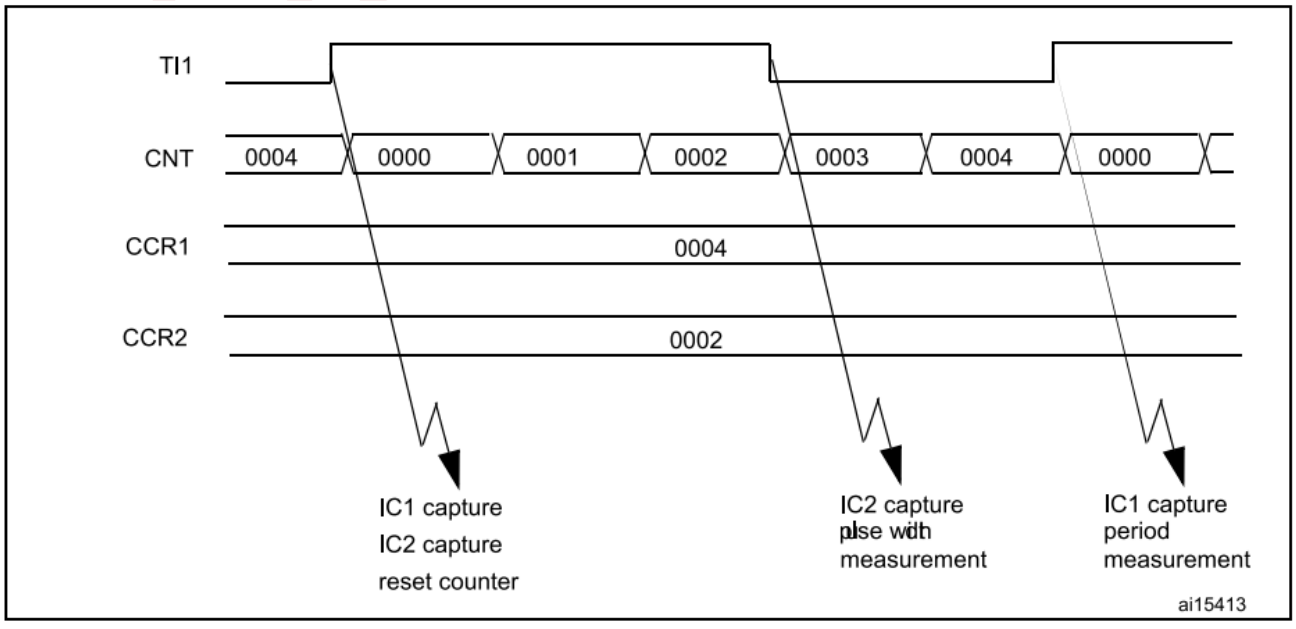
This mode is a particular case of input capture mode. The procedure is the same except:

- Two ICx signals are mapped on the same TIx input.
- These 2 ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, user can measure the period (in CCR1 register) and the duty cycle (in CCR2 register) of the PWM applied on TI1 using the following procedure (depending on CK_INT frequency and prescaler value):

- Select the active input for CCR1: write the CC1S bits to 01 in the CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used both for capture in CCR1 and counter clear): write the CC1P bit to '0' (active on rising edge).
- Select the active input for CCR2: write the CC2S bits to 10 in the CCMR1 register (TI1 selected).
- Select the active polarity for TI1FP2 (used for capture in CCR2): write the CC2P bit to '1' (active on falling edge).
- Select the valid trigger input: write the TS bits to 101 in the SMCR register (TI1FP1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 100 in the SMCR register.
- Enable the captures: write the CC1E and CC2E bits to '1' in the CCER register.

Figure 31. PWM input mode timing



1. The PWM input mode can be used only with the CH1/CH2 signals due to the fact that only TI1FP1 and TI2FP2 are connected to the slave mode controller.

8.3.8 Forced output mode

In output mode (CCxS bits = 00 in the CCMRx register), each output compare signal (OCxREF and then OCx/OCxN) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (OCXREF/OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding CCMRx register. Thus OCXREF is forced high (OCxREF is always active high) and OCx get opposite value to CCxP polarity bit.

For example: CCxP=0 (OCx active high) => OCx is forced to high level.

The OCxREF signal can be forced low by writing the OCxM bits to 100 in the CCMRx register.

Anyway, the comparison between the CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

8.3.9 Output compare mode

This function is used to control an output waveform or indicating 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 (OCxM bits in the CCMRx register) and the output polarity (CCxP bit in the CCER register). The output pin can keep its level (OCXM=000), be set active (OCxM=001), be set inactive (OCxM=010) or can toggle (OCxM=011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the DIER register, CCDS bit in the CR2 register for the DMA request selection).

The CCRx registers can be programmed with or without preload registers using the OCxPE bit in the CCMRx register.

In output compare mode, the update event UEV has no effect on OCxREF and OCx

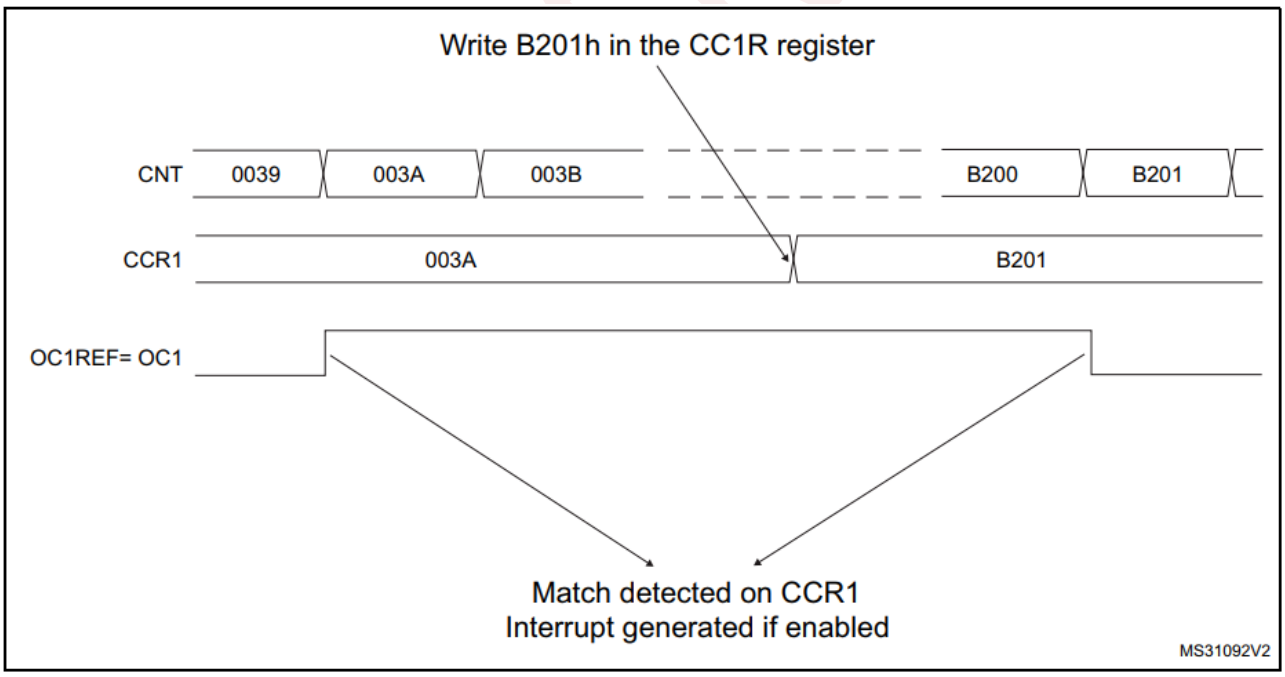
output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One Pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the ARR and CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle OCx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register – Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the CR1 register.

The CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE='0', else CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 32*.

Figure 32. Output compare mode, toggle on OC1.



8.3.10 PWM mode

Pulse Width Modulation mode allows generating a signal with a frequency determined by the value of the ARR register and a duty cycle determined by the value of the CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxM bits in the CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the CCMRx register, and eventually the auto-reload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user must initialize all the registers by setting the UG bit in the EGR register.

OCx polarity is software programmable using the CCxP bit in the CCER register. It can be programmed as active high or active low. OCx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI and OSSR bits (CCER and BDTR registers). Refer to the CCER register description for more details.

In PWM mode (1 or 2), CNT and CCRx are always compared to determine whether CCRx
CNT or CNT CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the CR1 register.

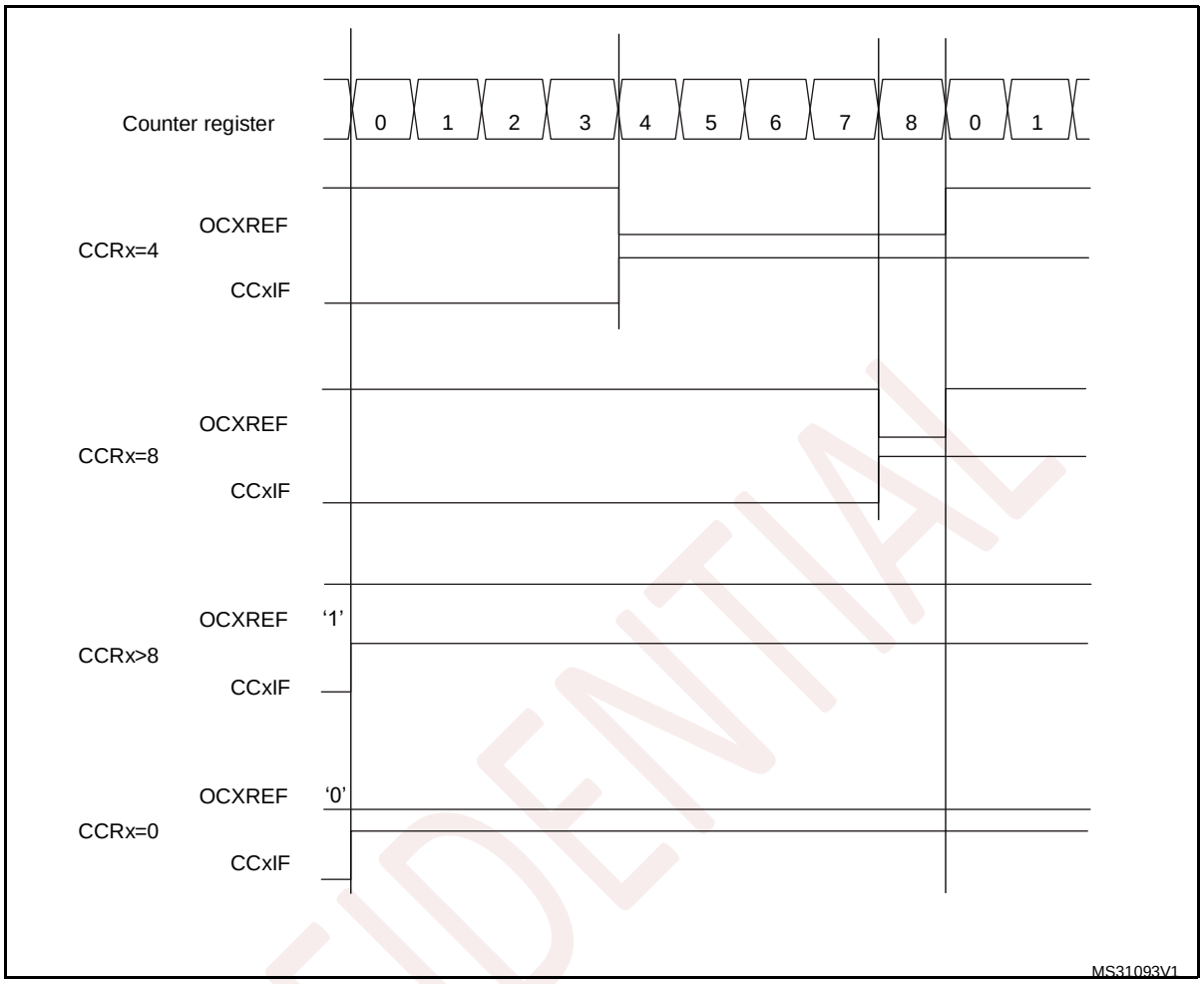
PWM edge-aligned mode

Upcounting configuration

Upcounting is active when the DIR bit in the CR1 register is low. Refer to *Upcounting mode*.

In the following example, we consider PWM mode 1. The reference PWM signal OCxREF is high as long as CNT < CCRx else it becomes low. If the compare value in CCRx is greater than the auto-reload value (in ARR) then OCxREF is held at '1'. If the compare value is 0 then OCxRef is held at '0'. *Figure 33* shows some edge-aligned PWM waveforms in an example where ARR=8.

Figure 33. Edge-aligned PWM waveforms (ARR=8)



Downcounting configuration

Downcounting is active when DIR bit in CR1 register is high. Refer to *Downcounting mode*

In PWM mode 1, the reference signal OCxRef is low as long as

CNT > CCRx else it becomes high. If the compare value in CCRx is greater than the auto-reload value in ARR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

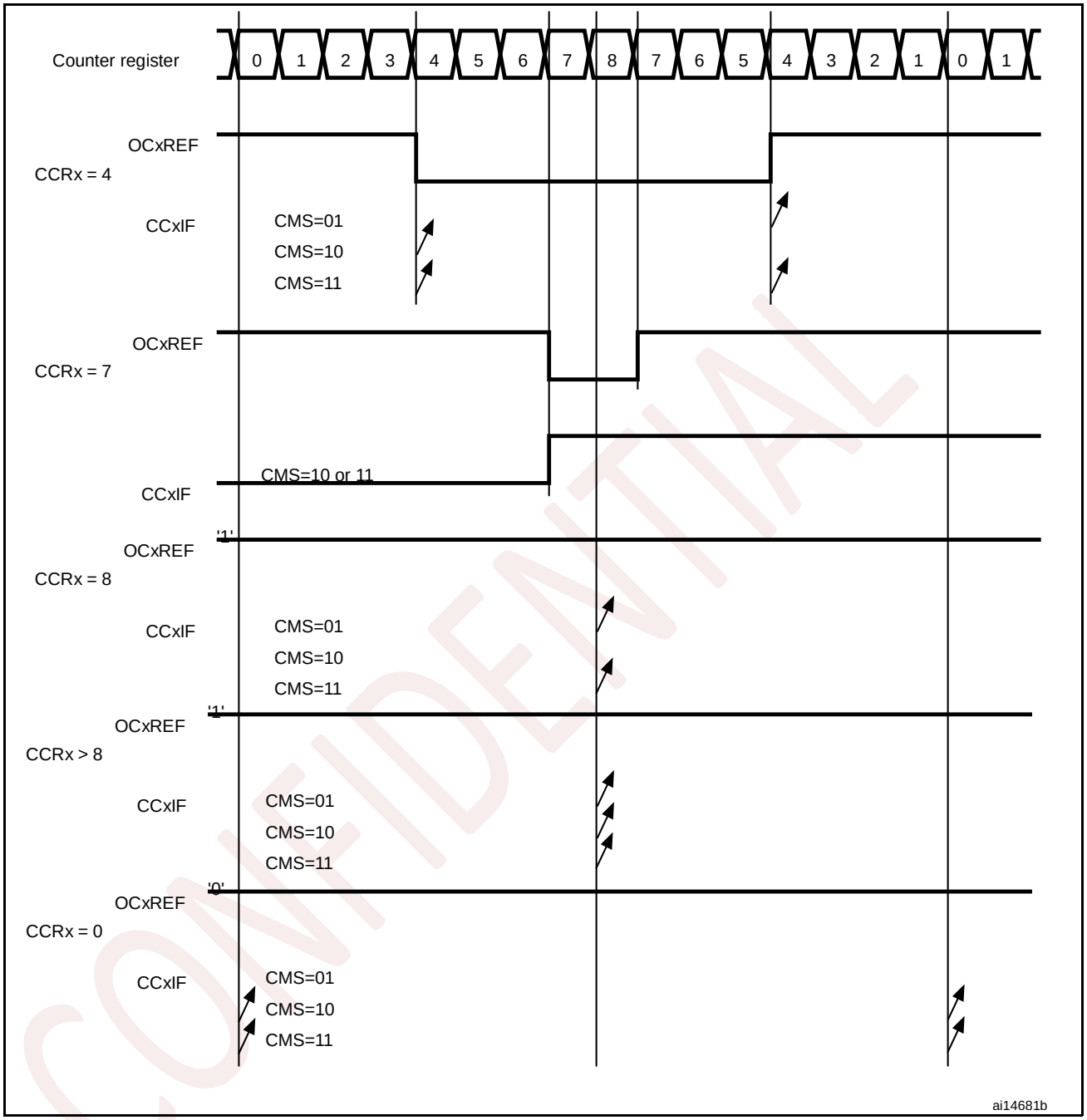
PWM center-aligned mode

Center-aligned mode is active when the CMS bits in CR1 register are different from ‘00’ (all the remaining configurations having the same effect on the OCxRef/OCx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit (DIR) in the CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 34 shows some center-aligned PWM waveforms in an example where:

- ARR=8,
- PWM mode is the PWM mode 1,
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS=01 in CR1 register.

Figure 34. 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. It means that the counter counts up or down depending on the value written in the DIR bit in the CR1 register. Moreover, the DIR and 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 if the user writes a value in the counter greater than the auto-reload value (CNT>ARR). For example, if the counter was counting up, it will continue to count up.
  - The direction is updated if the user writes 0 or write the ARR value in the counter but no Update Event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the EGR register) just before starting the counter and not to write the counter while it is running.

8.3.11 Complementary outputs and dead-time insertion

The advanced-control timers can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjust it depending on the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays due to power switches...)

User can select the polarity of the outputs (main output OCx or complementary OCxN) independently for each output. This is done by writing to the CCxP and CCxNP bits in the CCER register.

The complementary signals OCx and OCxN are activated by a combination of several control bits: the CCxE and CCxNE bits in the CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the BDTR and CR2 registers. In particular, the dead-time is activated when switching to the IDLE state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. DTG[7:0] bits of the BDTR register are used to control the dead-time generation for all channels. From a reference waveform OCxREF, it generates 2 outputs OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The OCxN output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (OCx or OCxN) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal OCxREF. (we suppose CCxP=0, CCxNP=0, MOE=1, CCxE=1 and CCxNE=1 in these examples)

Figure 35. Complementary output with dead-time insertion.

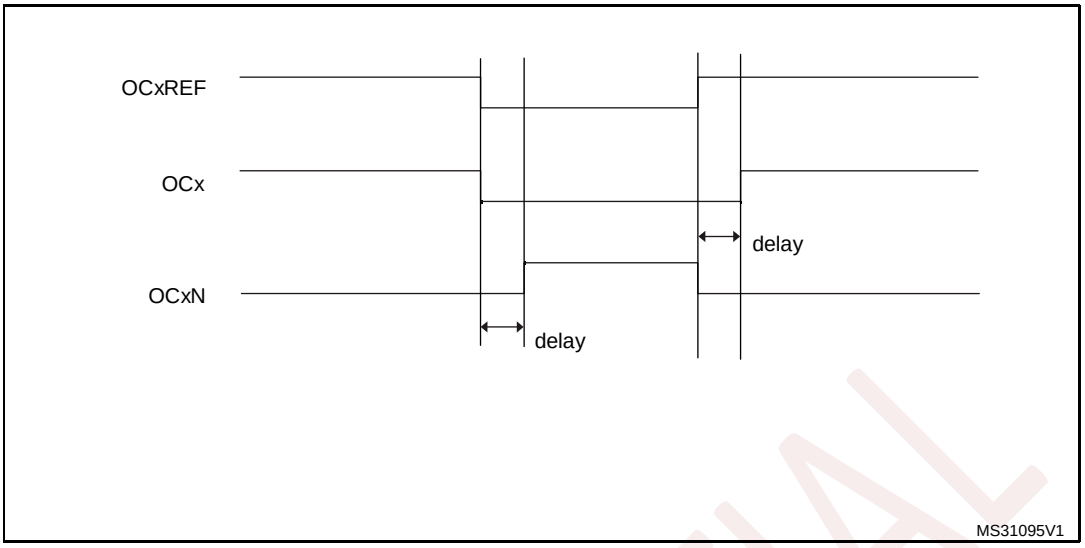


Figure 36. Dead-time waveforms with delay greater than the negative pulse.

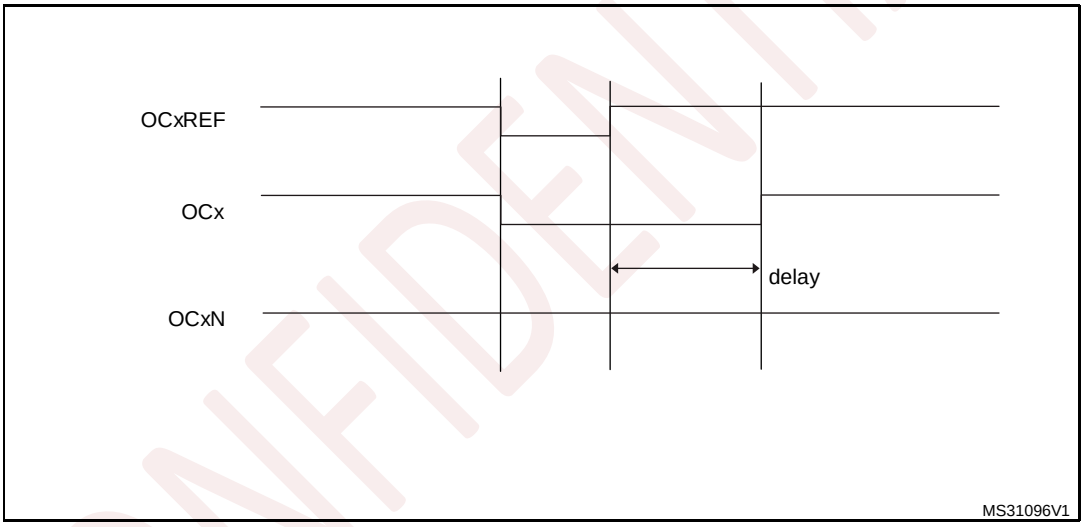
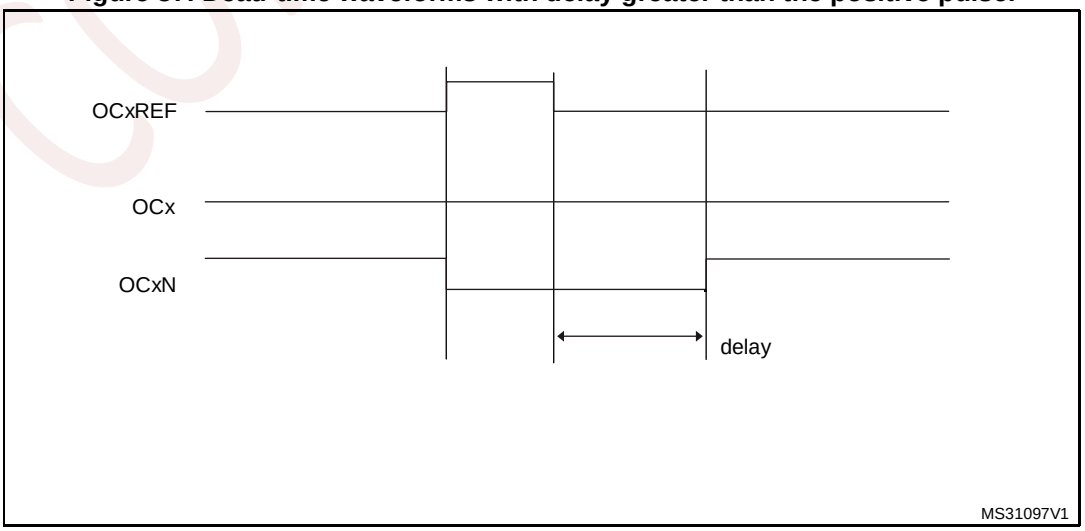


Figure 37. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the BDTR register. Refer to *break and dead-time register (BDTR)* for delay calculation.

Re-directing OCxREF to OCx or OCxN

In output mode (forced, output compare or PWM), OCxREF can be re-directed to the OCx output or to OCxN output by configuring the CCxE and CCxNE bits in the CCER register.

This allows the user to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: *When only OCxN is enabled (CCxE=0, CCxNE=1), it is not complemented and becomes active as soon as OCxREF is high. For example, if CCxNP=0 then OCxN=OCxRef. On the other hand, when both OCx and OCxN are enabled (CCxE=CCxNE=1) OCx becomes active when OCxREF is high whereas OCxN is complemented and becomes active when OCxREF is low.*

8.3.12 Using the break function

When using the break function, the output enable signals and inactive levels are modified according to additional control bits (MOE, OSSI and OSSR bits in the BDTR register, OISx and OISxN bits in the CR2 register). In any case, the OCx and OCxN outputs cannot be set both to active level at a given time..

The break source can be either the break input pin or a clock failure event, generated by the Clock Security System (CSS), from the Reset Clock Controller. For further information on the Clock Security System.

When exiting from reset, the break circuit is disabled and the MOE bit is low. User can enable the break function by setting the BKE bit in the BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait 1 APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is written to 1 whereas it was low, a delay (dummy instruction) must be inserted before reading it correctly. This is because the user writes an asynchronous signal, but reads a synchronous signal.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state or in reset state (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the CR2 register as soon as MOE=0. If OSSI=0 then the timer releases the enable output else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
  - If OSSI=0 then the timer releases the enable outputs else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the SR register) is set. An interrupt can be generated if the BIE bit in the DIER register is set. A DMA request can be sent if the BDE bit in the DIER register is set.
- If the AOE bit in the BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written to '1' again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: *The break inputs is acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.*

The break can be generated by the BRK input which has a programmable polarity and an enable bit BKE in the BDTR register.

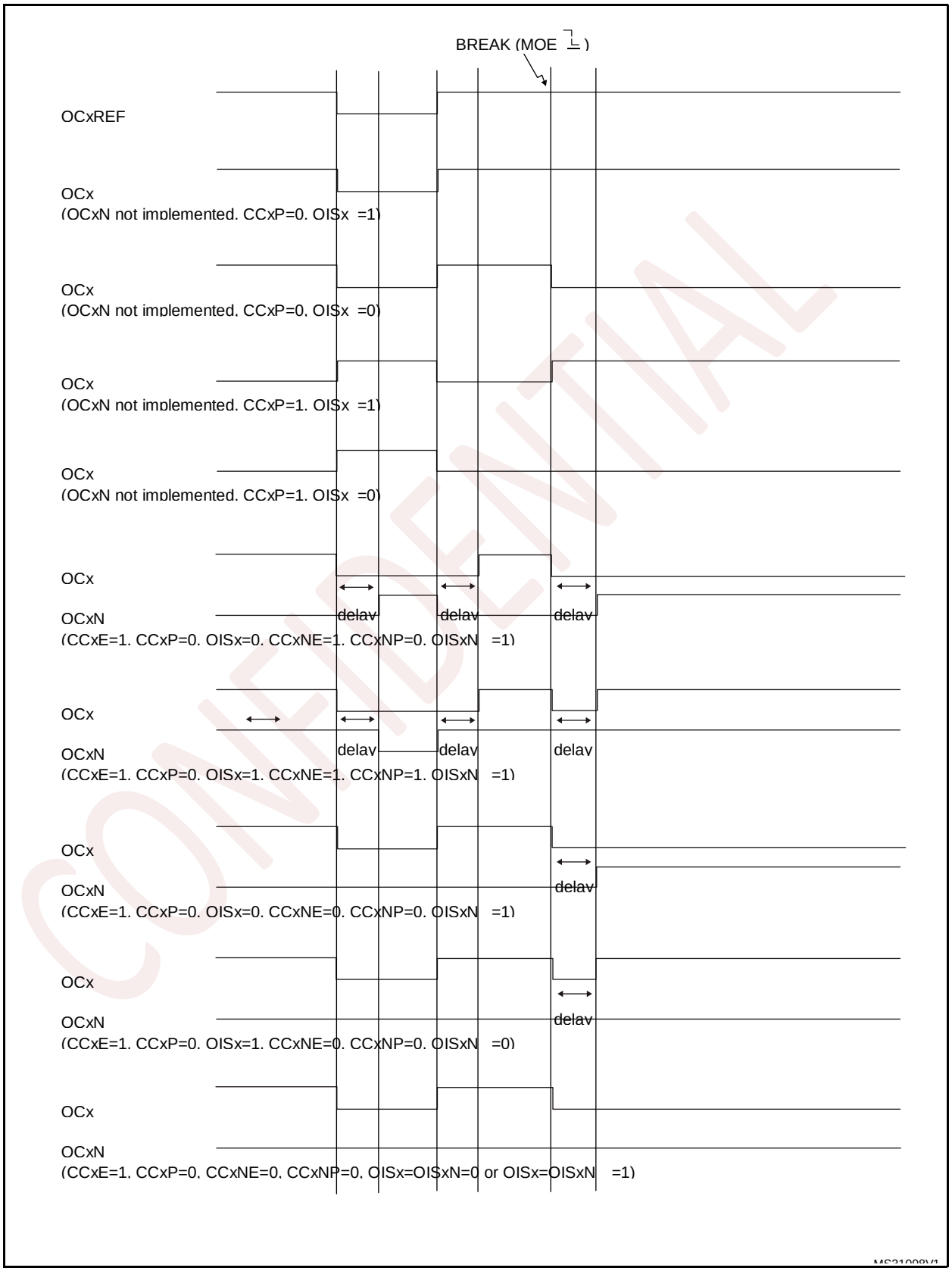
There are two solutions to generate a break:

- By using the BRK input which has a programmable polarity and an enable bit BKE in the BDTR register
- By software through the BG bit of the EGR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxM configurations, break enable and polarity). The user can choose from three levels of protection selected by the LOCK bits in the BDTR register. Refer to *Section : break and dead-time register (BDTR)*. The LOCK bits can be written only once after an MCU reset.

Figure 38 shows an example of behavior of the outputs in response to a break.

Figure 38. Output behavior in response to a break.



8.3.13 Clearing the OCxREF signal on an external event

The OCxREF signal for a given channel can be driven Low by applying a High level to the ETRF input (OCxCE enable bit of the corresponding CCMRx register set to ‘1’). The OCxREF signal remains Low until the next update event, UEV, occurs.

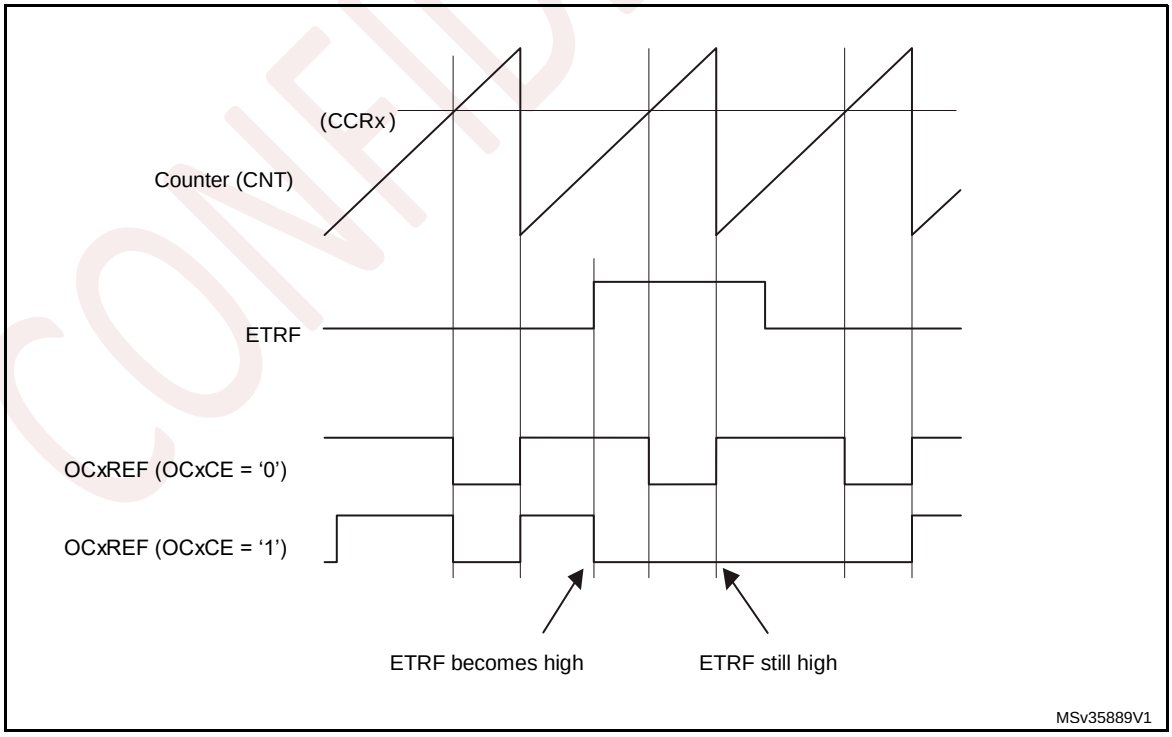
This function can only be used in output compare and PWM modes, and does not work in forced mode.

For example, the ETR signal can be connected to the output of a comparator to be used for current handling. In this case, the ETR must be configured as follow:

1. The External Trigger Prescaler should be kept off: bits ETPS[1:0] of the SMCR register set to ‘00’.
2. The external clock mode 2 must be disabled: bit ECE of the SMCR register set to ‘0’.
3. The External Trigger Polarity (ETP) and the External Trigger Filter (ETF) can be configured according to the user needs.

*Figure 39*shows the behavior of the OCxREF signal when the ETRF Input becomes High, for both values of the enable bit OCxCE. In this example, the timer is programmed in PWM mode.

Figure 39. Clearing OCxREF



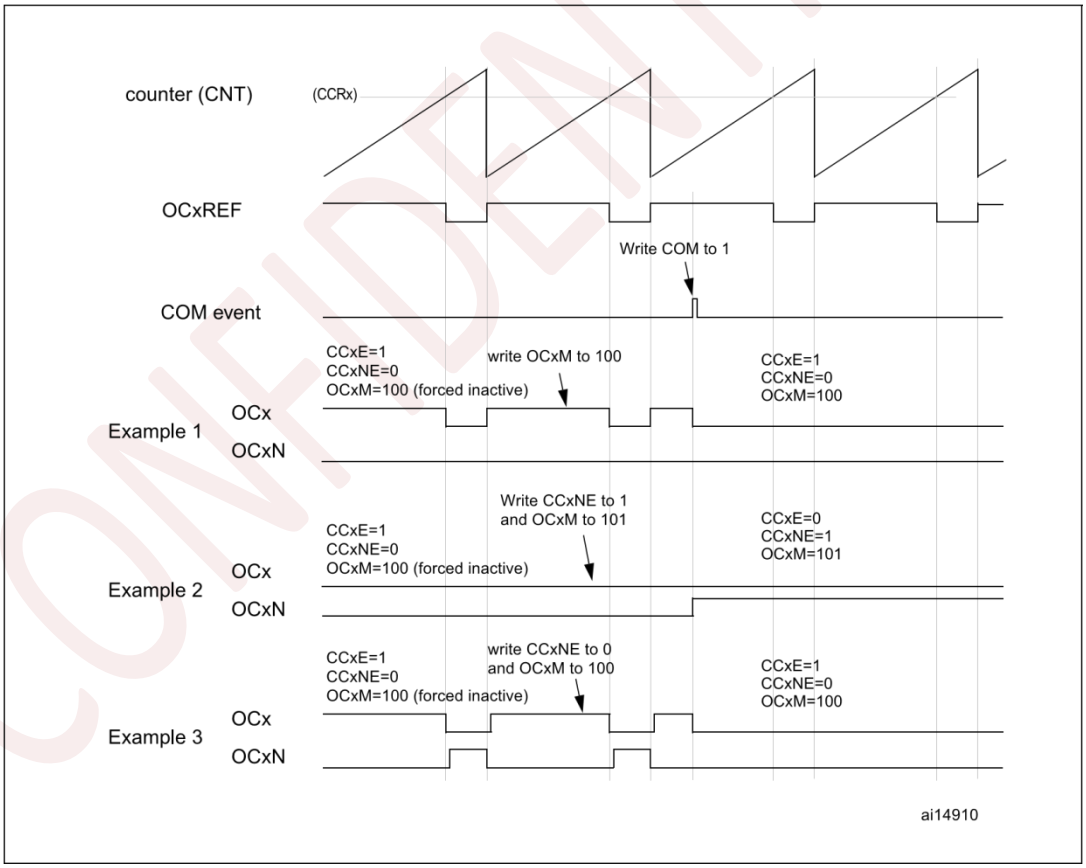
8.3.14 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. The user can thus program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the EGR register or by hardware (on TRGI rising edge).

A flag is set when the COM event occurs (COMIF bit in the SR register), which can generate an interrupt (if the COMIE bit is set in the DIER register) or a DMA request (if the COMDE bit is set in the DIER register).

Figure 40 describes the behavior of the OCx and OCxN outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 40. 6-step generation, COM example (OSSR=1)



8.3.15 One-pulse mode

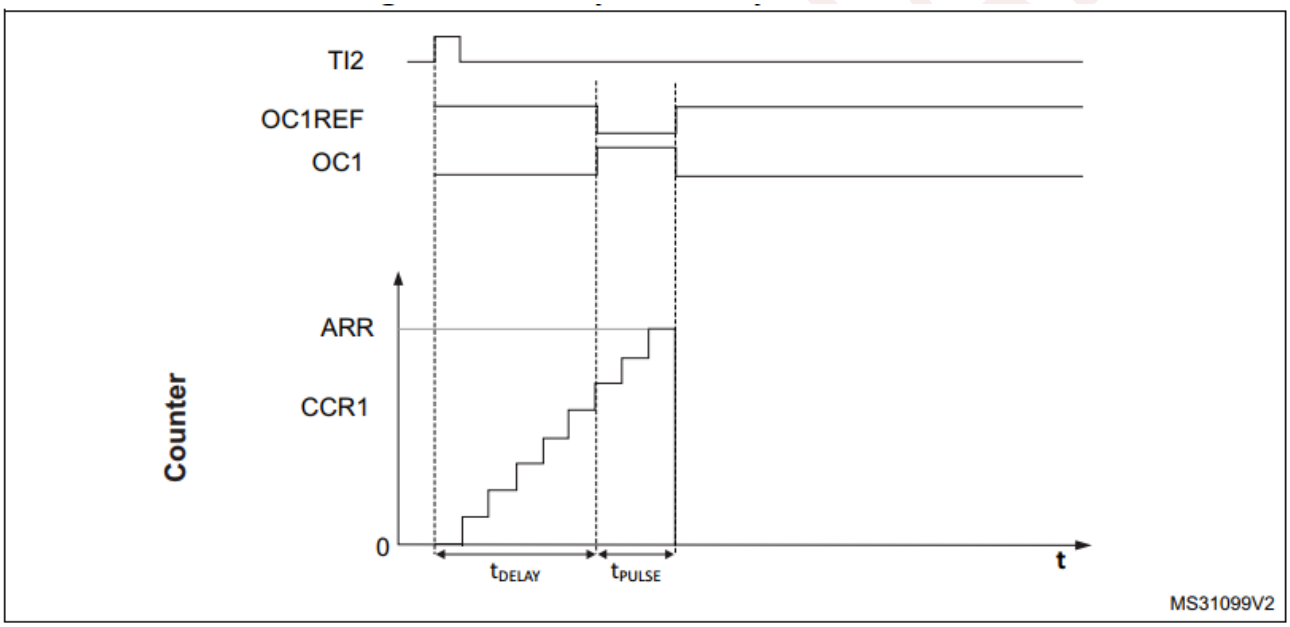
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

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 One-pulse mode by setting the OPM bit in the CR1 register. This makes the counter stop automatically at the next update event UEV.

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 upcounting: CNT < CCRx ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 41. Example of one pulse mode.



For example the user may want to generate a positive pulse on OC1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the TI2 input pin.

Let's use TI2FP2 as trigger 1:

- Map TI2FP2 to TI2 by writing CC2S='01' in the CCMR1 register.
- TI2FP2 must detect a rising edge, write CC2P='0' in the CCER register.
- Configure TI2FP2 as trigger for the slave mode controller (TRGI) by writing TS='110' in the SMCR register.
- TI2FP2 is used to start the counter by writing SMS to '110' in the SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the CCR1 register.
- The t_{PULSE} is defined by the difference between the auto-reload value and the compare value (ARR - CCR1).
- Let us say the user wants 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. To do this, enable PWM mode 2 by writing OC1M=111 in the CCMR1 register. The user can optionally enable the preload registers by writing OC1PE=’1’ in the CCMR1 register and ARPE in the CR1 register. In this case the compare value must be written in the CCR1 register, the auto-reload value in the ARR register, generate an update by setting the UG bit and wait for external trigger event on TI2. CC1P is written to ‘0’ in this example.

In our example, the DIR and CMS bits in the CR1 register should be low.

The user only wants one pulse (Single mode), so '1’ must be written in the OPM bit in the CR1 register to stop the counter at the next update event (when the counter rolls over from the auto-reload value back to 0). When OPM bit in the CR1 register is set to '0', so the Repetitive Mode is selected.

Particular case: OCx fast enable:

In One-pulse mode, the edge detection on TIx input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If the user wants to output a waveform with the minimum delay, the OCxFE bit in the CCMRx register must be set. Then OCxRef (and OCx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

8.3.16 Encoder interface mode

To select Encoder Interface mode write SMS=‘001’ in the SMCR register if the counter is counting on TI2 edges only, SMS=’010’ if it is counting on TI1 edges only and SMS=’011’ if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the CC1P and CC2P bits in the CCER register. When needed, the user can program the input filter as well.

The two inputs TI1 and TI2 are used to interface to an incremental encoder. 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) assuming that it is enabled (CEN bit in
CR1 register written to ‘1’). 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, the DIR bit in the CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (TI1 or TI2), whatever 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 ARR register (0 to ARR or ARR down to 0 depending on the direction). So user must configure ARR before starting. in the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the incremental encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor.

Table 1 summarizes the possible combinations, assuming TI1 and TI2 do not switch at the same time.

Table 1. Counting direction versus encoder signals

Active edge	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 be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to an external interrupt input and trigger a counter reset.

Figure 42 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S='01' (CCMR1 register, TI1FP1 mapped on TI1).
- CC2S='01' (CCMR2 register, TI1FP2 mapped on TI2).

- CC1P='0', and IC1F = '0000' (CCER register, TI1FP1 non-inverted, TI1FP1=TI1).
- CC2P='0', and IC2F = '0000' (CCER register, TI1FP2 non-inverted, TI1FP2= TI2).
- SMS='011' (SMCR register, both inputs are active on both rising and falling edges).
- CEN='1' (CR1 register, Counter enabled).

Figure 42. Example of counter operation in encoder interface mode.

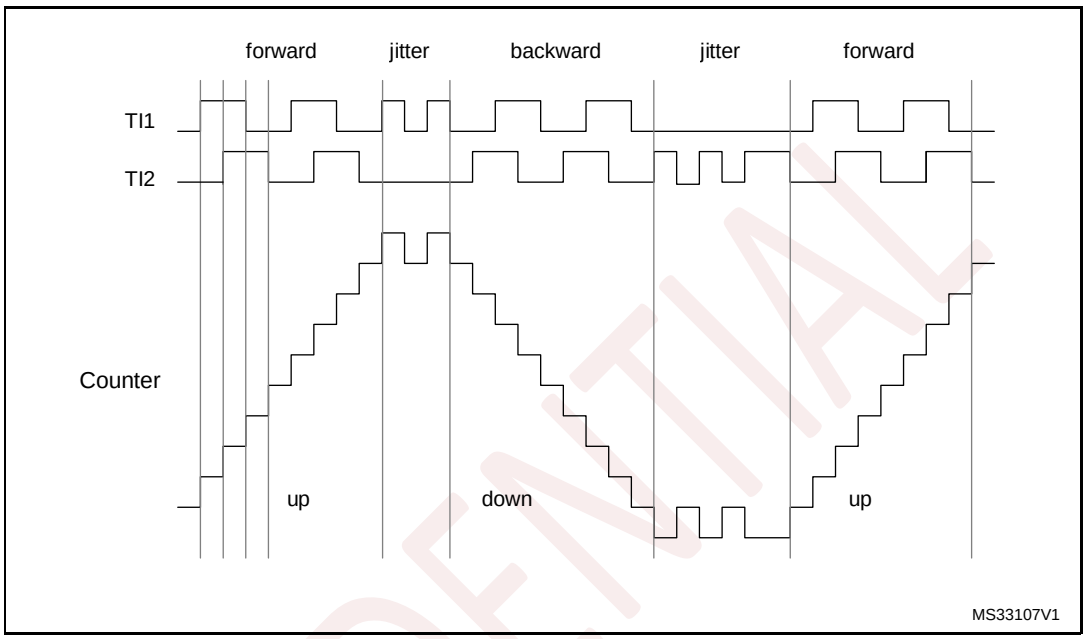
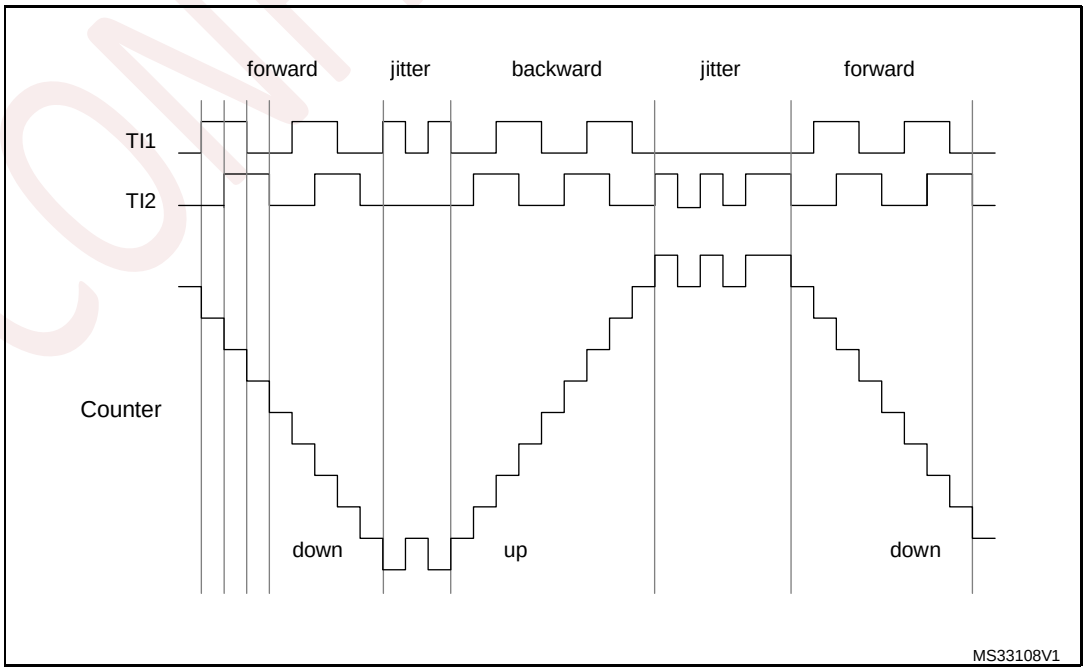


Figure 43 gives an example of counter behavior when TI1FP1 polarity is inverted (same configuration as above except CC1P='1').

Figure 43. Example of encoder interface mode with TI1FP1 polarity inverted.



The timer, when configured in Encoder Interface mode provides information on the sensor's current position.The user can obtain dynamic information (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which 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. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). when available, it is also possible to read its value through a DMA request generated by a real-time clock.

8.3.17 Timer input XOR function

The TI1S bit in the CR2 register, allows the input filter of channel 0 to be connected to the output of a XOR gate, combining the three input pins CH1, CH2 and CH3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

8.3.18 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer referred to as
"interfacing timer" in *Figure 44*. The "interfacing timer" captures the 3 timer input pins (CH1, CH2, and CH3) connected through a XOR to the TI1 input channel (selected by setting the TI1S bit in the CR2 register).

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Thus, each time one of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 0 is configured in capture mode, capture signal is TRC (see *Figure 27*). The captured value, which corresponds to the time elapsed between 2 changes on the inputs, gives information about motor speed.

The "interfacing timer" can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the TRGO output.

Example: the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the timers.

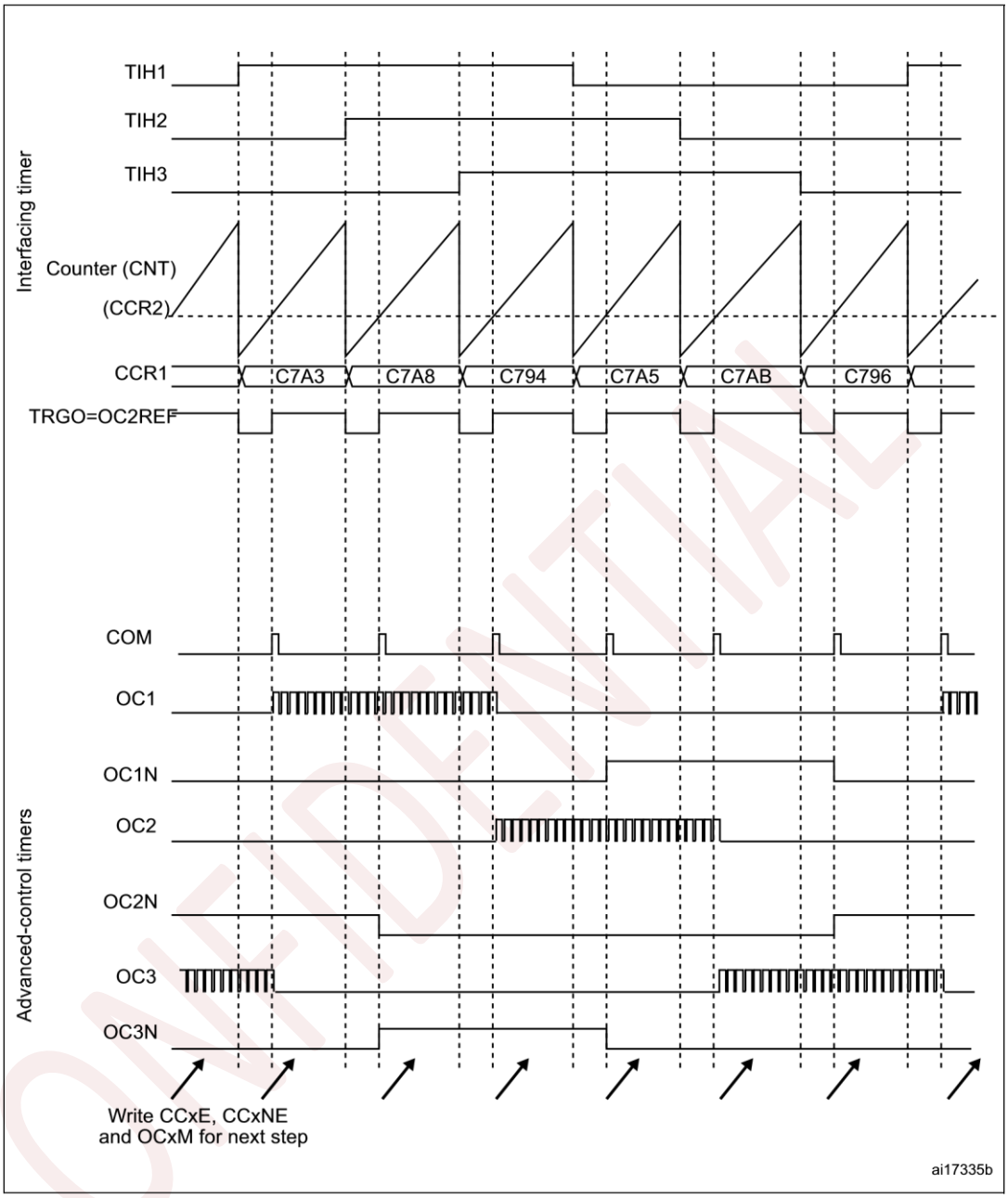
- Configure 3 timer inputs ORed to the TI1 input channel by writing the TI1S bit in the CR2 register to '1',

- Program the time base: write the ARR to the max value (the counter must be cleared by the TI1 change. Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors,
- Program channel 0 in capture mode (TRC selected): write the CC1S bits in the CCMR1 register to ‘11’. The user can also program the digital filter if needed,
- Program channel 1 in PWM 2 mode with the desired delay: write the OC2M bits to ‘111’ and the CC2S bits to ‘00’ in the CCMR1 register,
- Select OC2REF as trigger output on TRGO: write the MMS bits in the CR2 register to ‘101’,

In the advanced-control timer, the right ITR input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC=1 in the CR2 register) and the COM event is controlled by the trigger input (CCUS=1 in the CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of OC2REF).

Figure 44 describes this example.

Figure 44. Example of Hall sensor interface



8.3.19 External trigger synchronization

The timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode and Trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the CR1 register is low, an update event UEV is generated. Then all the preloaded registers (ARR, CCRx) are updated.

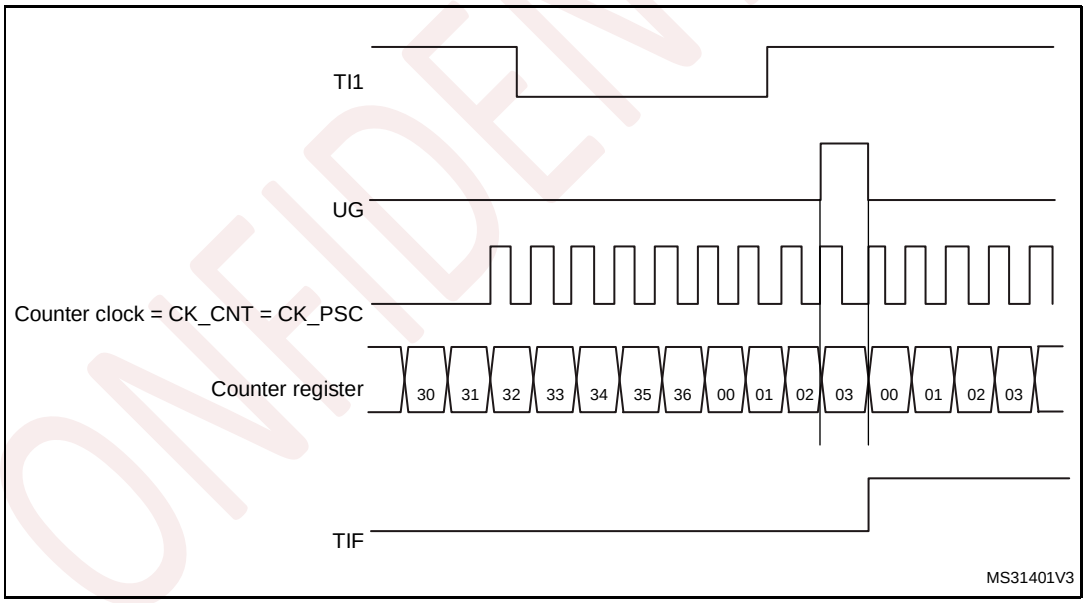
In the following example, the upcounter is cleared in response to a rising edge on TI1 input:

- Configure the channel 0 to detect rising edges on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC1S bits select the input capture source only, CC1S = 01 in the CCMR1 register. Write CC1P=0 in CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS=100 in SMCR register. Select TI1 as the input source by writing TS=101 in SMCR register.
- Start the counter by writing CEN=1 in the CR1 register.

The counter starts counting on the internal clock, then behaves normally until TI1 rising edge. When TI1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in DIER register).

The following figure shows this behavior when the auto-reload register ARR=0x36. The delay between the rising edge on TI1 and the actual reset of the counter is due to the resynchronization circuit on TI1 input.

Figure 45. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TI1 input is low:

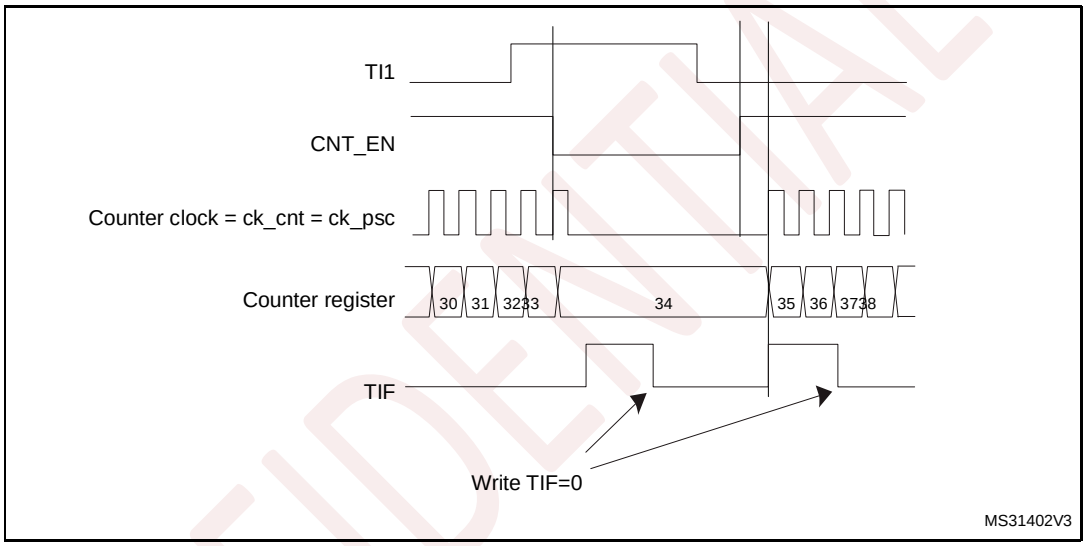
- Configure the channel 0 to detect low levels on TI1. Configure the input filter duration (in this example, we don't need any filter, so we keep IC1F=0000). The capture prescaler is not used for triggering, so the user does not need to configure it. The CC1S bits select the input capture source only, CC1S=01 in CCMR1 register. Write CC1P=1 in CCER register to validate the polarity (and detect low level only).

- Configure the timer in gated mode by writing SMS=101 in SMCR register. Select TI1 as the input source by writing TS=101 in SMCR register.
- Enable the counter by writing CEN=1 in the CR1 register (in gated mode, the counter doesn't start if CEN=0, whatever is 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 TIF flag in the SR register is set both when the counter starts or stops.

The delay between the rising edge on TI1 and the actual stop of the counter is due to the resynchronization circuit on TI1 input.

Figure 46. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

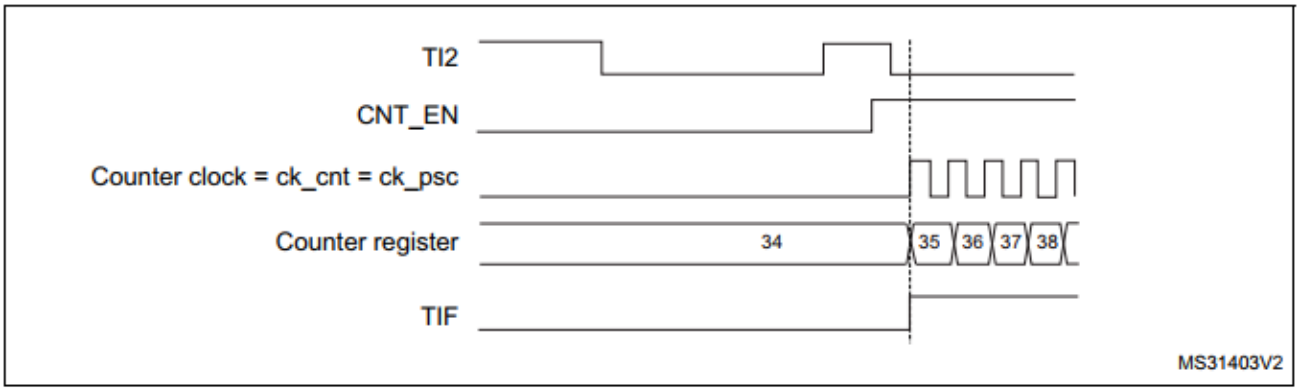
In the following example, the upcounter starts in response to a rising edge on TI2 input:

- Configure the channel 1 to detect rising edges on TI2. Configure the input filter duration (in this example, we don't need any filter, so we keep IC2F=0000). The capture prescaler is not used for triggering, so there's no need to configure it. The CC2S bits are configured to select the input capture source only, CC2S=01 in CCMR1 register. Write CC2P=1 in CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS=110 in SMCR register. Select TI2 as the input source by writing TS=110 in SMCR register.

When a rising edge occurs on TI2, the counter starts counting on the internal clock and the TIF 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 TI2 input.

Figure 47. Control circuit in trigger mode



Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to 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 (in reset mode, gated mode or trigger mode). It is recommended not to select ETR as TRGI through the TS bits of SMCR register.

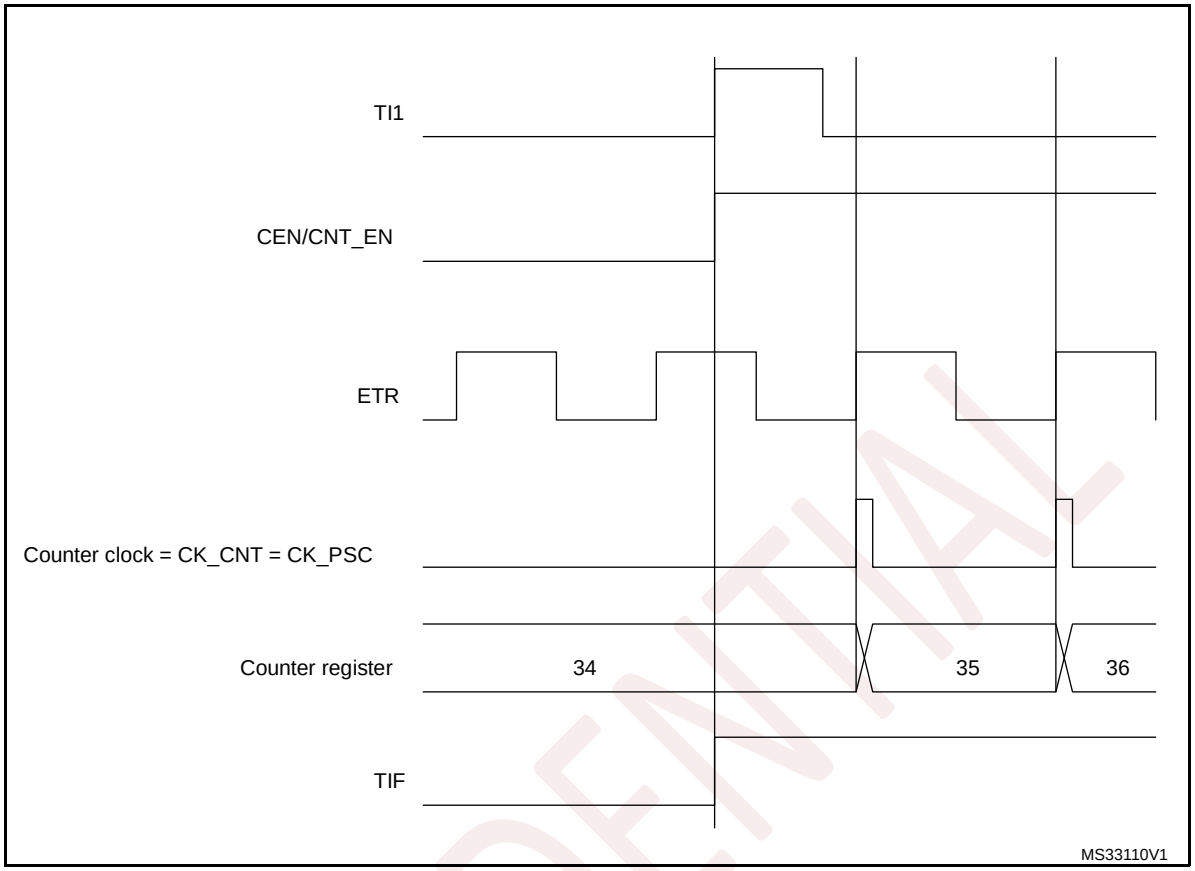
In the following example, the upcounter is incremented at each rising edge of the ETR signal as soon as a rising edge of TI1 occurs:

1. Configure the external trigger input circuit by programming the SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on ETR and ECE=1 to enable the external clock mode 2.
2. Configure the channel 0 as follows, to detect rising edges on TI:
   - IC1F=0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S=01 in CCMR1 register to select only the input capture source
   - CC1P=0 in CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS=110 in SMCR register. Select TI1 as the input source by writing TS=101 in SMCR register.

A rising edge on TI1 enables the counter and sets the TIF 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 48. Control circuit in external clock mode 2 + trigger mode



8.3.20 Timer synchronization

The TIM timers are linked together internally for timer synchronization or chaining.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

8.3.21 Debug mode

When the microcontroller enters debug mode, the counter either continues to work normally or stops, depending on APB_CLK STOP configuration bit in the system control module.

8.4 registers

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

8.4.1 control register 1 (CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bit-field indicates the division ratio between the timer clock (CK_INT) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (ETR, TIx),

00: $t_{DTS}=t_{CK\_INT}$

01: $t_{DTS}=2*t_{CK\_INT}$

10: $t_{DTS}=4*t_{CK\_INT}$

11: $t_{DTS}=8*t_{CK\_INT}$

Bit 7 **ARPE**: Auto-reload preload enable

0: ARR register is not buffered

1: ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS=00 in CCMRx 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 channels configured in output (CCxS=00 in CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare

compare interrupt flags of channels configured in output (CCxS=00 in CCMRx register) are are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN=1)

Bit 4 **DIR**: Direction

0: Counter used as upcounter

1: Counter used as downcounter

Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One pulse mode

0: Counter is not stopped at update event

1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

8.4.2 control register 2 (CR2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OIS3	OIS2N	OIS2	OIS1N	OIS1	OIS0N	OIS0	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **OIS3**: Output Idle state 4 (OC3 output)

refer to OIS1 bit

Bit 13 **OIS2N**: Output Idle state 3 (OC2N output)

refer to OIS1N bit

Bit 12 **OIS2**: Output Idle state 3 (OC2 output)

refer to OIS1 bit

Bit 11 **OIS1N**: Output Idle state 2 (OC1N output)

refer to OIS1N bit

Bit 10 **OIS1**: Output Idle state 2 (OC1 output)

refer to OIS1 bit

Bit 9 **OIS0N**: Output Idle state 1 (OC0N output)

0: OC0N=0 after a dead-time when MOE=0

1: OC0N=1 after a dead-time when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in BDTR register).

Bit 8 **OIS0**: Output Idle state 1 (OC0 output)

0: OC0=0 (after a dead-time if OC1N is implemented) when MOE=0

1: OC0=1 (after a dead-time if OC1N is implemented) when MOE=0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in BDTR register).

Bit 7 **TI1S**: TI1 selection

0: The CH1 pin is connected to TI1 input

1: The CH1, CH2 and CH3 pins are connected to the TI1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000: **Reset** - the UG bit from the EGR register is used as trigger output (TRGO). If the reset 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** - the Counter Enable signal CNT_EN is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic OR between CEN control bit and 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 MSM bit description in SMCR register).

010: **Update** - The update event is selected as trigger output (TRGO). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (TRGO).

100: **Compare** - OC0REF signal is used as trigger output (TRGO)

101: **Compare** - OC1REF signal is used as trigger output (TRGO)

110: **Compare** - OC2REF signal is used as trigger output (TRGO)

111: **Compare** - OC3REF signal is used as trigger output (TRGO)

Note: The clock of the slave timer and ADC must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC=1), they are updated by setting the COMG bit or when an rising edge occurs on TRGI

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on TRGI, depending on the CCUS bit).

8.4.3 slave mode control register (SMCR)

Address offset: 0x08

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Res.	rw	rw	rw

Reset value: 0x0000

Bit 15 **ETP**: External trigger polarity

This bit selects whether ETR or ETR is used for trigger operations 0: ETR is non-inverted, active at high level or rising edge. 1: ETR is inverted, active at low level or falling edge.

Bit 14 **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: **1:** Setting the ECE bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMS=111 and TS=111).*

***2:** It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TS bits must not be 111).*

***3:** If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.*

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal ETRP frequency must be at most 1/4 of CLK frequency. A prescaler can be enabled to reduce ETRP frequency. It is useful when inputting fast external clocks.

00: Prescaler OFF

01: ETRP frequency divided by 2

10: ETRP frequency divided by 4

11: ETRP frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bit-field then defines the frequency used to sample ETRP signal and the length of the digital filter applied to ETRP. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output: 0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bit 7 **MSM:** Master/slave mode

0: No action

1: The effect of an event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bit-field selects the trigger input to be used to synchronize the counter.

000: Internal Trigger 0 (ITR0)

001: Internal Trigger 1 (ITR1)

010: Internal Trigger 2 (ITR2)

011: Internal Trigger 3 (ITR3)

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 (e.g. when SMS=000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **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 (see Input Control register and Control register description.

000: Slave mode disabled - if CEN = '1' then the prescaler is clocked directly by the internal clock.

001: Encoder mode 1 - Counter counts up/down on TI2FP1 edge depending on TI1FP2 level.

010: Encoder mode 2 - Counter counts up/down on TI1FP2 edge depending on TI2FP1

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 input (TRGI) reinitializes the counter and generates an update of the registers.

101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

110: Trigger Mode - The counter starts at a rising edge of the trigger TRGI (but it is not reset). Only the start of the counter is controlled.

111: External Clock Mode 1 - Rising edges of the selected trigger (TRGI) clock the counter.

Note: The gated mode must not be used if TI1F_ED is selected as the trigger input (TS='100'). Indeed, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

The clock of the slave timer must be enabled prior to receiving events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Table 2. Internal trigger connection

Slave TIM	ITR0 (TS = 000)	ITR1 (TS = 001)	ITR2 (TS = 010)	ITR3 (TS = 011)
GPTIMER0	GPTIMER1_TRGO	GPTIMER2_TRGO	GPTIMER3_TRGO	GPTIMER4_TRGO
GPTIMER1	GPTIMER2_TRGO	GPTIMER3_TRGO	GPTIMER4_TRGO	GPTIMER0_TRGO
GPTIMER2	GPTIMER3_TRGO	GPTIMER4_TRGO	GPTIMER0_TRGO	GPTIMER1_TRGO
GPTIMER3	GPTIMER4_TRGO	GPTIMER0_TRGO	GPTIMER1_TRGO	GPTIMER2_TRGO
GPTIMER4	GPTIMER0_TRGO	GPTIMER1_TRGO	GPTIMER2_TRGO	GPTIMER3_TRGO

8.4.4 DMA/interrupt enable register (DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC3DE	CC2DE	CC1DE	CC0DE	UDE	BIE	TIE	COMIE	CC3IE	CC2IE	CC1IE	CC0IE	UIE
	rw	Rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bit 12 **CC3DE**: Capture/Compare 3 DMA request enable

0: CC3 DMA request disabled

1: CC3 DMA request enabled

Bit 11 **CC2DE**: Capture/Compare 2 DMA request enable

0: CC2 DMA request disabled

1: CC2 DMA request enabled

Bit 10 **CC1DE**: Capture/Compare 1 DMA request enable

0: CC1 DMA request disabled

1: CC1 DMA request enabled

Bit 9 **CC0DE**: Capture/Compare 0 DMA request enable

0: CC0 DMA request disabled

1: CC0 DMA request enabled

Bit 8 **UDE**: Update DMA request enable

0: Update DMA request disabled

1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable

0: Break interrupt disabled

1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled

1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable

0: COM interrupt disabled

1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable

0: CC3 interrupt disabled

1: CC3 interrupt enabled

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable

0: CC2 interrupt disabled

1: CC2 interrupt enabled

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC0 interrupt disabled

1: CC0 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

8.4.5 status register (SR)

Address offset: 0x10

Reset value: 0x0000

15 14 13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CC3OF	CC2OF	CC1OF	CC0OF	Res.	BIF	TIF	COMIF	CC3IF	CC2IF	CC1IF	CC0IF	UIF
	rc_w0	rc_w0	rc_w0	rc_w0	Res.	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **CC3OF**: Capture/Compare 3 overcapture flag

refer to CC0OF description

Bit 11 **CC2OF**: Capture/Compare 2 overcapture flag

refer to CC0OF description

Bit 10 **CC1OF**: Capture/Compare 1 overcapture flag

refer to CC0OF description

Bit 9 **CC0OF**: Capture/Compare 0 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to '0'.

0: No overcapture has been detected.

1: The counter value has been captured in CCR0 register while CC0IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on trigger event (active edge detected on TRGI input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when Capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC3IF**: Capture/Compare 3 interrupt flag

refer to CC0IF description

Bit 3 **CC2IF**: Capture/Compare 2 interrupt flag

refer to CC0IF description

Bit 2 **CC1IF**: Capture/Compare 1 interrupt flag

refer to CC0IF description

Bit 1 **CC0IF**: Capture/Compare 0 interrupt flag

If channel CC0 is configured as output:

This flag is set by hardware when the counter matches the compare value, with some exception in center-aligned mode (refer to the CMS bits in the CR0 register description). It is cleared by software.

0: No match.

1: The content of the counter CNT matches the content of the CCR0 register. When the contents of CCR0 are greater than the contents of ARR, the CC0IF bit goes high on the counter overflow (in upcounting and up/down-counting modes) or underflow (in downcounting mode)

If channel CC0 is configured as input:

This bit is set by hardware on a capture. It is cleared by software or by reading the CCR0 register.

0: No input capture occurred

1: The counter value has been captured in CCR0 register (An edge has been detected on IC0 which matches the selected polarity)

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS=0 in the CR1 register.

– When CNT is reinitialized by software using the UG bit in EGR register, if URS=0 and UDIS=0 in the CR0 register.

– When CNT is reinitialized by a trigger event (refer to *Section: slave mode control register (SMCR)*), if URS=0 and UDIS=0 in the CR1 register.

8.4.6 event generation register (EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								BG	TG	COMG	CC3G	CC2G	CC1G	CC0G	UG
								w	w	w	w	w	w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: The TIF flag is set in SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware 0: No action

1: When CCPC bit is set, it allows to update CCxE, CCxNE and OCxM bits *Note: This bit acts only on channels having a complementary output.*

Bit 3 **CC3G**: Capture/Compare 3 generation refer to CC1G description

Bit 2 **CC2G**: Capture/Compare 2 generation refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation refer to CC1G description

Bit 0 **CC0G**: Capture/Compare 0 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action

1: A capture/compare event is generated on channel 0:

If channel CC0 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC0 is configured as input:

The current value of the counter is captured in CCR0 register. The CC0IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC0OF flag is set if the CC0IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR=0 (upcounting), else it takes the auto-reload value (ARR) if DIR=1 (downcounting).

8.4.7 capture/compare mode register 1 (CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. So the user must take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC1CE	OC1M[2:0			OC1PE	OC1FE	CC1S[1:0]		OC0CE	OC0M[2:0			OC0PE	OC0FE	CC0S[1:0]	
IC1F[3:0]				IC1PSC[1:0]				IC0F[3:0]				IC0PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bit 15 **OC1CE:** Output compare 2 clear enable

Bits 14:12 **OC1M[2:0]**: Output compare 2 mode

Bit 11 **OC1PE**: Output compare 2 preload enable

Bit 10 **OC1FE**: Output compare 2 fast enable

Bits 9:8 **CC1S[1:0]**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1channel is configured as input, IC1 is mapped on TI1

10: CC1 channel is configured as input, IC1 is mapped on TI0

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through the TS bit (SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in CCER).

Bit 7 **OC0CE:** Output compare 1 clear enable

OC0CE: Output compare 1 Clear Enable 0:

OC0Ref is not affected by the ETRF Input

1: OC0Ref is cleared as soon as a High level is detected on ETRF input

Bits 6:4 **OC0M**: Output compare 1 mode

These bits define the behavior of the output reference signal OC0REF from which OC0 and OC0N are derived. OC0REF is active high whereas OC0 and OC0N active level depends on CC0P and CC0NP bits.

000: Frozen - The comparison between the output compare register CCR0 and the counter CNT has no effect on the outputs.(this mode is used to generate a timing base).

001: Set channel 0 to active level on match. OC0REF signal is forced high when the counter CNT matches the capture/compare register 1 (CCR0).

010: Set channel 0 to inactive level on match. OC1REF signal is forced low when the counter CNT matches the capture/compare register 1 (CCR0).

011: Toggle - OC0REF toggles when CNT=CCR0.

100: Force inactive level - OC0REF is forced low.

101: Force active level - OC0REF is forced high.

110: PWM mode 1 - In upcounting, channel 0 is active as long as CNT<CCR0

else inactive. In downcounting, channel 0 is inactive (OC0REF='0') as long as CNT>CCR0 else active (OC0REF='1').

111: PWM mode 2 - In upcounting, channel 0 is inactive as long as CNT<CCR0 else active. In downcounting, channel 0 is active as long as CNT>CCR0 else inactive.

Note: ***1:*** *These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in BDTR register) and CC0S='00' (the channel is configured in output).*

2: *In PWM mode 1 or 2, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.*

Bit 3 **OC0PE**: Output compare 1 preload enable

0: Preload register on CCR0 disabled. CCR0 can be written at anytime, the new value is taken in account immediately.

1: Preload register on CCR0 enabled. Read/Write operations access the preload register. CCR0 preload value is loaded in the active register at each update event.

Note: ***1:*** *These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in BDTR register) and CC0S='00' (the channel is configured in output).*

2: *The PWM mode can be used without validating the preload register only in one pulse mode (OPM bit set in CR0 register). Else the behavior is not guaranteed.*

Bit 2 **OC0FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC0 behaves normally depending on counter and CCR0 values even when the trigger is ON. The minimum delay to activate CC0 output 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 CC0 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC0 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC0S**: Capture/Compare 1 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC0 channel is configured as output

01: CC0 channel is configured as input, IC0 is mapped on TI1

10: CC0 channel is configured as input, IC0 is mapped on TI2

11: CC0 channel is configured as input, IC0 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC0S bits are writable only when the channel is OFF (CC0E = '0' in CCER).

Input capture mode

Bits 15:12 **IC1F**: Input capture 2 filter

Bits 11:10 **IC1PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC1S**: Capture/Compare 2 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, IC1 is mapped on TI2

10: CC1 channel is configured as input, IC1 is mapped on TI1

11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = '0' in CCER).

Bits 7:4 **IC0F[3:0]**: Input capture 1 filter

This bit-field defines the frequency used to sample TI1 input and the length of the digital filter applied to TI1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING}=f_{CK\_INT}$, N=2

0010: $f_{SAMPLING}=f_{CK\_INT}$, N=4

0011: $f_{SAMPLING}=f_{CK\_INT}$, N=8

0100: $f_{SAMPLING}=f_{DTS}/2$, N=6

0101: $f_{SAMPLING}=f_{DTS}/2$, N=8

0110: $f_{SAMPLING}=f_{DTS}/4$, N=6

0111: $f_{SAMPLING}=f_{DTS}/4$, N=8

1000: $f_{SAMPLING}=f_{DTS}/8$, N=6

1001: $f_{SAMPLING}=f_{DTS}/8$, N=8

1010: $f_{SAMPLING}=f_{DTS}/16$, N=5

1011: $f_{SAMPLING}=f_{DTS}/16$, N=6

1100: $f_{SAMPLING}=f_{DTS}/16$, N=8

1101: $f_{SAMPLING}=f_{DTS}/32$, N=5

1110: $f_{SAMPLING}=f_{DTS}/32$, N=6

1111: $f_{SAMPLING}=f_{DTS}/32$, N=8

Bits 3:2 **IC0PSC**: Input capture 1 prescaler

This bit-field defines the ratio of the prescaler acting on CC0 input (IC0).

The prescaler is reset as soon as CC0E='0' (CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC0S**: Capture/Compare 1 Selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC0 channel is configured as output

01: CC0 channel is configured as input, IC0 is mapped on TI1

10: CC0 channel is configured as input, IC0 is mapped on TI2

11: CC0 channel is configured as input, IC0 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC0S bits are writable only when the channel is OFF (CC0E = '0' in CCER).

8.4.8 capture/compare mode register 2 (CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]		OC3 CE.	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]	
IC3F[3:0]				IC3PSC[1:0]				IC2F[3:0]				IC2PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bit 15 OC3CE: Output compare 4 clear enable

Bits 14:12 **OC3M**: Output compare 4 mode

Bit 11 **OC3PE**: Output compare 4 preload enable

Bit 10 **OC3FE**: Output compare 4 fast enable

Bits 9:8 **CC3S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on TI2

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in CCER).

Bit 7 **OC2CE:** Output compare 3 clear enable

Bits 6:4 **OC2M**: Output compare 3 mode

Bit 3 **OC2PE**: Output compare 3 preload enable

Bit 2 **OC2FE**: Output compare 3 fast enable

Bits 1:0 **CC2S**: Capture/Compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC3 is mapped on TI3

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in CCER).

Input capture mode

Bits 15:12 **IC3F**: Input capture 4 filter

Bits 11:10 **IC3PSC**: Input capture 4 prescaler

Bits 9:8 **CC3S**: Capture/Compare 4 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, IC3 is mapped on TI3

10: CC3 channel is configured as input, IC3 is mapped on T2

11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = '0' in CCER).

Bits 7:4 **IC2F**: Input capture 3 filter

Bits 3:2 **IC2PSC**: Input capture 3 prescaler

Bits 1:0 **CC2S**: Capture/compare 3 selection

This bit-field defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, IC2 is mapped on TI2

10: CC2 channel is configured as input, IC2 is mapped on TI3

11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode is working only if an internal trigger input is selected through TS bit (SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = '0' in CCER).

8.4.9 capture/compare enable register (CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC3NP	Res.	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E	CC0NP	CC0NE	CC0P	CC0E
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15 **CC3NP**: Capture/Compare 4 complementary output polarity

refer to CC1NP description

Bit 14 **CC3NE**: Capture/Compare 3 complementary output enable

refer to CC0NE description

Bit 13 **CC3P**: Capture/Compare 4 output polarity

refer to CC0P description

Bit 12 **CC3E**: Capture/Compare 4 output enable

refer to CC0E description

Bit 11 **CC2NP**: Capture/Compare 3 complementary output polarity

refer to CC0NP description

Bit 10 **CC2NE**: Capture/Compare 3 complementary output enable

refer to CC0NE description

Bit 9 **CC2P**: Capture/Compare 3 output polarity

refer to CC0P description

Bit 8 **CC2E**: Capture/Compare 3 output enable

refer to CC0E description

Bit 7 **CC1NP**: Capture/Compare 2 complementary output polarity

refer to CC0NP description

Bit 6 **CC1NE**: Capture/Compare 2 complementary output enable

refer to CC0NE description

Bit 5 **CC1P**: Capture/Compare 2 output polarity

refer to CC0P description

Bit 4 **CC1E**: Capture/Compare 2 output enable

refer to CC0E description

Bit 3 **CC0NP**: Capture/Compare 1 complementary output polarity

0: OC0N active high.

1: OC0N active low.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in BDTR register) and CC0S="00" (the channel is configured in output).

Bit 2 **CC0NE**: Capture/Compare 1 complementary output enable

0: Off - OC0N is not active. OC0N level is then function of MOE, OSSI, OSSR, OIS0, OIS0N and CC0E bits.

1: On - OC0N signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS0, OIS0N and CC0E bits.

Bit 1 **CC0P**: Capture/Compare 1 output polarity

CC0 channel configured as output:

0: OC0 active high

1: OC0 active low

CC0 channel configured as input:

This bit selects whether IC0 or IC0 is used for trigger or capture operations.

0: non-inverted: capture is done on a rising edge of IC0. When used as external trigger, IC0 is non-inverted.

1: inverted: capture is done on a falling edge of IC0. When used as external trigger, IC0 is inverted.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in BDTR register).

Bit 0 **CC0E**: Capture/Compare 1 output enable

CC0channel configured as output:

0: Off - OC0 is not active. OC0 level is then function of MOE, OSSI, OSSR, OIS0, OIS0N and CC0NE bits.

1: On - OC0 signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS0, OIS0N and CC0NE bits.

CC0 channel configured as input:

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (CCR0) or not.

0: Capture disabled.

1: Capture enabled.

Table 3. Output control bits for complementary OCx and OCxN channels with break feature

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	OCx output state	OCxN output state
1	X	0	0	0	Output Disabled (not driven by the timer), OCx=0, OCx_EN=0	Output Disabled (not driven by the timer), OCxN=0, OCxN_EN=0
		0	0	1	Output Disabled (not driven by the timer), OCx=0, OCx_EN=0	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		0	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Output Disabled (not driven by the timer) OCxN=0, OCxN_EN=0
		0	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
		1	0	0	Output Disabled (not driven by the timer) OCx=CCxP, OCx_EN=0	Output Disabled (not driven by the timer) OCxN=CCxNP, OCxN_EN=0
		1	0	1	Off-State (output enabled with inactive state) OCx=CCxP, OCx_EN=1	OCxREF + Polarity OCxN=OCxREF xor CCxNP, OCxN_EN=1
		1	1	0	OCxREF + Polarity OCx=OCxREF xor CCxP, OCx_EN=1	Off-State (output enabled with inactive state) OCxN=CCxNP, OCxN_EN=1
		1	1	1	OCREF + Polarity + dead-time OCx_EN=1	Complementary to OCREF (not OCREF) + Polarity + dead-time OCxN_EN=1
0	0	X	0	0	Output Disabled (not driven by the timer) Asynchronously: OCx=CCxP, OCx_EN=0, OCxN=CCxNP, OCxN_EN=0	

					Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state.
	0		0	1	
	0		1	0	
	0		1	1	
	1		0	0	
	1		0	1	Off-State (output enabled with inactive state) Asynchronously: OCx=CCxP, OCx_EN=1, OCxN=CCxNP, OCxN_EN=1 Then if the clock is present: OCx=OISx and OCxN=OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and OCxN both in active state
	1		1	0	
	1		1	1	

1. When both outputs of a channel are not used (CCxE = CCxNE = 0), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIOand AFIO registers.

8.4.10 counter (CNT)

Address offset: 0x24

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNT[31:0]**: Counter value

8.4.11 prescaler (PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (CK_CNT) is equal to f_{CK_PSC} / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of EGR register or through trigger controller when configured in “reset mode”).

8.4.12 auto-reload register (ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **ARR[31:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

The counter is blocked while the auto-reload value is null.

8.4.13 repetition counter register (RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter value

These bits allow the user to set-up the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable, as well as the update interrupt generation rate, if this interrupt is enable.

Each time the REP_CNT related downcounter reaches zero, an update event is generated and it restarts counting from REP value. As REP_CNT is reloaded with REP value only at the repetition update event U_RC, any write to the RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode
– the number of half PWM period in center-aligned mode.

8.4.14 capture/compare register 1 (CCR0)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR0[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 31:0 **CCR0[31:0]**: Capture/Compare 1 value

If channel CC0 is configured as output:

CCR0 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the CCMR0 register (bit OC0PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter CNT and signaled on OC0 output.

If channel CC0 is configured as input:

CCR0 is the counter value transferred by the last input capture 1 event (IC0). The CCR0 register is read-only and cannot be programmed.

8.4.15 capture/compare register 2 (CCR1)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 31:0 **CCR1[31:0]**: Capture/Compare 2 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter CNT and signalled on OC1 output.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 2 event (IC1). The CCR1 register is read-only and cannot be programmed.

8.4.16 capture/compare register 3 (CCR2)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[31:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 31:0 **CCR2[31:0]**: Capture/Compare value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter CNT and signalled on OC2 output.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 3 event (IC2). The CCR2

register is read-only and cannot be programmed.

8.4.17 capture/compare register 4 (CCR3)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[31:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

Bits 31:0 **CCR3[31:0]**: Capture/Compare value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter CNT and signalled on OC3 output.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 4 event (IC3). The CCR2 register is read-only and cannot be programmed.

8.4.18 break and dead-time register (BDTR)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	Rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits AOE, BKP, BKE, OSSI, OSSR and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the BDTR register.

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the break input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: OC and OCN outputs are disabled or forced to idle state.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE

in CCER register).

See OC/OCN enable description for more details (*Section: capture/compare enable register (CCER)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input BRK is active low

1: Break input BRK is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (BRK and CSS clock failure event) disabled

1; Break inputs (BRK and CSS clock failure event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE=1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section : capture/compare enable register (CCER)*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE=1 or CCxNE=1. Then, OC/OCN enable output signal=1

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE=0 on channels configured as outputs.

See OC/OCN enable description for more details (*Section : capture/compare enable register (CCER)*).

0: When inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0).

1: When inactive, OC/OCN outputs are forced first with their idle level as soon as CCxE=1 or CCxNE=1. OC/OCN enable output signal=1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in BDTR register, OISx and OISxN bits in CR2 register and BKE/BKP/AOE bits in BDTR register can no longer be written. 10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bit-field defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5]=0xx => DT=DTG[7:0]x t_{dtg} with $t_{dtg}=t_{DTS}$.

DTG[7:5]=10x => DT=(64+DTG[5:0])xt_{dtg} with $T_{dtg}=2xt_{DTS}$.

DTG[7:5]=110 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=8xt_{DTS}$.

DTG[7:5]=111 => DT=(32+DTG[4:0])xt_{dtg} with $T_{dtg}=16xt_{DTS}$.

Example if T_{DTS}=125ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 us to 31750 ns by 250 ns steps,

32 us to 63us by 1 us steps,

64 us to 126 us by 2 us steps

Note: This bit-field can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in BDTR register).

8.4.19 register map

registers are mapped as 16-bit addressable registers as described in the table below:

Table 4. register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CR1	Reserved																						CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x04	CR2	Reserved																	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Reserved	CCPC
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x08	SMCR	Reserved																ETP	ECE	ETP S [1:0]		ETF[3:0]				MSM	TS[2:0]			Reserved	SMS[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0C	DIER	Reserved																	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SR	Reserved																			CC4OF	CC3OF	CC2OF	CC1OF	Reserved	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value																				0	0	0	0		0	0	0	0	0	0	0	0
0x14	EGR	Reserved																								BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																									0	0	0	0	0	0	0	0
0x18	CCMR1 Output compare mode	Reserved																OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CCMR1 Input capture mode	Reserved																IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	CCMR2 Output compare mode	Reserved																OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	CCMR2 Input capture mode	Reserved																IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	CCER	Reserved																CC4NP	Reserved	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CNT	Reserved																CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	PSC	Reserved																PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4. register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	**ARR**	Reserved																ARR[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	**RCR**	Reserved																								REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	**CCR1**	Reserved																CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	**CCR2**	Reserved																CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	**CCR3**	Reserved																CCR3[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	**CCR4**	Reserved																CCR4[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	**BDTR**	Reserved																MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

9 Watchdogs

9.1 Overview

AG32 device provide a independent watchdog, which connects to the Advanced Peripheral Bus(APB)。

The independent watchdog is based on a 12-bit downcounter and 8-bit prescaler. It is clocked from an independent 32 kHz internal RC and as it operates independently from the main clock, it can operate in Stop and Standby modes. It can be used either as a watchdog to reset the device when a problem occurs, or as a free-running timer for application timeout management. It is hardware- or software-configurable through the option bytes.

The Watchdog module is an AMBA slave module and connects to the Advanced Peripheral Bus (APB). The Watchdog module consists of a 32-bit down counter with a programmable timeout interval that has the capability to generate an interrupt and a reset signal on timing out. It is intended to be used to apply a reset to a system in the event of a software failure.

9.2 Independent watchdog (IWDG)

9.2.1 IWDG main features

(1) Free-running down-counter

(2) clocked from an LSI oscillator when Stop and normal modes and from LSE when Standby mode)

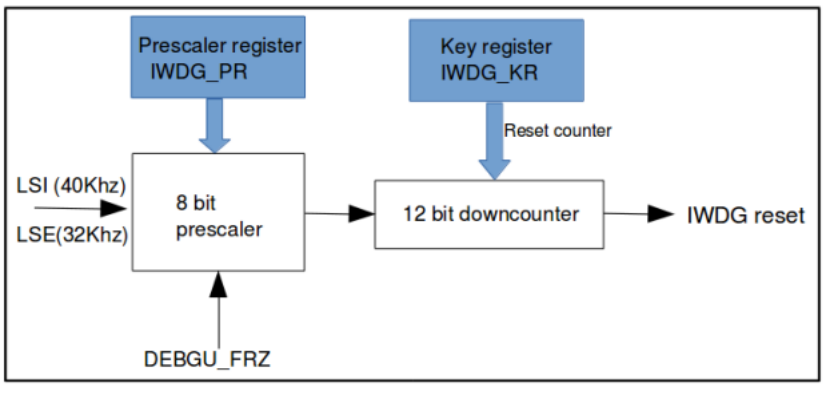
(3) Reset (if watchdog activated) when the down-counter value of 0x000 is reached

9.2.2 IWDG functional description

Figure below shows the functional blocks of the independent watchdog module.

When the independent watchdog is started, the counter starts counting down from the reset value of 0xFFF. When it reaches the end of count value (0x000) a reset signal is generated (IWDG reset).

Whenever the key value 1010 is written in the IWDG_KR register, the down-counter is initialized and the watchdog reset is prevented.



9.2.3 Watchdog clock

If the Independent watchdog (IWDG) is started by either hardware option or software access,

(1) Under run or stop mode

Select LSE or LSI clock source by setting the IWDG_STOP_CLKSEL bit in the Backup domain control register(RCC_BDCR).

(2) Under Standby mode

HW will select LSE as clock source for IWDG.

9.2.4 Debug mode

When the mcu enter debug mode, the IWDG counter either continues to work normally or stop, depending on DBG_IWDG_STOP configuration bit in DBG module.

IWDG timeout period (in ms) at 40 kHz (LSI)

Pre-scaler divider	**PR[2:0] bits**	**timeout (ms)**
/2	0	204.8
/4	1	409.6
/8	2	819.2
/16	3	1638.4
/32	4	3276.8
/64	5	6553.6
/128	6	13107.2
/256	7	26214.4

9.2.5 IWDG registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWDG_KR[3:0]				reserved			IWDG_EN	reserved	IWDG_STOP_CKSEL	IWDG_STDBY_FRZ	IWDG_STOP_FRZ	reserved	IWDG_PR[2:0]		
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bits 15:12 IWDG_KR<3:0>: Key value (write only, read 1010h)

These bits must be written by software at regular intervals with the key value 1010h, otherwise the watchdog generates a reset when the counter reaches 0.

Bits 8 IWDG_EN: IWDG enable control

0: IWDG disable

1: IWDG enable

Bits 6 IWDG_STOP_CKSEL: when STOP mode, select IWDG clock source.

0: select LSI

1: select LSE

note: when STANDBY mode, force to select LSI.

Bits 5 IWDG_STDBY_FRZ: when STANDBY mode, the IWDG counter either continues to work normally or stop.

0: normal work

1: stop

Bits 4 IWDG_STOP_FRZ: when STOP mode, the IWDG counter either continues to work normally or stop.

0: normal work

1: stop

Bits 2:0 IWDG_PR[2:0]: Pre-scaler divider

These bits are written by software to select the prescaler divider feeding the counter clock.

000: divider /2

001: divider /4

010: divider /8

011: divider /16

100: divider /32

101: divider /64

110: divider /128

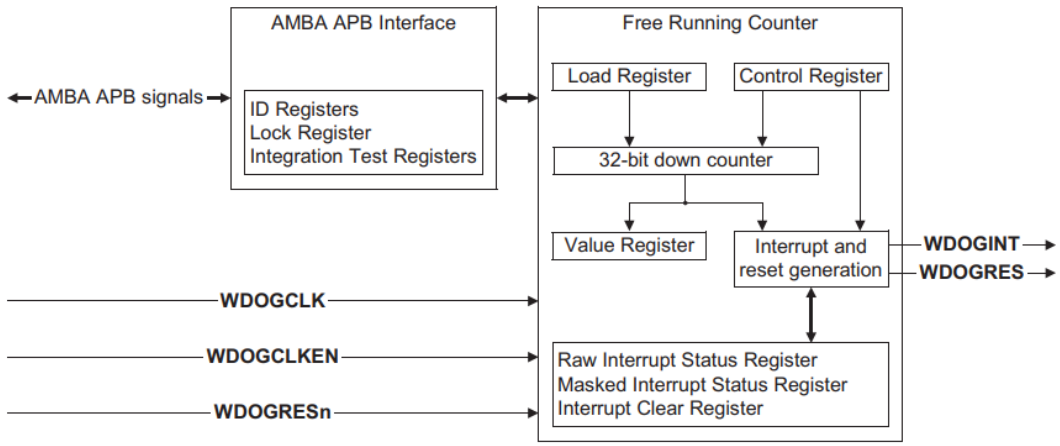
111: divider /256

9.3 Functional overview

9.3.1 Features

The features of the Watchdog module are:
• 32-bit down counter with a programmable timeout interval.
• Separate Watchdog clock with clock enable for flexible control of the timeout interval.
• Interrupt output generation on timeout.
• Reset signal generation on timeout if the interrupt from the previous timeout remains unserviced by software.
• Lock register to protect registers from being altered by runaway software.
• Identification registers that uniquely identify the Watchdog module. These can be used by software to automatically configure itself

Figure below shows a simplified block diagram of the Watchdog module.



Programmable parameters
The following Watchdog module parameters are programmable:
• interrupt generation enable/disable
• interrupt masking
• reset signal generation enable and/disable
• interrupt interval.

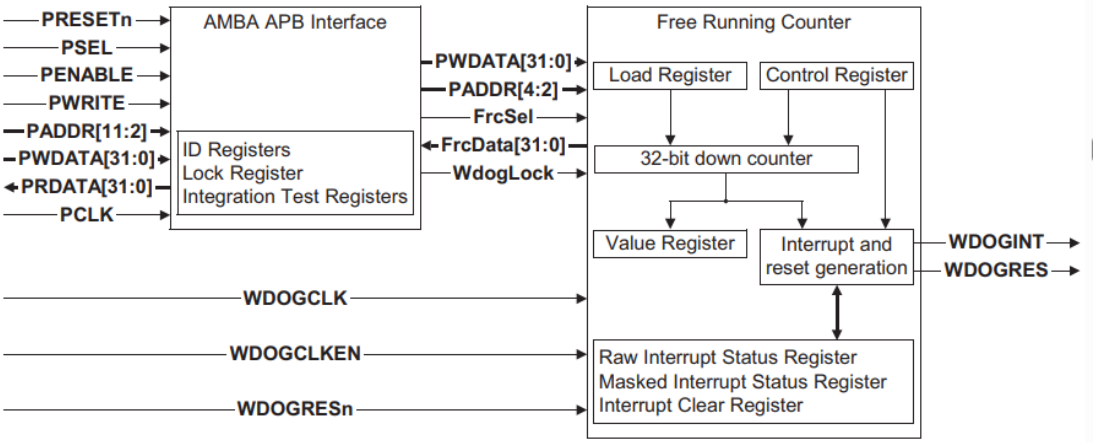
9.3.2 Watchdog module overview

The Watchdog module is based around a 32-bit down counter that is initialized from the Reload Register, WdogLoad. The counter decrements by one on each positive clock edge of **WDOGCLK** when

the clock enable **WDOGCLKEN** is HIGH. When the counter reaches zero, an interrupt is generated. On the next enabled **WDOGCLK** clock edge the counter is reloaded from the WdogLoad Register and the count down sequence continues. If the interrupt is not cleared by the time that the counter next reaches zero then the Watchdog module asserts the reset signal, **WDOGRES**, and the counter is stopped.
WDOGCLK can be equal to or be a sub-multiple of the **PCLK** frequency. However, the positive edges of **WDOGCLK** and **PCLK** must be synchronous and balanced.
The Watchdog module interrupt and reset generation can be enabled or disabled as required by use of the Control Register, WdogControl. When the interrupt generation is disabled then the counter is stopped. When the interrupt is re-enabled then the counter starts from the value programmed in WdogLoad, and not from the last count value.
Write access to the registers in the Watchdog module can be disabled by the use of the Watchdog module Lock Register, WdogLock. Writing a value of 0x1ACCE551 to the register enables write accesses to all of the other registers. Writing any other value disables write accesses to all registers except the Lock Register. This feature protects the Watchdog module registers from being spuriously changed by runaway software that might otherwise disable the Watchdog module operation.

9.3.3 Functional description

The Watchdog module block diagram is shown in Figure below.



AMBA APB interface

The AMBA APB slave interface generates read and write decodes for accesses to all registers in the Watchdog module. The Lock Register, WdogLock, is used to control the enabling of write accesses to all the other registers in order to ensure software cannot unintentionally disable the Watchdog module operation.

Free running counter block

The free running counter block contains the 32-bit down counter functionality and generates the interrupt and reset signal outputs. The counter and interrupt/reset logic is clocked independently of **PCLK** by **WDOGCLK** in conjunction with a clock enable, **WDOGCLKEN**, although there are

constraints on the relationship between **PCLK** and **WDOGCLK**. See Clock signals for details of these constraints.

Interface resets

The Watchdog module is reset by:

• the global reset signal, **PRESETn**

• a block specific reset signal, **WDOGRESn**.

PRESETn can be asserted asynchronously to **PCLK** but must be deasserted synchronously to the rising edge of **PCLK**. **PRESETn** is used to reset the state of the Watchdog module registers. The Watchdog module requires **PRESETn** to be asserted LOW for at least one period of **PCLK**. The values of the registers after reset are defined in Chapter 3 Programmer's Model.

WDOGRESn can be asserted asynchronously to **WDOGCLK** but must be deasserted synchronously to the rising edge of **WDOGCLK**. **WDOGRESn** is used to reset the state of registers in the **WDOGCLK** domain. The Watchdog module requires **WDOGRESn** to be asserted LOW for at least one period of **WDOGCLK**.

Clock signals

The Watchdog uses two input clocks:

PCLK This is used to time all APB accesses to the Watchdog module registers.

WDOGCLK

This clock, in conjunction with its clock enable, **WDOGCLKEN**, is used to clock the Watchdog module counter and its associated interrupt and reset generation logic. The Watchdog counter only decrements on a rising edge of **WDOGCLK** when **WDOGCLKEN** is HIGH. The relationship between **WDOGCLK** and **PCLK** must observe the following constraints:

• the rising edges of **WDOGCLK** must be synchronous and balanced with a rising edge of **PCLK**

• the **WDOGCLK** frequency cannot be greater than the **PCLK** frequency.

9.3.4 Operation

After the initial application and release of **PRESETn** and **WDOGRESn**, the Control Register is reset and interrupt and reset generation is disabled. The Lock Register, WdogLock, is initialized in the unlocked state so that write access to all Watchdog module registers is enabled. The Watchdog counter remains at its initial value (0xFFFFFFFF) until the interrupt generation is enabled by setting the INTEN bit in the WdogControl Register.

The WdogLoad Register must be programmed with the desired timeout interval before the Watchdog module is enabled. After the INTEN bit is set, the counter is loaded with the value in the WdogLoad Register on the next rising edge of **WDOGCLK** enabled by **WDOGCLKEN**. On each subsequent enabled **WDOGCLK** rising edge the counter decrements by one. When the counter reaches zero an interrupt is generated and the Watchdog interrupt signal, **WDOGINT**, is asserted. The counter is then reloaded from the value in the WdogLoad Register and starts another count down sequence.

The interrupt is cleared by a write of any data value to the WdogIntClr Register. This causes the counter to reload with the value held in the WdogLoad Register and another count down sequence starts. If the interrupt is not cleared before the counter next reaches zero then the **WDOGRES** signal is

asserted if the reset enable bit, RESEN, in the WdogControl Register is set. After the **WDOGRES** signal is asserted, the counter stops.

In a SoC, the **WDOGRES** signal is used to reset a system that has got into an unpredictable state. Therefore, the Watchdog module expects to be reset by **PRESETn** and **WDOGRESn** and the initialization procedure starts again.

To protect the Watchdog module registers from being changed unintentionally, the Lock Register, WdogLock, must be used to disable the write access to the Watchdog module registers after registers have been modified. To enable write access to all registers, write 0x1ACCE551 to the Lock Register, WdogLock. After writing to the required Watchdog registers, disable write access to all registers except the Lock Register by writing any value other than 0x1ACCE551 to the Lock Register. Reading the Lock Register returns the lock status rather than the 32-bit value written. Therefore, when write accesses are disabled, reading the lock register returns 0x00000001 (locked) otherwise the return value is 0x00000000 (unlocked).

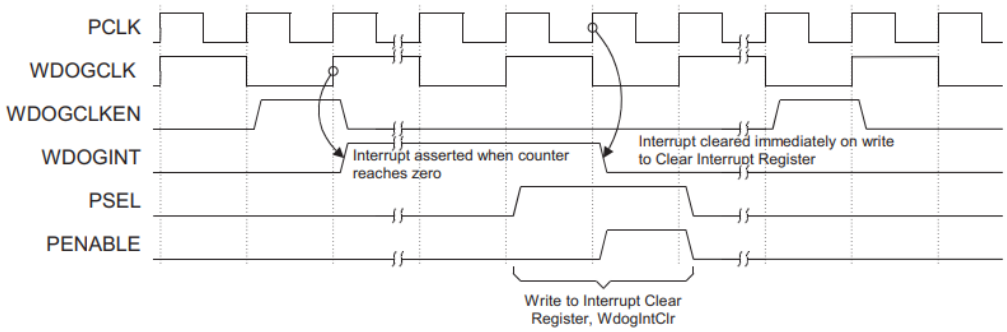
If the Load Register, WdogLoad, is written to with a new value while the Watchdog counter is decrementing then the counter is reloaded immediately with the new load value and continues decrementing from the new value. Writing to WdogLoad does not clear an active interrupt. An interrupt must be specifically cleared by writing to the Interrupt Clear Register, WdogIntClr.

If the interrupt generation is disabled by clearing the INTEN bit in the Control Register, WdogControl, the counter stops at its current value. When the interrupt generation is enabled again the counter reloads from the Load Register, WdogLoad, and starts to decrement.

Interrupt behavior

When the Watchdog raises an interrupt by asserting **WDOGINT**, the timing of this signal is generated from a rising clock edge of **WDOGCLK** enabled by **WDOGCLKEN**. When the interrupt is cleared by a write to the Interrupt Clear Register, WdogIntClr, the **WDOGINT** signal is deasserted immediately in the PCLK domain rather than waiting for the next enabled **WDOGCLK** rising edge.

Figure below shows an example of the timing for an interrupt being raised and cleared.



Programming the timeout interval

The Watchdog module counter is clocked by the rising edge of **WDOGCLK** when **WDOGCLKEN** is HIGH. In the case where **WDOGCLKEN** is permanently HIGH, the count rate is equal to the **WDOGCLK** frequency. When **WDOGCLKEN** is periodically pulsed HIGH for one **WDOGCLK** rising edge then the count rate is equal to the frequency of the **WDOGCLKEN** pulses. The frequency of enabled clock edges is referred to as the effective watchdog clock frequency and the period is referred

to as the effective watchdog clock period.

The Watchdog counter is reloaded from the Load Register, WdogLoad, whenever:
• the counter reaches zero
• the interrupt generation is enabled by setting the INTEN bit in the Control Register, WdogControl, when it was previously disabled
• an interrupt is cleared by writing to the Interrupt Clear register, WdogIntClr
• a new value is written to the Load Register, WdogLoad.

The time interval between the counter load occurring, and the counter reaching zero and generating an interrupt is given by the following expression:

Interrupt interval = (WdogLoad+1) x effective watchdog clock period

The initial reset value for WdogLoad is 0xFFFFFFFF and for an example effective watchdog frequency of 1MHz (period of 1ms) the interrupt interval is 4295 seconds.
The minimum valid value for WdogLoad is 0x00000001. If WdogLoad is set to 0x00000000, an interrupt is always generated immediately.
Table below shows examples of WdogLoad values required for a variety of interrupt intervals when the effective watchdog clock frequency is 1MHz.

Table 2-1 Example values for a 1MHz clock

Interrupt interval (ms)	WdogLoad	
	Hex	Decimal
1	0x000003E7	999
50	0x00001387	4999
100	0x0001869F	99999
500	0x0007A11F	499999
1000	0x000F423F	999999

9.3.5 Summary of registers

Summary of Watchdog module registers

Address	Type	Width	Reset value	Name
Base + 0x00	Read/write	32	0xFFFFFFFF	WdogLoad
Base + 0x04	Read-only	32	0xFFFFFFFF	WdogValue
Base + 0x08	Read/write	2	0x0	WdogControl
Base + 0x0C	Write-only	-	-	WdogIntClr

Base + 0x10	Read-only	1	0x0	WdogRIS
Base + 0x14	Read-only	1	0x0	WdogMIS
Base + 0x18-0xBFC	-	-	-	-
Base + 0xC00	Read/write	32	0x0	WdogLock

9.3.6 Register descriptions

Load Register, WdogLoad

This is a 32-bit read/write register that contains the value from which the counter is to decrement. When this register is written to, the count is immediately restarted from the new value. The minimum valid value for WdogLoad is 1. If WdogLoad is set to 0 then an interrupt is generated immediately.

Value Register, WdogValue

This read-only 32-bit register gives the current value of the decrementing counter.

Control register, WdogControl

This is a read/write register that enables the software to control the Watchdog module. Table below shows the bit assignment of the WdogControl Register.

Table 3-2 Control register bit assignment

Bit	Name	Type	Function
[31:2]	-	-	Reserved.
[1]	RESEN	Read/write	Enable Watchdog module reset output, **WDOGRES**. Acts as a mask for the reset output. Set HIGH to enable the reset, and LOW to disable the reset.
[0]	INTEN	Read/write	Enable the interrupt event, **WDOGINT**. Set HIGH to enable the counter and the interrupt, and set LOW to disable the counter and interrupt. Reloads the counter from the value in WdogLoad when the interrupt is enabled, and was previously disabled.

Interrupt Clear Register, WdogIntClr

A write of any value to this location clears the Watchdog module interrupt, and reloads the counter from the value in the WdogLoad Register.

Raw Interrupt Status Register, WdogRIS

This register indicates the raw interrupt status from the counter. The Raw Interrupt Status Register indicates that an interrupt has been raised by the Watchdog counter reaching zero. Table below shows the bit assignment of the WdogRIS Register

Table 3-3 Raw Interrupt Status Register bit assignment

Bit	Name	Type	Function
[31:1]	-	-	Reserved
[0]	WDOGRIS	Read	Raw interrupt status from the counter

Masked Interrupt Status Register, WdogMIS

This register indicates the masked interrupt status from the counter. This value is the logical AND of the raw interrupt status with the INTEN bit from the Control Register, and is the same value that is passed to the interrupt output pin WDOGINT. Table below shows the bit assignment of the WdogMIS Register.

Table 3-4 Interrupt Status Register bit assignment

Bit	Name	Type	Function
[31:1]	-	-	Reserved
[0]	WDOGMIS	Read	Enabled interrupt status from the counter

Lock Register, WdogLock

This register allows write-access to all other registers to be disabled. This is to prevent rogue software from disabling the Watchdog module operation. Writing a value of 0x1ACCE551 enables write access to all other registers. Writing any other value disables write accesses. A read from this register returns the lock status rather than the value written:

- 0 indicates that write access is enabled (not locked)
- 1 indicates that write access is disabled (locked).

Table below shows the bit assignment of the WdogLock Register.

Table 3-5 Lock Register bit assignment

Bit	Name	Type	Function
[31:0]	WDOGLOCK	Read/write	Writing `0x1ACCE551` to this register enables write access to all other registers. Writing any other value disables write access to all other registers. A read returns the lock status: `0x00000000` = write access to all other registers is enabled `0x00000001` = write access to all other registers is disabled.

Appendix A

Signal Descriptions

A.1 AMBA APB signals

The Watchdog module is connected to the AMBA APB as a bus slave. Table A-1 describes the APB interface signals.

Table A-1 AMBA APB signal descriptions

Name	Type	Source/Destination	Description
PRESETn	Input	Reset controller	APB bus reset signal, active LOW.
PCLK	Input	Clock generator	AMBA APB clock.
PENABLE	Input	APB bridge	AMBA APB enable signal. **PENABLE** is asserted HIGH for one cycle of **PCLK** to enable a bus transfer.
PSEL	Input	APB bridge	Watchdog module select signal from the decoder within the APB bridge. When HIGH this signal indicates the slave device is selected by the APB bridge, and that a data transfer is required.
PWRITE	Input	APB bridge	AMBA APB transfer direction signal, indicates a write access when HIGH, read access when LOW.
PADDR[11:2]	Input	APB bridge	Subset of AMBA APB address bus.
PWDATA[31:0]	Input	APB bridge	Unidirectional AMBA APB write data bus.
PRDATA[31:0]	Output	APB bridge	Unidirectional AMBA APB read data bus.

A.2 Non-AMBA signals

Table A-2 describes the Watchdog module non-AMBA signals.

Table A-2 Non-AMBA signals

Name	Type	Source/Destination	Description
WDOGCLK	Input	Clock generator	Watchdog module clock
WDOGCLKEN	Input	Clock generator	Watchdog module clock enable
WDOGRESn	Input	Reset generator	Watchdog module reset signal, active LOW
WDOGINT	Output	Interrupt controller	Watchdog module interrupt, active HIGH
WDOGRES	Output	Reset controller	Watchdog module timeout reset, active HIGH
SCANENABLE	Input	Test controller	Placeholder for Watchdog module scan enable signal
SCANINPCLK	Input	Test controller	Placeholder for Watchdog module input scan signal
SCANOUTPCLK	Output	Test controller	Placeholder Watchdog module output scan signal

10 Real-time clock (RTC)

The real-time clock is an independent timer. The RTC provides a set of continuously running counters which can be used, with suitable software, to provide a clock-calendar function. The counter values can be written to set the current time/date of the system.

The RTC core and clock configuration are in the Backup domain, which means that RTC setting and time are kept after reset or wakeup from Standby mode. After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access.

To enable access to the Backup registers and the RTC, proceed as follows:

(1) Enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register

(2) Set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

10.1 RTC main features:

Programmable pre-scaler : division factor up to 2^20

32-bit programmable counter for long-term measurement

The RTC clock source could be any of the following ones:

(1) CLKLOCAL from interconnect logic

(2) LSE oscillator clock

(3) LSI oscillator clock

Two separate reset types:

(1) The APB interface is reset by system reset

(2) The RTC Core (Pre-scaler, Alarm, Counter and Divider) is reset only by a Backup domain reset.

Three dedicate interrupt lines:

(1) Alarm interrupt, for generating a software programmable alarm interrupt.

(2) Seconds interrupt, for generating a periodic interrupt signal with a programmable period length.(up to 1 second).

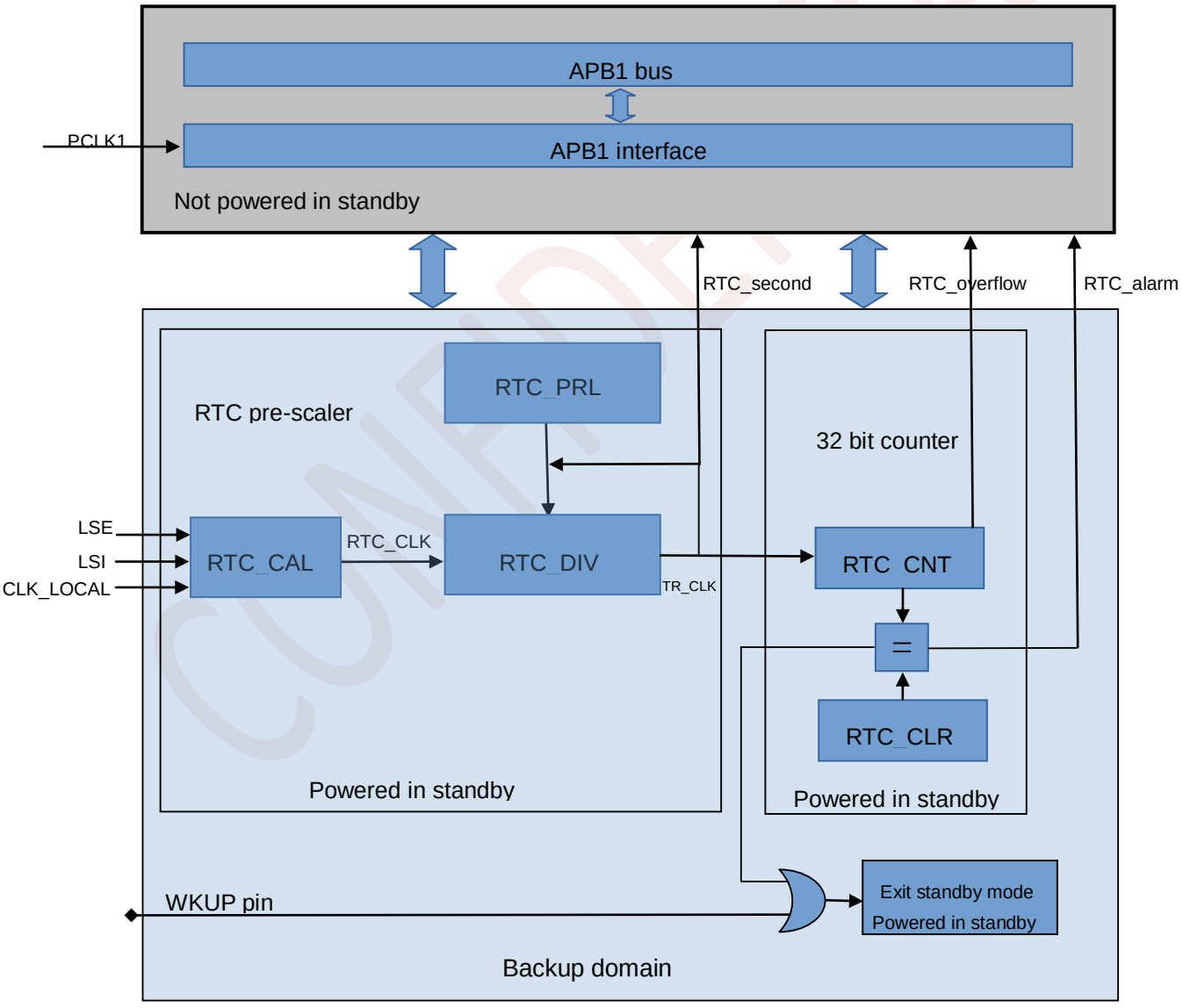
(3) Overflow interrupt, to detect when the internal programmable counter rolls over to zero.

10.2 RTC functional description

The RTC consists of two main units. The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus in read or write mode. The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC Core) consists of a chain of programmable counters made of two main blocks. The first block is the RTC pre-scaler block, which generates the RTC time base TR_CLK that can be programmed to have a period of up to 1 second. It includes a 20-bit programmable divider (RTC Pre-scaler). Every TR_CLK period, the RTC generates an interrupt (Second Interrupt) if it is enabled in the RTC_CR register. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented at the TR_CLK rate and compared with a programmable date (stored in the RTC_ALR register) in order to generate an alarm interrupt, if enabled in the RTC_CR control register.

RTC simplified block diagram



Resetting RTC registers

All system registers are asynchronously reset by a System Reset or Power Reset, except for RTC_PRL,

RTC_ALR, RTC_CNT, and RTC_DIV.
The RTC_PRL, RTC_ALR, RTC_CNT, and RTC_DIV registers are reset only by a Backup Domain reset.

Reading RTC registers

The RTC core is completely independent from the RTC APB1 interface.
Software accesses the RTC pre-scaler, counter and alarm values through the APB1 interface but the associated readable registers are internally updated at each rising edge of the RTC clock resynchronized by the RTC APB1 clock. This is also true for the RTC flags.
This means that the first read to the RTC APB1 registers may be corrupted (generally read as 0) if the APB1 interface has previously been disabled and the read occurs immediately after the APB1 interface is enabled but before the first internal update of the registers. This can occur if:
(1) A system reset or power reset has occurred
(2) The MCU has just woken up from Standby mode
(3) The MCU has just woken up from Stop mode
In all the above cases, the RTC core has been kept running while the APB1 interface was disabled (reset, not clocked or unpowered).
Consequently when reading the RTC registers, after having disabled the RTC APB1 interface, the software must first wait for the RSF bit (Register Synchronized Flag) in the RTC_CRL register to be set by hardware.
Note that the RTC APB1 interface is not affected by WFI and WFE low-power modes.

Configuring RTC registers

To write in the RTC_PRL, RTC_CNT, RTC_ALR registers, the peripheral must enter Configuration Mode. This is done by setting the CNF bit in the RTC_CRL register.
In addition, writing to any RTC register is only enabled if the previous write operation is finished. To enable the software to detect this situation, the RTOFF status bit is provided in the RTC_CR register to indicate that an update of the registers is in progress. A new value can be written to the RTC registers only when the RTOFF status bit value is ’1’.
Configuration procedure
1. Poll RTOFF, wait until its value goes to ‘1’
2. Set the CNF bit to enter configuration mode
3. Write to one or more RTC registers
4. Clear the CNF bit to exit configuration mode
5. Poll RTOFF, wait until its value goes to ‘1’ to check the end of the write operation.
The write operation only executes when the CNF bit is cleared; it takes at least three RTCCLK cycles to complete.

RTC flag assertion

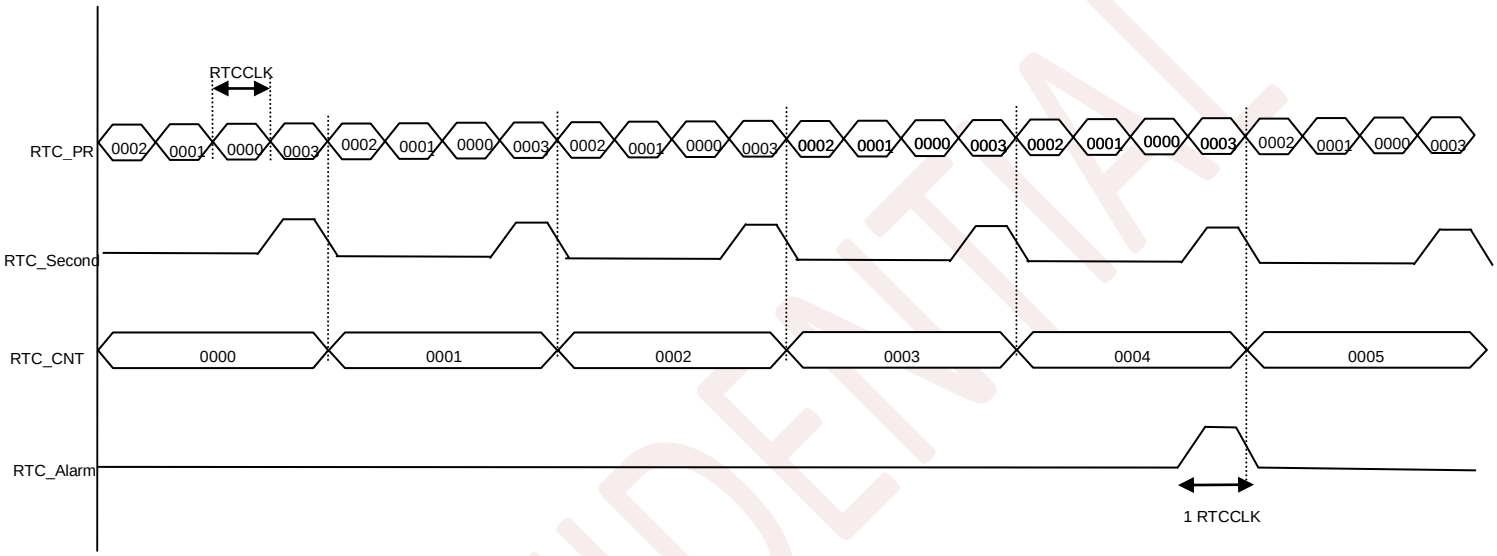
The RTC Second flag (SECF) is asserted on each RTC Core clock cycle before the update of the RTC Counter.
The RTC Overflow flag (OWF) is asserted on the last RTC Core clock cycle before the counter reaches 0x0000.
The RTC_Alarm and RTC Alarm flag (ALRF) are asserted on the last RTC Core clock cycle before

the counter reaches the RTC Alarm value stored in the Alarm register increased by one (RTC_ALR + 1). The write operation in the RTC Alarm and RTC Second flag must be synchronized by using one of the following sequences:

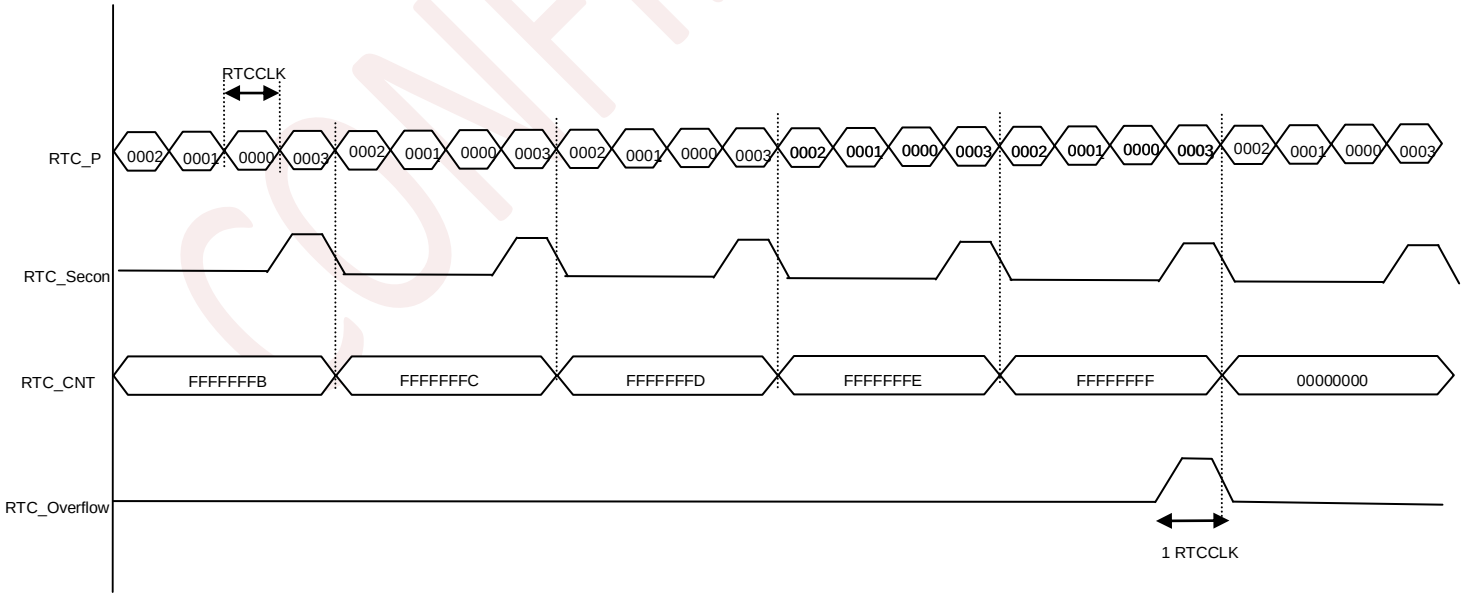
(1) Use the RTC Alarm interrupt and inside the RTC interrupt routine, the RTC Alarm and/or RTC Counter registers are updated.

(2) Wait for SECF bit to be set in the RTC Control register. Update the RTC Alarm and/or the RTC Counter register.

RTC second and alarm waveform example with PR=0003, ALARM=00004



RTC Overflow waveform example with PR=0003



11 DMA

11.1 Overview

The flexible general-purpose DMA controllers provide a hardware method of transferring data between peripherals and/or memory without intervention from the CPU, thereby freeing up bandwidth for other system functions. Three types of access method are supported: peripheral to memory, memory to peripheral, memory to memory.

Each channel is connected to fixed hardware DMA requests. The priorities of DMA channel requests are determined by software configuration and hardware channel number. Transfer size of source and destination are independent and configurable.

Features of the DMAC

The DMAC offers:

• Eight DMA channels. Each channel can support a unidirectional transfer.

• 16 DMA requests. The DMAC provides 16 peripheral DMA request lines.

• Single DMA and burst DMA request signals. Each peripheral connected to the DMAC can assert either a burst DMA request or a single DMA request. You set the DMA burst size by programming the DMAC.

• Memory-to-memory, memory-to-peripheral, peripheral-to-memory, and peripheral-to-peripheral transfers.

• Scatter or gather DMA support through the use of linked lists.

• Hardware DMA channel priority. Each DMA channel has a specific hardware priority. DMA channel 0 has the highest priority and channel 7 has the lowest priority. If requests from two channels become active at the same time, the channel with the highest priority is serviced first.

• AHB slave DMA programming interface. You program the DMAC by writing to the DMA control registers over the AHB slave interface.

• Two AHB bus masters for transferring data. Use these interfaces to transfer data when a DMA request goes active.

• 32-bit AHB master bus width.

• Incrementing or non-incrementing addressing for source and destination.

• Programmable DMA burst size. You can programme the DMA burst size to transfer data more efficiently. The burst size is usually set to half the size of the FIFO in the peripheral.

• Internal four word FIFO per channel.

• Supports eight, 16, and 32-bit wide transactions.

• Big-endian and little-endian support. The DMAC defaults to little-endian mode on reset

Raw interrupt status. You can read the DMA error and DMA count raw interrupt status prior to masking.

• Test registers for use in block and integration system level testing.

• Identification registers that uniquely identify the DMAC. An operating system can use these to automatically configure itself

11.2 Functional Overview

11.2.1 Functional description

The DMAC enables the following transactions:

• memory-to-memory

• memory-to-peripheral

• peripheral-to-memory

• peripheral-to-peripheral.

Each DMA stream provides unidirectional serial DMA transfers for a single source and destination. For example, a bidirectional port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral, and you can access them through the same AHB master, or one area by each master. Figure 2-1 shows a block diagram of the DMAC.

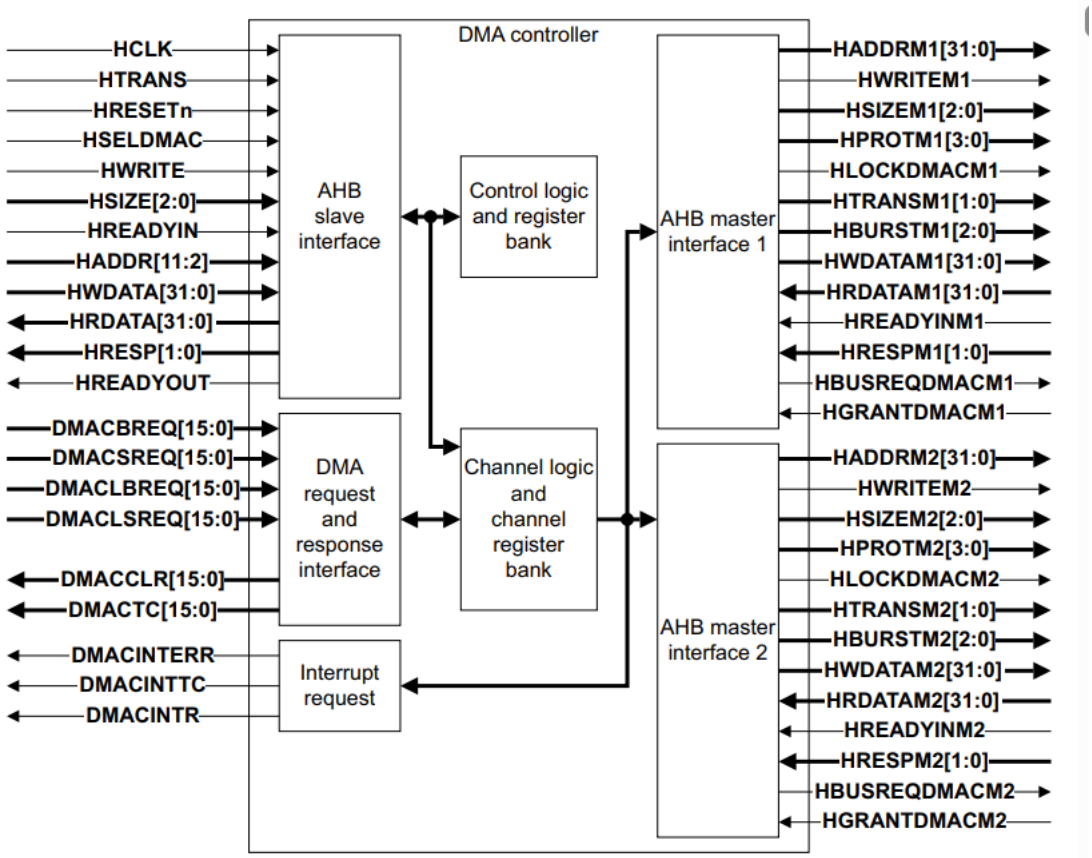


Figure 2-1 DMAC block diagram

11.2.1.1 AHB slave interface

All transactions on the AHB slave programming bus of the DMAC are 32 bits. This eliminates endian issues when programming the DMAC.

11.2.1.2 Control logic and register bank

The register block stores data written, or to be read across the AMBA AHB interface. Program the DMAC with this block using an AMBA AHB slave interface.

11.2.1.3 DMA request and response interface

See Appendix B DMA Interface for information on the DMA request and response interface.

11.2.1.4 Channel logic and channel register bank

The channel logic and channel register bank contains registers and logic that each DMA channel requires.

11.2.1.5 Interrupt request

The interrupt request generates interrupts to the core processor.

11.2.1.6 AHB master interfaces

The DMAC contains two full AHB masters. Figure below shows a block diagram of the two masters connected into a system. This enables, for example, the DMAC to transfer data directly from the memory connected to AHB port 1 to any AHB peripheral connected to AHB port 2. It also enables transactions between the DMAC and any APB peripheral to occur independently of transactions on AHB bus 1.

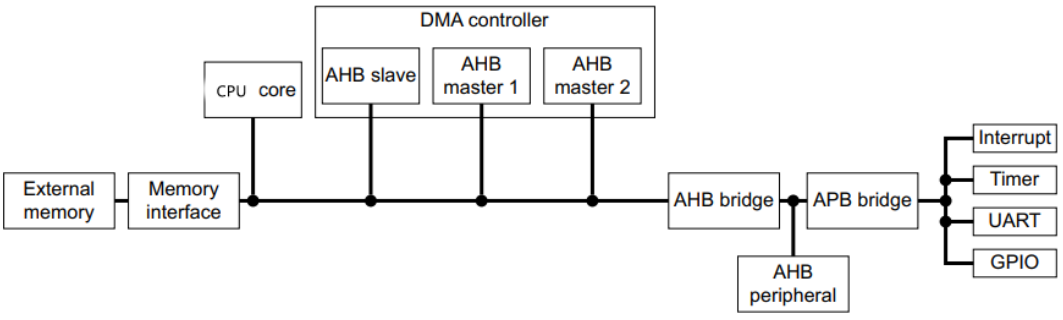


Figure 2-2 Dual AHB masters

The two AHB masters are each capable of dealing with all types of AHB transactions, including:

• Split, retry, and error responses from slaves. If a peripheral performs a split or retry, the DMAC stalls and waits until the transaction can complete.

- Locked transfers for source and destination of each stream.
- Setting of protection bits for transfers on each stream.

All AHB signals are connected as defined in the AHB Specification. The two AHB masters must be synchronous. They must use the same **HCLK**. Support for asynchronous AHB buses is not defined within the DMAC, and you must implement it by using wrappers, if required.

Bus and transfer widths

The two AHB masters are connected to buses of the same width. The default is a 32-bit bus. Source and destination transfers can be different widths, and can be the same width or narrower than the physical bus width. The DMAC packs or unpacks data as appropriate. The DMAC uses **HSIZE1** or **HSIZE2** to indicate the width of a transfer, and if this fails to match the width expected by the peripheral, then the peripheral can assert an error on **HRESP1** or **HRESP2**.

Endian behavior

The DMAC can cope with both little-endian and big-endian addressing. You can set the endianness of each AHB master individually.

Internally, the DMAC treats all data as a stream of bytes instead of 16-bit or 32-bit quantities. This means that when performing mixed-endian activity, where the endianness of the source and destination are different, byte swapping of the data within the 32-bit data bus occurs.

Note:

If you do not require byte swapping, avoid using different endianness between the source and destination addresses.

Error conditions

An error during a DMA transfer is flagged directly by the peripheral by asserting an Error response on the AHB bus during the transfer. The DMAC automatically disables the DMA stream after the current transfer has completed, and optionally generates an error interrupt to the CPU. You can mask this error interrupt.

11.2.1.7 Channel hardware

A dedicated hardware channel supports each stream, including source and destination controllers, and a FIFO. This enables better latency than a DMAC with only a single hardware channel shared between several DMA streams, and also simplifies the control logic.

11.2.1.8 Test registers

Test registers are provided for integration testing. You must not read or write to test registers during normal use. The integration testing verifies that the DMAC is connected into a system correctly, enabling you to write to and read each input and output.

11.2.1.9 DMA request priority

DMA channel priority is fixed. DMA channel 0 has the highest priority and DMA channel 7 has the lowest priority.

If the DMAC is transferring data for a lower priority channel, and then a higher priority channel goes active, it completes the number of transfers delegated to the master interface by the lower priority channel before switching over to transfer data for the higher priority channel. In the worst case, this is as large as one quadword.

The two lowest priority channels in the DMAC, 6 and 7, are designed so that they cannot saturate the AHB bus. If one of these lower priority channels goes active, the DMAC relinquishes the bus for one cycle each four transfers of the programmed WIDTH irrespective of the size of the transfer. For example, if the programmed size WIDTH is 8, then after four transfers of 8 bits the DMAC relinquishes the bus. This enables other AHB masters to access the bus.

It is recommended that memory-to-memory transactions use one of these low-priority channels or other lower priority AHB bus masters cannot access the bus during DMAC memory-to-memory transfer.

11.2.2 System considerations

Reducing the number of transactions that occur on the buses reduces the latency on the bus, improves system performance, and reduces power consumption. Therefore, the following design considerations are recommended:

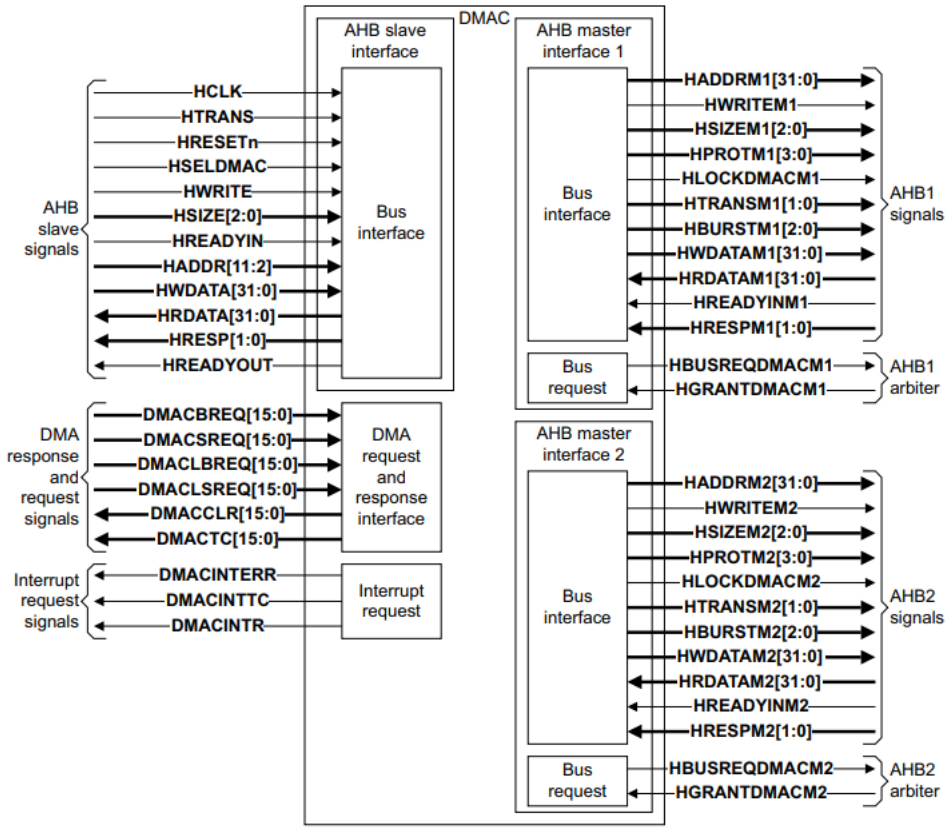
• All memory transactions are, in the standard configuration, 32 bits wide to improve bus efficiency.

• Peripherals with natural word sizes that are less than 32 bits must contain byte or halfword packing hardware so that all transactions can be made 32 bits wide.

• Slow peripherals that normally use wait states must contain FIFOs so you can transfer data at full speed using burst transfers.

11.2.3 System connectivity

Figure below shows how the DMAC connects to a system



11.2.3.1 AHB interfaces

The AHB slave and master interfaces all execute from the same clock, HCLK. Each master is entirely separate and there is no shared logic between them.

11.2.3.2 AHB slave interface

The AHB slave interface programs the DMAC. Figure 2-3 on page 2-13 shows the port-level connections of the AHB slave interface module.

11.2.3.3 AHB master interface

Unless otherwise stated, you must connect this interface as the AMBA Specification describes. You can set the AHB signals while performing DMA transfers.

Protection control

Software programs HPROT[3:0] bits for each DMA channel. The bits are set as follows:

HPROT[0]	Opcode, or data. This bit is hardcoded to Data-1.
HPROT[1]	User or privileged: user = 0 privileged = 1. Programmed by software. See Channel Control Registers. During LLI loads, HPROT[1] is made 1, privileged.
HPROT[2]	Bufferable or non-bufferable: non-bufferable = 0 bufferable = 1. Programmed by software. See Channel Control Registers. During LLI loads, HPROT[2] is made 0.
HPROT[3]	Cacheable or non-cacheable: non-cacheable = 0 cacheable = 1. Programmed by software. See Channel Control Registers. During LLI loads, HPROT[3] is made 1.

Peripherals can interpret the **HPROT** information as required to help perform efficient transactions. For example:

• You can use the **HPROT[1]** user or privileged bit to protect certain peripherals or memory spaces from user mode transactions.

• You can use the **HPROT[2]** bufferable or nonbufferable bit to indicate to an AMBA bridge that the write can complete in zero wait states on the source bus. This is without waiting for it to arbitrate for the destination bus, and for the slave to accept the data.

• An AMBA bridge can use the **HPROT[3]** cacheable or noncacheable bit so that on the first read of a burst of eight, it can transfer the whole burst of eight reads on the destination bus, rather than pass the transactions through one at a time.

Lock control

Set the lock bit by programming bit 16 in the DMACCxConfiguration Register. See Channel Configuration Registers.

When a burst occurs, the AHB arbiter must not degrant the master during the burst until the lock is deasserted. You can lock the DMAC for a single burst such as a long source fetch burst or a long destination drain burst. The DMAC does not usually assert the lock continuously for a source fetch burst followed by a destination drain burst.

There are situations when the DMAC asserts the lock for source transfers followed by destination transfers. This is possible when internal conditions in the DMAC enable it to perform a source fetch followed by a destination drain back-to-back, and when the following conditions are both met:

• Source width = destination width, and,

• Source burst size is a minimum of 4.

Bus width

The source width, SWidth, or destination width, DWidth, values in the DMACCxControl Register program the **HSIZE[1:0]** bits.

11.2.3.4 Interrupt generation logic

The DMAC generates the individual maskable active HIGH interrupts. A combined interrupt output is also generated as an OR function of the individual interrupt requests.

You can use the single combined interrupt with a system interrupt controller that provides another level of masking on a per-peripheral basis. This enables you to use modular device drivers that always know where to find the interrupt source control register bits.

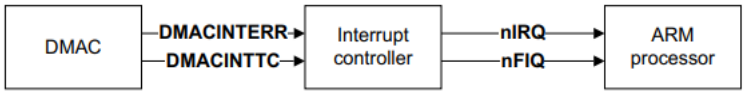
You can also use the individual interrupt requests with a system interrupt controller that provides masking for the outputs of each peripheral. In this way, a global interrupt service routine can read the entire set of sources from one wide register in the system interrupt controller. This is useful when the time to read from the peripheral registers is significant compared to the CPU clock speed in a real-time system.

The peripheral supports both of these methods.

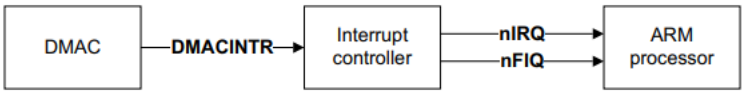
11.2.3.5 Interrupt controller connectivity

You can connect the interrupt request signals of the DMAC to an interrupt controller in one of two ways.

• For higher performance systems, you must connect the DMACINTERR and DMACINTTC interrupt request signals to the interrupt controller. Figure below shows connections to higher performance systems.



For lower performance systems, where the interrupt controller has fewer interrupt request input lines, you can use the DMACINTR interrupt request signal. Figure below shows connections to lower performance systems.



11.2.3.6 DMA request and response connectivity

Figure below shows how you can connect the DMA request and response signals to a peripheral. However, some peripherals do not use all of these signals. You can leave output signals that are not required unconnected and you can tie input signals that are not required LOW. See Appendix B DMA Interface for more information on the DMA request and response interface.

Figure below shows an example of a peripheral that uses all of the DMA request and grant signals

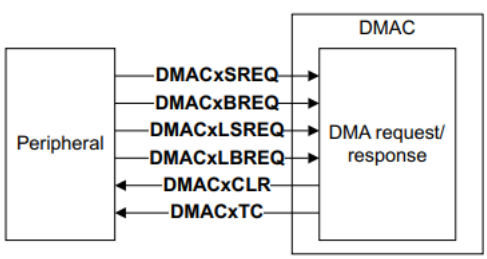
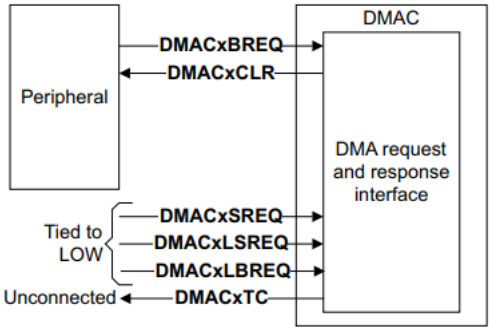


Figure below shows a simple example of connectivity.



11.2.4 Software considerations

You must take into account the following software considerations when programming the DMAC:

• There must not be any write-operation to Channel registers in an active channel after the channel enable is made HIGH. If you must reprogram any DMAC channel parameters, you must reprogram after disabling the DMAC channel.

• If the source width is less than the destination width, the TransferSize value multiplied by the source width must be an integral multiple of the destination width.

• When the source peripheral is the flow controller and the source width is less than the destination width, the number of transfers that the source peripheral performs, before asserting an **DMACLSREQ** or **DMACLBREQ**, must be so that the number of transfers multiplied by the source width is an integral multiple of the destination width. If this case is violated, the data can get stuck and lost in the FIFO causing UNPREDICTABLE results. You can abort the transfer by disabling the relevant DMAC channel.

• You must not program the SrcPeripheral and DestPeripheral bit fields in the DMACCxConfig Register with any value greater than 15.

• The SWidth and DWidth bit fields in the DMACCxControl Register must not indicate more than a 32-bit wide peripheral.

• After the software disables a channel by clearing the ChannelEnable bit in the DMACCxConfig Register, see Channel Configuration Registers on page 3-27, it must re-enable the bit only after it has polled a 0 in the corresponding DMACEnbldChns Register bit, see Enabled Channel Register. This is because the actual disabling does not immediately happen with the clearing of ChannelEnable bit. You must accommodate the latency of the ongoing AHB burst.

• The LLI field in the DMACCxLLIReg Register must not indicate an address greater than 0xFFFFFFF0, otherwise the four-word LLI burst wraps over at 0x00000000 and the LLI data structure is not in contiguous memory locations. See Channel Linked List Item Registers.

• When the transfer size programmed in the DMAC is greater than the depth of the FIFO in a source or destination peripheral, you must only program the DMAC for non-incrementing address generation.

• A peripheral is expected to deassert any **DMACSREQ**, **DMACBREQ**, **DMACLSREQ**, or **DMACLBREQ** signals on receiving the **DMACCLR** signal irrespective of the request the **DMACCLR** was asserted in response to. This is because **DMACCLR** is not specific to a single-request signal, **DMACSREQ**, or burst-request signal, **DMACSBEQ**. The handshaking of **DMACCLR** is achieved with a logical OR of all the DMA requests in the DMAC.

• If you program the TransferSize field in the DMACCxControl Register, see Channel Control Registers, as zero, and the DMAC is the flow controller, the TransferSize field has no meaning in other flow-control modes, then the channel does not initiate any transfers. It is your responsibility to disable the channel by writing into the channel enable bit of the DMACCxConfig Register and reprogramming the channel again.

• You must not run the normal read-write tests on the DMACCxControl Register, see Channel Control Registers, because the TransferSize field is not a typical write and read-back register field. While writing, the TransferSize bit-field is like a control register because it determines how many transfers the DMAC performs. However, during read-back, TransferSize behaves like a status register because it returns the number of remaining transfers in terms of source width. So when TransferSize is read back, it returns the number of destination-transfer-completed stored in a separate counter called TrfSizeDst multiplied by a factor. The same physical register is not being written into and read from, and normal write and read-back tests are not applicable.

• In the destination flow control mode, with peripheral-to-peripheral transfer, if sufficient data is present in the channel FIFO to service a **DMACLSREQ** or **DMACLBREQ** request raised by a destination peripheral without requiring data to be fetched from the source peripheral, then the source peripheral is issued a **DMACTC**.

• For destination flow controlled case, peripheral-to-peripheral transfer, with DWidth < SWidth, the number of data bytes requested by the destination peripheral must be an integral multiple of Swidth expressed in bytes. If you do not ensure this, then the DMAC might fetch more data from the source peripheral than is required. This can result in data loss

• At the end of accesses corresponding to low-priority channels, an IDLE cycle is inserted on the AHB bus to enable other masters to access the bus. This ensures that a low-priority channel does not monopolize the bus. It does, however, mean that the bus might be occupied by transactions corresponding to a low priority for up to 16 cycles in the worst case. This applies to all transfer configurations, including memory-to-memory transfers.

11.3 Programmer's Model

11.3.1 About the programmer's model

The DMAC enables the following types of transactions:

• memory-to-memory

• memory-to-peripheral

• peripheral-to-memory

• peripheral-to-peripheral.

Each DMA stream is configured to provide unidirectional DMA transfers for a single source and destination.

For example, a bidirectional serial port requires one stream for transmit and one for receive. The source and destination areas can each be either a memory region or a peripheral, and you can access them through the same AHB master, or one area by each master.

The base address of the DMAC is not fixed, and can be different for any particular system implementation. However, the offset of any particular register from the base address is fixed.

11.3.2 Programming the DMAC

11.3.2.1 Enabling the DMAC

Enable the DMAC by setting the DMA Enable, E, bit in the DMACConfiguration Register. See Configuration Register

11.3.2.2 Disabling the DMAC

To disable the DMAC:

1. Read the DMACEnbldChns Register and ensure that you have disabled all the DMA channels. If any channels are active, see Disabling a DMA channel.

2. Disable the DMAC by writing 0 to the DMA Enable bit in the DMACConfiguration Register. See Configuration Register.

11.3.2.3 Enabling a DMA channel

Enable the DMA channel by setting the Channel Enable bit in the relevant DMA channel Configuration Register. See Channel Configuration Registers.

Note:

You must fully initialize the channel before you enable it. Additionally, you must set the Enable bit of the DMAC before you enable any channels.

11.3.2.4 Disabling a DMA channel

You can disable a DMA channel in the following ways:

• Write directly to the Channel Enable bit

• Use the Active and Halt bits in conjunction with the Channel Enable bit.

• Wait until the transfer completes. The channel is then automatically disabled.

Disabling a DMA channel and losing data in the FIFO

Clear the relevant Channel Enable bit in the relevant channel Configuration Register. See Channel Configuration Registers. The current AHB transfer, if one is in progress, completes and the channel is disabled.

Disabling a DMA channel without losing data in the FIFO

To disable a DMA channel without losing data in the FIFO:

1. Set the Halt bit in the relevant channel Configuration Register. See Channel Configuration Registers. This causes any subsequent DMA requests to be ignored.

2. Poll the Active bit in the relevant channel Configuration Register until it reaches 0. This bit indicates whether there is any data in the channel that has to be transferred.

3. Clear the Channel Enable bit in the relevant channel Configuration Register.

11.3.2.5 Setting up a new DMA transfer

To set up a new DMA transfer:

1. If the channel is not set aside for the DMA transaction:

   a. Read the DMACEnbldChns Register and determine the channels that are inactive. See Enabled Channel Register.

   b. Choose an inactive channel that has the necessary priority.

2. Program the DMAC.

11.3.2.6 Halting a DMA channel

Set the Halt bit in the relevant DMA channel Configuration Register. The current source request is serviced. Any subsequent source DMA requests are ignored until the Halt bit is cleared.

11.3.2.7 Programming a DMA channel

To program a DMA channel:

1. Choose a free DMA channel with the necessary priority. DMA channel 0 has the highest priority and DMA channel 7 has the lowest priority.

2. Clear any pending interrupts on the channel you want to use by writing to the DMACIntTCClear and DMACIntErrClr Registers. See Interrupt Terminal Count Clear Register and Interrupt Error Clear Register. The previous channel operation might have left interrupts active.

3. Write the source address into the DMACCxSrcAddr Register. See Channel Source Address Registers.

4. Write the destination address into the DMACCxDestAddr Register. See Channel Destination Address Registers.

5. Write the address of the next LLI into the DMACCxLLI Register. See Channel Linked List Item Registers. If the transfer consists of a single packet of data, you must write 0 into this register.

6. Write the control information into the DMACCxControl Register. See Channel Control Registers.

7. Write the channel configuration information into the DMACCxConfiguration Register. See Channel Configuration Registers. If the Enable bit is set, then the DMA channel is automatically enabled.

11.3.3 Summary of registers

Name	Address (base+)	Type	Reset value	Description
DMACIntStatus	0x000	RO	0x00	See *Interrupt Status Register*

DMACIntTCStatus	0x004	RO	0x00	See *Interrupt Terminal Count Status Register*
DMACIntTCClear	0x008	WO	-	See *Interrupt Terminal Count Clear Register*
DMACIntErrorStatus	0x00C	RO	0x00	See *Interrupt Error Status Register*
DMACIntErrClr	0x010	WO	-	See *Interrupt Error Clear Register*
DMACRawIntTCStatus	0x014	RO	-	See *Raw Interrupt Terminal Count Status Register*
DMACRawIntErrorStatus	0x018	RO	-	See *Raw Error Interrupt Status Register*
DMACEnbldChns	0x01C	RO	0x00	See *Enabled Channel Registe*
DMACSoftBReq	0x020	R/W	0x0000	See *Software Burst Request Register*
DMACSoftSReq	0x024	R/W	0x0000	See *Software Single Request Register*
DMACSoftLBReq	0x028	R/W	0x0000	See *Software Last Burst Request Register*
DMACSoftLSReq	0x02C	R/W	0x0000	See *Software Last Single Request Register*
DMACConfiguration	0x030	R/W	0b000	See *Configuration Register*
DMACSync	0x34	R/W	0x0000	See *Synchronization Register*
	0x38 – 0x0EC-			Reserved
From 0x100, 8 channel of each below				
DMACC0SrcAddr	0x100	R/W	0x00000000	See *Channel Source Address Registers*
DMACC0DestAddr	0x104	R/W	0x00000000	See *Channel Destination Address Registers*
DMACC0LLI	0x108	R/W	0x00000000	See *Channel Linked List Item Registers*
DMACC0Control	0x10C	R/W	0x00000000	See *Channel Control Registers*
DMACC0Configuration	0x110	R/W	0x	See *Channel*

			00000	*Configuration Registers*
DMACC1SrcAddr	0x120	R/W	0x00000000	See Channel Source Address Registers
DMACC1DestAddr	0x124	R/W	0x00000000	See Channel Destination Address Registers
DMACC1LLI	0x128	R/W	0x00000000	See Channel Linked List Item Registers
DMACC1Control	0x12C	R/W	0x00000000	See Channel Control Registers
DMACC1Configuration	0x130	R/W	0x00000	See Channel Configuration Registers
DMACC2SrcAddr	0x140	R/W	0x00000000	See Channel Source Address Registers
DMACC2DestAddr	0x144	R/W	0x00000000	See Channel Destination Address Registers
DMACC2LLI	0x148	R/W	0x00000000	See Channel Linked List Item Registers
DMACC2Control	0x14C	R/W	0x00000000	See Channel Control Registers
DMACC2Configuration	0x150	R/W	0x00000	See Channel Configuration Registers
DMACC3SrcAddr	0x160	R/W	0x00000000	See Channel Source Address Registers
DMACC3DestAddr	0x164	R/W	0x00000000	See Channel Destination Address Registers
DMACC3LLI	0x168	R/W	0x00000000	See Channel Linked List Item Registers
DMACC3Control	0x16C	R/W	0x00000000	See Channel Control Registers
DMACC3Configuration	0x170	R/W	0x00000	See Channel Configuration Registers
DMACC4SrcAddr	0x180	R/W	0x	See Channel Source

			00000000	Address Registers
DMACC4DestAddr	0x184	R/W	0x00000000	See Channel Destination Address Registers
DMACC4LLI	0x188	R/W	0x00000000	See Channel Linked List Item Registers
DMACC4Control	0x18C	R/W	0x00000000	See Channel Control Registers
DMACC4Configuration	0x190	R/W	0x00000	See Channel Configuration Registers
DMACC5SrcAddr	0x1A0	R/W	0x00000000	See Channel Source Address Registers
DMACC5DestAddr	0x1A4	R/W	0x00000000	See Channel Destination Address Registers
DMACC5LLI	0x1A8	R/W	0x00000000	See Channel Linked List Item Registers
DMACC5Control	0x1AC	R/W	0x00000000	See Channel Control Registers
DMACC5Configuration	0x1B0	R/W	0x00000	See Channel Configuration Registers on page 3-27
DMACC6SrcAddr	0x1C0	R/W	0x00000000	See Channel Source Address Registers
DMACC6DestAddr	0x1C4	R/W	0x00000000	See Channel Destination Address Registers
DMACC6LLI	0x1C8	R/W	0x00000000	See Channel Linked List Item Registers
DMACC6Control	0x1CC	R/W	0x00000000	See Channel Control Registers
DMACC6Configuration	0x1D0	R/W	0x00000	See Channel Configuration Registers
DMACC7SrcAddr	0x1E0	R/W	0x000000	See Channel Source Address Registers

			00	
DMACC7DestAddr	0x1E4	R/W	0x00000000	See Channel Destination Address Registers
DMACC7LLI	0x1E8	R/W	0x00000000	See Channel Linked List Item Registers
DMACC7Control	0x1EC	R/W	0x00000000	See Channel Control Registers
DMACC7Configuration	0x1F0	R/W	0x00000	ee Channel Configuration Registers
DMACPeriphID0	0xFE0	RO	0x80	See DMACPeriphID0 Register
DMACPeriphID1	0xFE4	RO	0x10	See DMACPeriphID1 Register
DMACPeriphID2	0xFE8	RO	0x04	See DMACPeriphID2 Register
DMACPeriphID3	0xFEC	RO	0x0A	See DMACPeriphID3 Register
DMACPCellID0	0xFF0	RO	0x0D	See DMACPCellID0 Register
DMACPCellID1	0xFF4	RO	0xF0	See DMACPCellID1 Register
DMACPCellID2	0xFF8	RO	0x05	See DMACPCellID2 Register
DMACPCellID3	0xFFC	RO	0xB1	See DMACPCellID3 Register
DMACITCR	0x500	R/W	0x0	See Test Control Register
DMACITOP1	0x504	R/W	0x0000	See Integration Test Output Register 1
DMACITOP2	0x508	R/W	0x0000	See Integration Test Output Register 2
DMACITOP3	0x50C	R/W	0x0	See Integration Test Output Register 3

11.3.4 Register descriptions

11.3.4.1 Interrupt Status Register

The read-only DMACIntStatus Register, with address offset of 0x000, shows the status of the interrupts after masking. A HIGH bit indicates that a specific DMA channel interrupt request is active. You can generate the request from either the error or terminal count interrupt requests. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	IntStatus	Status of the DMA interrupts after masking

11.3.4.2 Interrupt Terminal Count Status Register

The read-only DMACIntTCStatus Register, with address offset of 0x004, indicates the status of the terminal count after masking. You must use this register in conjunction with the DMACIntStatus Register if you use the combined interrupt request, **DMACINTR**, to request interrupts. If you use the **DMACINTTC** interrupt request, then you only have to read the DMACIntTCStatus Register to ascertain the source of the interrupt request. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	IntTCStatus	Interrupt terminal count request status

11.3.4.3 nterrupt Terminal Count Clear Register

The write-only DMACIntTCClear Register, with address offset of 0x008, clears a terminal count interrupt request. When writing to this register, each data bit that is set HIGH causes the corresponding bit in the Status Register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Undefined. Write as zero.
[7:0]	IntTCClear	Terminal count request clear.

11.3.4.4 Interrupt Error Status Register

The read-only DMACIntErrorStatus Register, with address offset of 0x00C, indicates the status of the error request after masking. You must use this register in conjunction with the DMACIntStatus Register if you use the combined interrupt request, DMACINTR, to request interrupts. If you use the DMACINTERR interrupt request, then only read the DMACIntErrorStatus Register. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	IntErrorStatus	Interrupt error status

11.3.4.5 Interrupt Error Clear Register

The write-only DMACIntErrClr Register, with address offset of 0x010, clears the error interrupt requests. When writing to this register, each data bit that is HIGH causes the corresponding bit in the Status Register to be cleared. Data bits that are LOW have no effect on the corresponding bit in the register.

Bits	Name	Function
[31:8]	-	Undefined. Write as zero.
[7:0]	IntErrClr	Interrupt error clear.

Figure below shows the register bit assignments.

11.3.4.6 Raw Interrupt Terminal Count Status Register

The read-only DMACRawIntTCStatus Register, with address offset of 0x014, indicates the DMA channels that are requesting a transfer complete, terminal count interrupt, prior to masking. A HIGH bit indicates that the terminal count interrupt request is active prior to masking. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	RawIntTCStatus	Status of the terminal count interrupt prior to masking

11.3.4.7 Raw Error Interrupt Status Register

The read-only DMACRawIntErrorStatus Register, with address offset of 0x018, indicates the DMA channels that are requesting an error interrupt prior to masking. A HIGH bit indicates that the error interrupt request is active prior to masking. Figure below shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	RawIntErrorStatus	Status of the error interrupt prior to masking

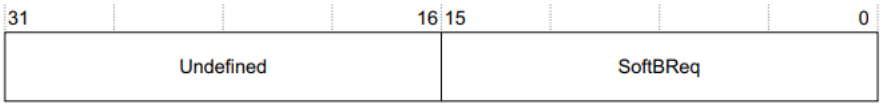
11.3.4.8 Enabled Channel Register

The read-only DMACEnbldChns Register, with address offset of 0x01C, indicates the DMA channels that are enabled, as indicated by the Enable bit in the DMACCxConfiguration Register. A HIGH bit indicates that a DMA channel is enabled. A bit is cleared on completion of the DMA transfer. Figure 3-8 shows the register bit assignments.

Bits	Name	Function
[31:8]	-	Read undefined
[7:0]	EnabledChannels	Channel enable status

11.3.4.9 Software Burst Request Register

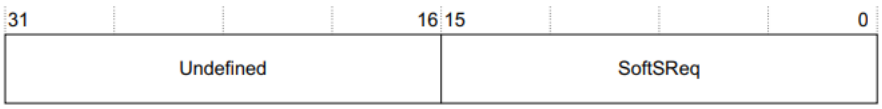
The read/write DMACSoftBReq Register, with address offset of 0x020, enables DMA burst requests to be generated by software. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting DMA burst transfers. You can generate a request from either a peripheral or the software request register. Figure below shows the register bit assignments.



Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	SoftBReq	Software burst request.

11.3.4.10 Software Single Request Register

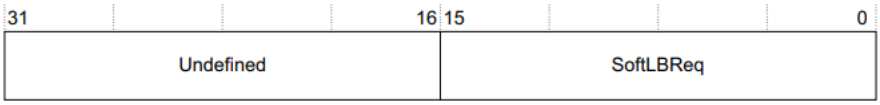
The read/write DMACSoftSReq Register, with address offset of 0x024, enables DMA single requests to be generated by software. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting single DMA transfers. You can generate a request from either a peripheral or the software request register. Figure below shows the register bit assignments.



Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	SoftSReq	Software single request.

11.3.4.11 Software Last Burst Request Register

The read/write DMACSoftLBReq Register, with address offset of 0x028, enables software to generate DMA last burst requests. You can generate a DMA request for each source by writing a 1 to the

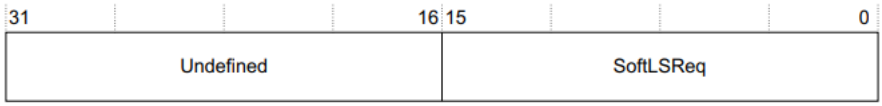


Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	SoftLBReq	Software last burst request.

corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting last burst DMA transfers. You can generate a request from either a peripheral or the software request register. Figure below shows the register bit assignments.

11.3.4.12 Software Last Single Request Register

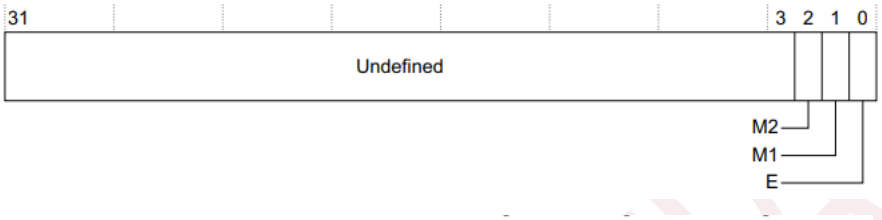
The read/write DMACSoftLSReq Register, with address offset of 0x02C, enables software to generate DMA last single requests. You can generate a DMA request for each source by writing a 1 to the corresponding register bit. A register bit is cleared when the transaction has completed. Writing 0 to this register has no effect. Reading the register indicates the sources that are requesting last single DMA transfers. You can generate a request from either a peripheral or the software request register. Figure below shows the register bit assignments.



Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	SoftLSReq	Software last single request.

11.3.4.13 Configuration Register

The read/write DMACConfiguration Register, with address offset of 0x030, configures the operation of the DMAC. You can alter the endianness of the individual AHB master interfaces by writing to the M1 and M2 bits of this register. The M1 bit enables you to alter the endianness of AHB master interface 1. The M2 bit enables you to alter the endianness of AHB master interface 2. The AHB master interfaces are set to little-endian mode on reset.



Bits	Name	Function
[31:3]	-	Read undefined. Write as zero.
[2]	M2	AHB Master 2 endianness configuration: 0 = little-endian mode 1 = big-endian mode. This bit is reset to 0.
[1]	M1	AHB Master 1 endianness configuration: 0 = little-endian mode 1 = big-endian mode. This bit is reset to 0.

Bits	Name	Function
[0]	E	DMAC enable: 0 = disabled 1 = enabled. This bit is reset to 0. Disabling the DMAC reduces power consumption.

11.3.4.14 Synchronization Register

The read/write DMACSync Register, with address offset of 0x034, enables or disables synchronization logic for the DMA request signals.

The DMA request signals consist of:

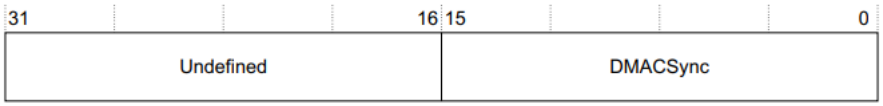
- DMACBREQ[15:0]
- DMACSREQ[15:0]
- DMACLBREQ[15:0]
- DMACLSREQ[15:0].

A bit set to 0 enables the synchronization logic for a particular group of DMA requests. A bit set to 1 disables the synchronization logic for a particular group of DMA requests. This register is reset to 0, and synchronization logic enabled

Note:

1. It is illegal for a peripheral to give a new DMACSREQ or DMACBREQ signal while DMACCLR is HIGH.

2. You must use synchronization logic when the peripheral generating the DMA request runs on a different clock to the DMAC. For peripherals running on the same clock as the DMAC, disabling the synchronization logic improves the DMA request response time. If necessary, synchronize the DMA response signals, DMACCLR and DMACTC, in the peripheral.



Bits	Name	Function
[31:16]	-	Read undefined. Write as zero.
[15:0]	DMACSync	DMA synchronization logic for DMA request signals enabled or disabled. A LOW bit indicates that the synchronization logic for the request signals is enabled. A HIGH bit indicates that the synchronization logic is disabled.

11.3.4.15 Channel registers

The channel registers are for programming a DMA channel. These registers consist of:

- eight DMACCxSrcAddr Registers
- eight DMACCxDestAddr Registers
- eight DMACCxLLI Registers
- eight DMACCxControl Registers
- eight DMACCxConfiguration Registers.

When performing scatter/gather DMA, the first four registers are automatically updated.

Note：

Unpredictable behavior can result if you update the channel registers when a transfer is taking place. If you want to change the channel configurations, you must disable the channel first and then reconfigure the relevant register.

Channel Source Address Registers

The eight read/write DMACCxSrcAddr Registers, with address offsets of 0x100, 0x120, 0x140, 0x160, 0x180, 0x1A0, 0x1C0, and 0x1E0 respectively, contain the current source address, byte-aligned, of the data to be transferred. Software programs each register directly before the appropriate channel is enabled

When the DMA channel is enabled, this register is updated:

- as the source address is incremented
- by following the linked list when a complete packet of data has been transferred.

Reading the register when the channel is active does not provide useful information. This is because by the time the software has processed the value read, the channel might have progressed. It is intended to be read-only when the channel has stopped, and in such case, it shows the source address of the last item read.

Bits	Name	Function
[31:0]	SrcAddr	DMA source address

Channel Destination Address Registers

The eight read/write DMACCxDestAddr Registers, with address offsets of 0x104, 0x124, 0x144, 0x164, 0x184, 0x1A4, 0x1C4, and 0x1E4 respectively, contain the current destination address, byte-aligned, of the data to be transferred.

Software programs each register directly before the channel is enabled. When the DMA channel is enabled, the register is updated as the destination address is incremented and by following the linked list when a complete packet of data has been transferred. Reading the register when the channel is active does not provide useful information. This is because by the time the software has processed the value read, the channel might have progressed. It is intended to be read-only when a channel has stopped. In this case, it shows the destination address of the last item read.

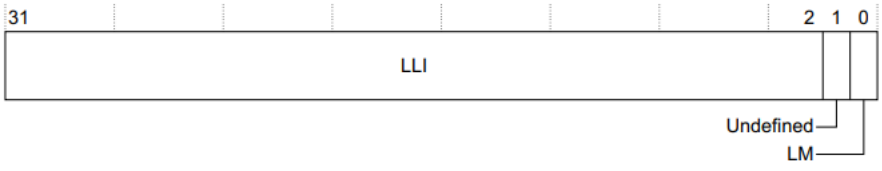
Bits	Name	Function
[31:0]	DestAddr	DMA destination address

Channel Linked List Item Registers

The eight read/write DMACCxLLI Registers, with address offsets of 0x108, 0x128, 0x148, 0x168, 0x188, 0x1A8, 0x1C8, and 0x1E8 respectively, contain a word-aligned address of the next LLI. If the LLI is 0, then the current LLI is the last in the chain, and the DMA channel is disabled after all DMA transfers associated with it are completed.

Note：

Programming this register when the DMA channel is enabled has unpredictable results.

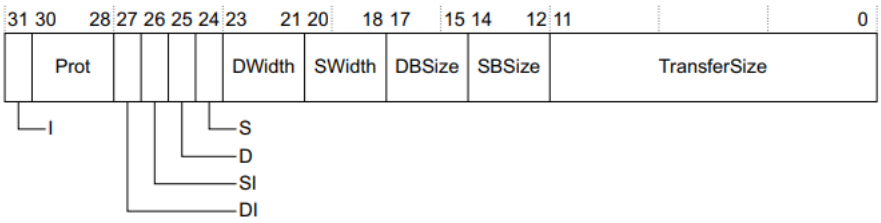


Bits	Name	Function
[31:2]	LLI	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.
[1]	-	Read undefined. Write as zero.
[0]	LM	AHB master select for loading the next LLI LM = 0 = AHB Master 1 LM = 1 = AHB Master 2.

Channel Control Registers

The eight read/write DMACCxControl Registers, with address offsets of 0x010C, 0x12C, 0x14C, 0x16C, 0x18C, 0x1AC, 0x1CC, and 0x1EC respectively, contain DMA channel control information such as the transfer size, burst size, and transfer width. Software programs each register directly before the DMA channel is enabled.

When the channel is enabled, the register is updated by following the linked list when a complete packet of data has been transferred. Reading the register while the channel is active does not give useful information. This is because by the time that software has processed the value read, the channel might have progressed. It is intended to be read-only when a channel has stopped.



Bits	Name	Function
[31]	I	*Terminal count* interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.
[30:28]	Prot	Protection.
[27]	DI	Destination increment. When set, the destination address is incremented after each transfer.
[26]	SI	Source increment. When set, the source address is incremented after each transfer.

Bits	Name	Function
[25]	D	Destination AHB master select: 0 = AHB master 1 selected for the destination transfer 1 = AHB master 2 selected for the destination transfer.
[24]	S	Source AHB master select: 0 = AHB master 1 selected for the source transfer 1 = AHB master 2 selected for the source transfer.
[23:21]	DWidth	Destination transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data when required.
[20:18]	SWidth	Source transfer width. Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data when required.
[17:15]	DBSize	Destination burst size. Indicates the number of transfers that make up a destination burst transfer request. You must set this value to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the **DMACxBREQ** signal goes active in the destination peripheral. The burst size is not related to the AHB **HBURST** signal.
[14:12]	SBSize	Source burst size. Indicates the number of transfers that make up a source burst. You must set this value to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the **DMACxBREQ** signal goes active in the source peripheral. The burst size is not related to the AHB **HBURST** signal.
[11:0]	TransferSize	Transfer size. A write to this field sets the size of the transfer when the DMAC is the flow controller. This value counts down from the original value to zero, and so its value indicates the number of transfers left to complete. A read from this field provides the number of transfers still to be completed on the destination bus. Reading the register when the channel is active does not give useful information because by the time the software has processed the value read, the channel might have progressed. Only use it when a channel is enabled, and then disabled. The *ARM PrimeCell DMA Controller (PL080) Design Manual* provides more information about the use of this field. Program the transfer size value to zero if the DMAC is not the flow controller. If you program the TransferSize to a non-zero value, the DMAC might attempt to use this value instead of ignoring the TransferSize.

Table below lists the values of the DBSize or SBSsize bits and their corresponding burst sizes.

Bit value of DBSize or SBSize	Source or destination burst transfer request size
0b000	1
0b001	4
0b010	8
0b011	16
0b100	32
0b101	64
0b110	128
0b111	256

Table below lists the value of the SWidth or DWidth bits and their corresponding widths

Bit value of SWidth or DWidth	Source or destination width
0b000	Byte, 8-bit
0b001	Halfword, 16-bit
0b010	Word, 32-bit
0b011	Reserved
0b100	Reserved
0b101	Reserved
0b110	Reserved
0b111	Reserved

Protection and access information

AHB access information is provided to the source and destination peripherals when a transfer occurs. The transfer information is provided by programming the DMA channel, the Prot bit of the DMACCxControl Register, and the Lock bit of the DMACCxConfiguration Register. Software programs these bits, and peripherals can use this information if necessary. Three bits of information are provided. Table below lists the purposes of the three protection bits.

Bits	Description	Purpose
[0]	Privileged or User	Indicates whether the access is in User, or Privileged mode: 0 = user mode 1 = privileged mode. This bit controls the AHB **HPROT[1]** signal.
[1]	Bufferable or Nonbufferable	Indicates whether or not the access can be buffered: 0 = non-bufferable 1 = bufferable. This bit indicates whether or not the access is bufferable. For example, you can use this bit to indicate to an AMBA bridge that the read can complete in zero wait states on the source bus without waiting for it to arbitrate for the destination bus and for the slave to accept the data. This bit controls the AHB **HPROT[2]** signal.
[2]	Cacheable or Noncacheable	Indicates whether or not the access can be cached: 0 = non-cacheable 1 = cacheable. This bit indicates whether or not the access is cacheable. For example, you can use this bit to indicate to an AMBA bridge that when it saw the first read of a burst of eight it can transfer the whole burst of eight reads on the destination bus, rather than pass the transactions through one at a time. This bit controls the AHB **HPROT[3]** signal.

Channel Configuration Registers

The eight DMACCxConfiguration Registers, with address offsets of 0x110, 0x130, 0x150, 0x170, 0x190, 0x1B0, 0x1D0, and 0x1F0 respectively, are read/write and configure the DMA channel. The registers are not updated when a new LLI is requested.

Figure below shows the bit assignments for these registers

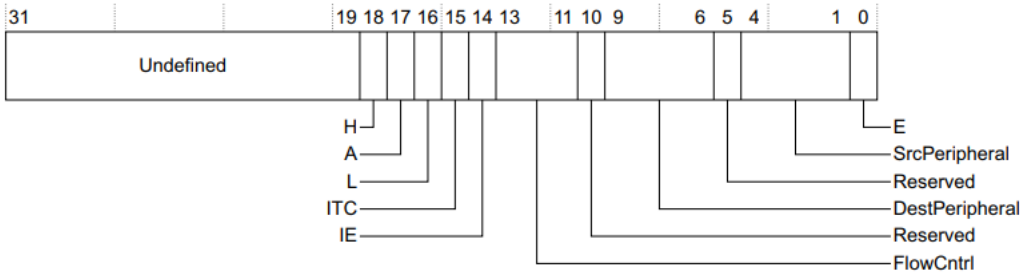


Table below lists the bit assignments for these registers

Bits	Name	Type	Function
[31:19]	-	-	Read undefined. Write as zero.
[18]	H	R/W	Halt: 0 = enable DMA requests 1 = ignore extra source DMA requests. The contents of the channels FIFO are drained. You can use this value with the Active and Channel Enable bits to cleanly disable a DMA channel.
[17]	A	RO	Active: 0 = there is no data in the FIFO of the channel 1 = the FIFO of the channel has data. You can use this value with the Halt and Channel Enable bits to cleanly disable a DMA channel.
[16]	L	R/W	Lock. When set, this bit enables locked transfers. For details of how lock control works, see *Lock control* on page 2-15.
[15]	ITC	R/W	Terminal count interrupt mask. When cleared, this bit masks out the terminal count interrupt of the relevant channel.

Bits	Name	Type	Function
[14]	IE	R/W	Interrupt error mask. When cleared, this bit masks out the error interrupt of the relevant channel.
[13:11]	FlowCntrl	R/W	Flow control and transfer type. This value indicates the flow controller and transfer type. The flow controller can be the DMAC, the source peripheral, or the destination peripheral. The transfer type can be memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral.
[10]	-	-	Read undefined. Write as zero.
[9:6]	DestPeripheral[a]	R/W	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory.
[5]	-	-	Read undefined. Write as zero.
[4:1]	SrcPeripheral[a]	R/W	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory.
[0]	E	R/W	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled 1 = channel enabled. You can also determine the Channel Enable bit status by reading the DMACEnbldChns register. You enable a channel by setting this bit. You can disable a channel by clearing the Enable bit. This causes the current AHB transfer, if one is in progress, to complete, and the channel is then disabled. Any data in the channel's FIFO is lost. Restarting the channel by setting the Channel Enable bit has unpredictable effects and you must fully re-initialize the channel. The channel is also disabled, and the Channel Enable bit cleared, when the last LLI is reached, or if a channel error is encountered. If a channel has to be disabled without losing data in a channel's FIFO, you must set the Halt bit so that subsequent DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating that there is no data left in the channel's FIFO. Finally, you can clear the Channel Enable bit.

Table below lists the bit values of the three flow control and transfer type bits.

Bit value	Transfer type	Controller
000	Memory-to-memory	DMA
001	Memory-to-peripheral	DMA
010	Peripheral-to-memory	DMA
011	Source peripheral-to-destination peripheral	DMA
100	Source peripheral-to-destination peripheral	Destination peripheral
101	Memory-to-peripheral	Peripheral
110	Peripheral-to-memory	Peripheral
111	Source peripheral-to-destination peripheral	Source peripheral

11.3.4.16 Peripheral Identification Registers 0-3

The DMACPeriphID0-3 Registers are four 8-bit registers, that span address locations 0xFE0-0xFEC. You can treat the registers conceptually as a 32-bit register. These read-only registers provide the following peripheral options:

PartNumber[11:0]

This identifies the peripheral. The three digit product code 0x080 is used.

Designer ID[19:12]

This is the identification of the designer. (ASCII A).

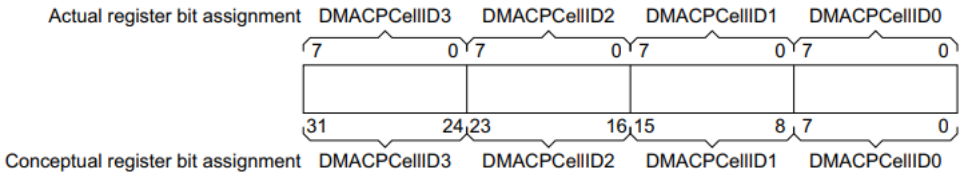
Revision[23:20]

This is the revision number of the peripheral. The revision number starts from 0.

Configuration[31:24]

This is the configuration option of the peripheral.

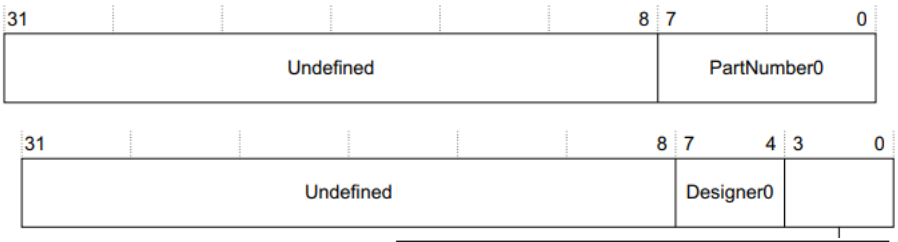
Figure below shows the bit assignments for these registers



DMACPeriphID0 Register

The read-only DMACPeriphID0 Register, with address offset of 0xFE0, is hard-coded and the fields in the register determine the reset value. Figure 3-19 shows the register bit assignments.

DMACPeriphID1 Register



Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	Designer0	These bits read back as 0x1
[3:0]	PartNumber1	These bits read back as 0x0

The read-only DMACPeriphID1 Register, with address offset of 0xFE4, is hard-coded and the fields in the register determine the reset value. Figure 3-20 shows the register bit assignments.

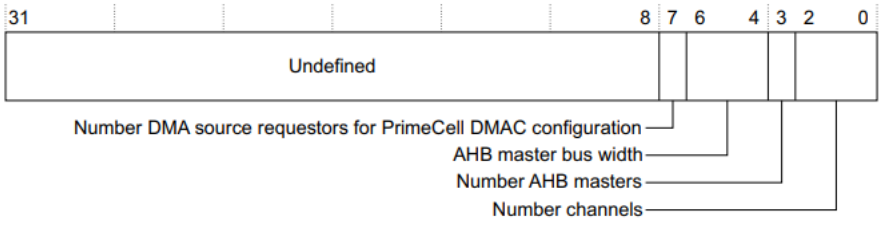
DMACPeriphID2 Register

The read-only DMACPeriphID2 Register, with address offset of 0xFE8, is hard-coded and the fields within the register determine the reset value. Figure 3-21 shows the register bit assignments.

Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	Revision	These bits read back as 0x1
[3:0]	Designer1	These bits read back as 0x4

DMACPeriphID3 Register

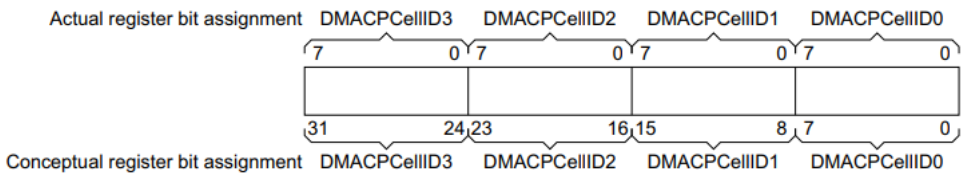
The read-only DMACPeriphID3 Register, with address offset of 0xFEC, is hard-coded and the fields in the register determine the reset value. Figure 3-22 shows the register bit assignments.



Bits	Name	Description
[31:8]	-	Read undefined.
[7]	Configuration	Indicates the number of DMA source requestors for the DMAC configuration: 0 = 16 DMA requestors 1 = 32 DMA requestors. This peripheral is set to 0.
[6:4]	Configuration	Indicates the AHB master bus width: 000 = 32-bit wide 001 = 64-bit wide 010 = 128-bit wide 011 = 256-bit wide 100 = 512-bit wide 101 = 1024-bit wide. This peripheral is set to 000.
[3]	Configuration	Indicates the number of AHB masters: 0 = one AHB master interface 1 = two AHB master interfaces. This peripheral is set to 1.
[2:0]	Configuration	Indicates the number of channels: 000 = 2 channels 001 = 4 channels 010 = 8 channels 011 = 16 channels 100 = 32 channels. This peripheral is set to 010.

11.3.4.17 PrimeCell Identification Registers 0-3

The DMACPCellID0-3 Registers are four 8-bit wide read-only registers that span address locations 0xFF0-0xFFC. You can treat the registers conceptually as a 32-bit register. The register is a standard cross-peripheral identification system. The DMACPCellID Register is set to 0xB105F00D. Figure below shows the bit assignments for these registers.



DMACPCellID0 Register

The read-only DMACPCellID0 Register, with address offset of 0xFF0, is hard-coded and the fields in the register determine the reset value. Figure below shows the register bit assignments.

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	DMACPCellID0	These bits read back as 0x0D

DMACPCellID1 Register

The read-only DMACPCellID1 Register, with address offset of 0xFF4, is hard-coded and the fields within the register determine the reset value. Figure below shows the register bit assignments.

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	DMACPCellID1	These bits read back as 0xF0

DMACPCellID2 Register

The read-only DMACPCellID2 Register, with address offset of 0xFF8, is hard-coded and the fields in the register determine the reset value. Figure below shows the register bit assignments.

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	DMACPCellID2	These bits read back as 0x05

DMACPCellID3 Register

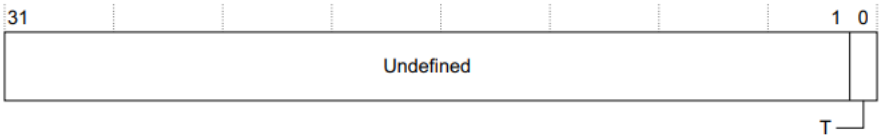
The read-only DMACPCellID3 Register, with address offset of 0xFFC, is hard-coded and the fields in the register determine the reset value. Figure below shows the register bit assignments.

Bits	Name	Description
[31:8]	-	Read undefined
[7:0]	DMACPCellID3	These bits read back as 0xB1

11.3.5 Test registers

11.3.5.1 Test Control Register

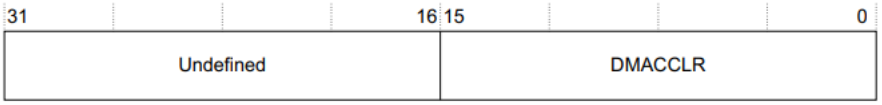
The read/write DMACITCR Register, with address offset of 0x500, is a 16-bit register that selects the various test modes and is cleared on reset. This register enables you to test the DMAC using TIC block-level tests and Built-In Self-Test (BIST) integration and system level tests. Figure below shows the register bit assignments.



Bits	Name	Description
[31:1]	-	Read undefined. Write as zero.
[0]	T	Test mode enable. Multiplex the test registers to control the input and output lines: 0 = normal operation 1 = test registers multiplexed onto input and outputs.

11.3.5.2 Integration Test Output Register 1

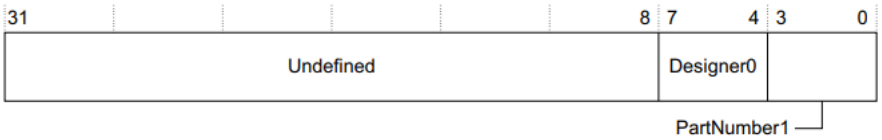
The read/write DMACITOP1 Register, with address offset of 0x504, is a 16-bit register that controls and reads the DMACCLR[15:0] output lines in test mode. Figure 4-2 shows the register bit assignments.



Bits	Name	Description
[31:16]	-	Read undefined. Write as zero.
[15:0]	DMACCLR	You can set the **DMACCLR[15:0]** response outputs to a certain value in test mode by writing to the register. A read returns the value on the outputs, after the test multiplexor.

11.3.5.3 Integration Test Output Register 2

The read/write DMACITOP2 Register, with address offset of 0x508, is a 16-bit register that controls and reads the DMACTC[15:0] output lines in test mode. Figure 4-3 shows the register bit assignments.



Bits	Name	Description
[31:8]	-	Read undefined
[7:4]	Designer0	These bits read back as 0x1
[3:0]	PartNumber1	These bits read back as 0x0

Bits	Name	Description
[31:2]	-	Read undefined. Write as zero.
[1]	E	You can set the **DMACINTERR** interrupt request to a certain value in test mode by writing to the register. A read returns the value on the output, after the test multiplexor.
[0]	TC	You can set the **DMACINTTC** interrupt request to a certain value in test mode by writing to the register. A read returns the value on the output, after the test multiplexor.

11.3.5.4 Integration Test Output Register 3

The read/write DMACITOP3 Register, with address offset of 0x50C, is a 16-bit register that controls and reads the interrupt request output lines in test mode. Figure 4-4 shows the register bit assignments

Bits	Name	Description
[31:2]	-	Read undefined. Write as zero.
[1]	E	You can set the **DMACINTERR** interrupt request to a certain value in test mode by writing to the register. A read returns the value on the output, after the test multiplexor.
[0]	TC	You can set the **DMACINTTC** interrupt request to a certain value in test mode by writing to the register. A read returns the value on the output, after the test multiplexor.

12 Analog-to-digital converter (ADC)

12.1 Overview

3×12-bit, 1.0 MSPS A/D converters are embedded and each ADC has up to 17 multiplexed channels allowing it measure signals from sixteen external and one internal sources,and 3 MSPS in triple interleaved mode

A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode.

The analog watchdog feature allows the application to detect if the input voltage goes outside the user-defined high or low thresholds.

The ADC input clock is generated from the inter-connection logic clock.

Characteristics

(1) ADC sampling rate: 1 MSPS for 12-bit resolution

(2) Programmable sampling time

(3) 16 external analog inputs and 1 channel for internal temperature sensor

(4) Converts a single channel or scans a sequence of channels

(5) Single mode converts selected inputs once per trigger

(6) Continuous mode converts selected inputs continuously

(7) Discontinuous mode

(8) Analog watchdog

(9) ADC supply requirements: 3.0V to 3.6V, and typical power supply voltage is 3.3V

(10) ADC input range: VSSA ≤VIN ≤VREFP

12.2 Pins and internal signals

ADC internal signals

Internal signal name	Signal type	Description
Vtemp-sense(ADC1)	input	Internal temperature sensor output voltage

ADC pins definition

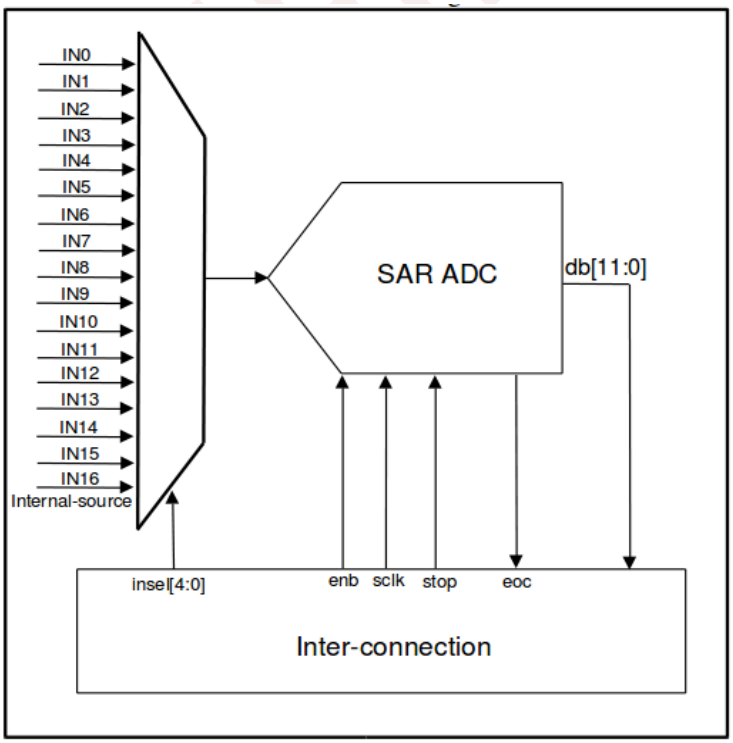
Name	Signal type	Description
VDDA	Analog power supply	Analog power supply equal to VDD33 and 3.0 V ≤ VDDA≤ 3.6 V
VSSA	Analog power ground	Ground for analog power supply equal to VSS33
VREFP	Analog reference positive	The positive reference voltage for the ADC, 3.0 V ≤ VREFP ≤ VDDA
IN[15:0]	Input, Analog signals	Up to 16 external channels

12.3 Temperature sensor

The temperature sensor can be used to measure the ambient temperature of the device. The sensor output voltage can be converted into a digital value by ADC. The sampling time for the temperature sensor is recommended to be set to at least 10μs.

The output voltage of the temperature sensor changes linearly with temperature. Because there is an offset, varies from chip to chip due to process variation, the internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperature.

12.4 ADC block pins



ADC block diagram

ADC block pins	Descriptions
in0~in16	Analog input channel
insel1[4:0]	input channel selection 00001: in0 00010: in1 00011: in2 00100: in3 00101: in4 00110: in5 00111: in6 01000: in7 01001: in8 01010: in9 01011: in10 01100: in11 01101: in12 01110: in13 01111: in14 10000: in15 10001: in16(internal source) for ADC1, inter source is Vtemp-sense
enb	Used to enable adc 0: adc enable 1: adc disable
stop	Stop mode enable 0: disable 1: enable, when stop enable, power down ADC block.
db[11:0]	adc 12 bits output data
eoc	adc end of conversion flag. when rising edge, adc output data will ready

12.5 ADC input signals vs package pins

ADC IP input pins	LQFP64
IN0	PIN14(WKUP_ADC_IN0_CMP_PA0)
IN1	PIN15 (ADC_IN1_CMP_PA1)
IN2	PIN16(ADC_IN2_CMP_PA2)
IN3	PIN17(ADC_IN3_CMP_PA3)
IN4	PIN20(ADC_IN4_CMP_PA4_DAC0)
IN5	PIN21(ADC_IN5_CMP_PA5_DAC1)
IN6	PIN22(ADC_IN6)
IN7	PIN23(ADC_IN7)
IN8	PIN26(ADC_IN8)
IN9	PIN27(ADC_IN9)
IN10	PIN8(ADC_IN10)
IN11	PIN9(ADC_IN11)
IN12	PIN10(ADC_IN12)
IN13	PIN11(ADC_IN13)
IN14	PIN24(ADC_IN14)
IN15	PIN25(ADC_IN15)

ADC IP input pins	LQFP100
IN0	PIN23(WKUP_ADC_IN0_CMP_PA0)
IN1	PIN24 (ADC_IN1_CMP_PA1)
IN2	PIN25(ADC_IN2_CMP_PA2)
IN3	PIN26(ADC_IN3_CMP_PA3)
IN4	PIN29(ADC_IN4_CMP_PA4_DAC0)
IN5	PIN30(ADC_IN5_CMP_PA5_DAC1)
IN6	PIN31(ADC_IN6)
IN7	PIN32(ADC_IN7)

IN8	PIN35(ADC_IN8)
IN9	PIN36(ADC_IN9)
IN10	PIN15(ADC_IN10)
IN11	PIN16(ADC_IN11)
IN12	PIN17(ADC_IN12)
IN13	PIN18(ADC_IN13)
IN14	PIN33(ADC_IN14)
IN15	PIN34(ADC_IN15)

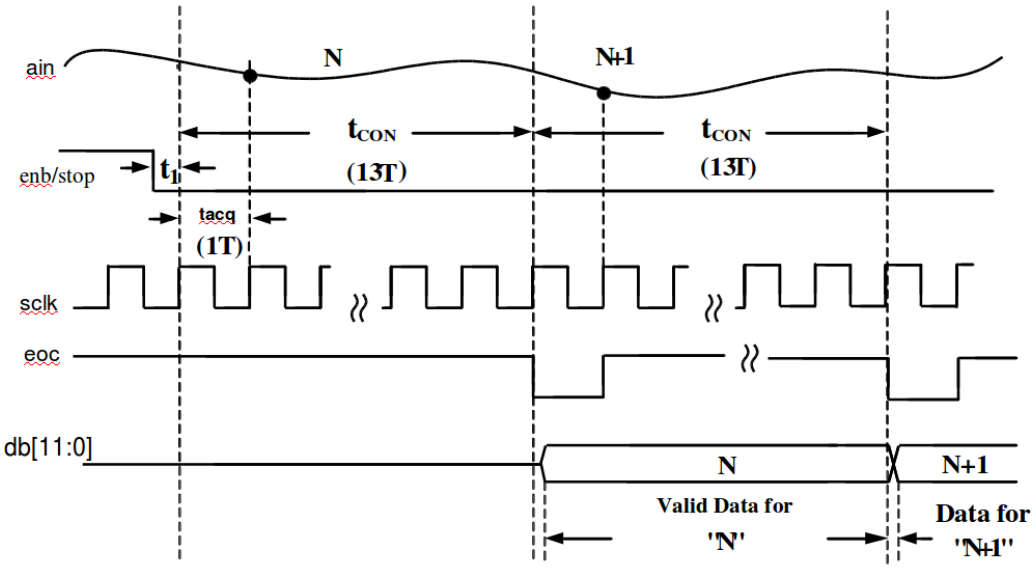
12.6 ADC characteristics

Symbol	Parameter	MIN	TYP	MAX	Unit
VDDA	Operating voltage	3.0	3.3	3.6	V
VIN	ADC input voltage range	0	-----	VREFP	V
fADC	ADC clock	0.5	-----	14	MHz
fs	Sampling rate	-----	-----	1	MHz
tconv	ADC conversion time	1	-----	20	μs
RADC	Input sampling switch resistance	-----	-----	2	kΩ
CADC	Input sampling capacitance	-----	10	-----	pF
tsu	Startup time	-----	-----	10	μs

12.7 ADC timing diagram

After the start of ADC conversion and after 13 clock cycles, the EOC flag is set and the 12-bit

Data is ready.



Parameter	Limit	Unit	Description
sclk	14	MHz	System Clock Frequency
tcon	13	Cycles	Time for a data conversion
tACQ	1	Cycles	Time for signal acquisition
t1	10	ns	ENB to SCLK Setup Time

13 Digital-to-analog converter (DAC)

13.1 Overview

The Digital-to-analog converter converts 10-bit digital data to a voltage on the external pins. The output voltage can be optionally buffered for higher drive capability. The two DACs can work independently or concurrently.

DAC main features

(1) Two DAC converters: one output channel each

(2) Conversion triggered by external triggers

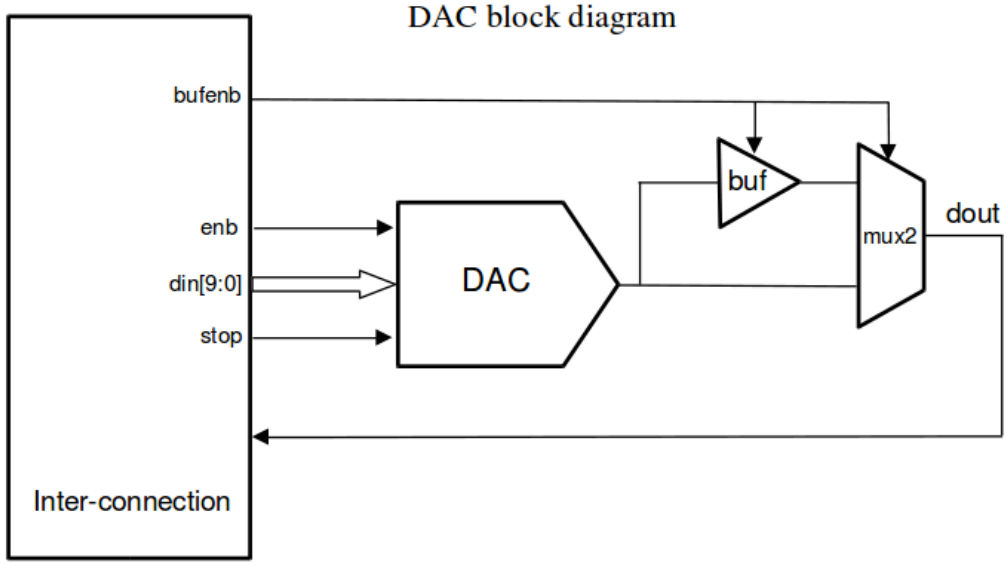
(3) Dual DAC channel independent or simultaneous conversions

(4) Configurable internal buffer

(5) External triggers for conversion

(6) Input voltage reference VREFP

DAC block diagram



13.2 DAC block pins

DAC block pins	Descriptions
enb	Used to enable dac 0: dac enable 1: dac disable
bufenb	Used to enable output buffer 0: buffer enable 1: buffer disable
stop	Stop mode enable 0: disable 1: enable, when stop enable, power down DAC blocks
din[9:0]	10 bits dac input data
dout	Analog dac output signal to IOs

13.3 DAC pins

Name	Signal type	Remarks
VDDA	Analog power supply	Analog power supply equal to VDD and 3.0 V ≤ VDDA≤ 3.6 V
VSSA	Analog power ground	Ground for analog power supply equal to VSS33
VREFP	Analog reference positive	The positive reference voltage for the ADC, 3.0 V ≤ VREFP ≤ VDDA
DOUT	DAC analog output	Analog output signal

13.4 DACs output signals vs package pins

DAC output pins	qfp100
dout_dac0	PIN_29(ADC_IN4_CMP_PA4_DAC0)
dout_dac1	PIN_30(ADC_IN5_CMP_PA5_DAC1)
DAC output pins	qfp64
dout_dac0	PIN_20(ADC_IN4_CMP_PA4_DAC0)
dout_dac1	PIN_21(ADC_IN5_CMP_PA5_DAC1)

13.5 DAC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDDA	Operating voltage		3.0	3.3	3.6	V
VREFP	Reference supply voltage	VREFP should always below VDDA	3.0	3.3	3.3	V
RLOAD	Load resistance	Resistive load vs. VSSA with buffer ON	5			kΩ
CLOAD	Load capacitance	No pin/pad capacitance included			50	pF
DAC_OUTmin	Lower DAC_OUT voltage with buffer ON					
DAC_OUTmax	Lower DAC_OUT voltage with buffer ON				VDDA-0.2	V
Update rate	Max frequency for a correct DAC_OUT change from code i to i±1LSBs	CLOAD≤50pF, RLOAD≥5kΩ			2	MS/s

13.6 DAC output voltage

The analog output voltage on the DAC pin is determined by the following equation:

$DACoutput = V_{REFP} * DAC\_Dout/1024$

The digital input is linearly converted to an analog output voltage, its range is 0 to V_{REFP}。

14 Comparator (CMP)

14.1 Overview

The general purpose comparators, CMP0 and CMP1, can work either standalone or together with the timers.
It could be used to wake up the MCU from low-power mode by an analog signal, provide a trigger source when an analog signal is in a certain condition, achieves some current control by working together with a PWM output of a timer and the DAC.

14.2 Characteristic

(1) Rail-to-rail comparators

(2) Configurable hysteresis

(3) Configurable speed and consumption

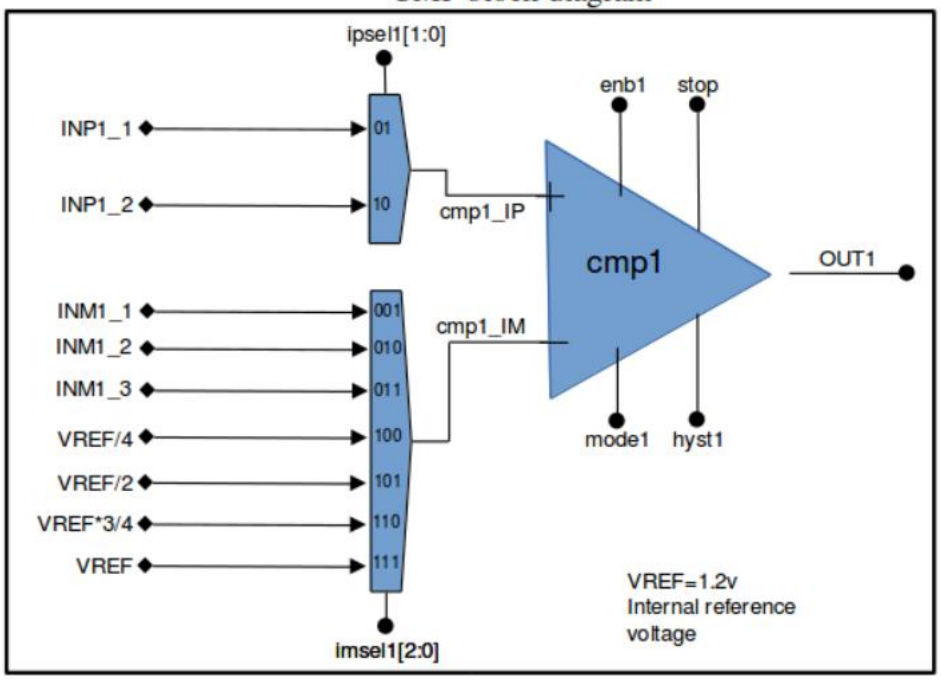
(4) Each comparator has configurable analog input source

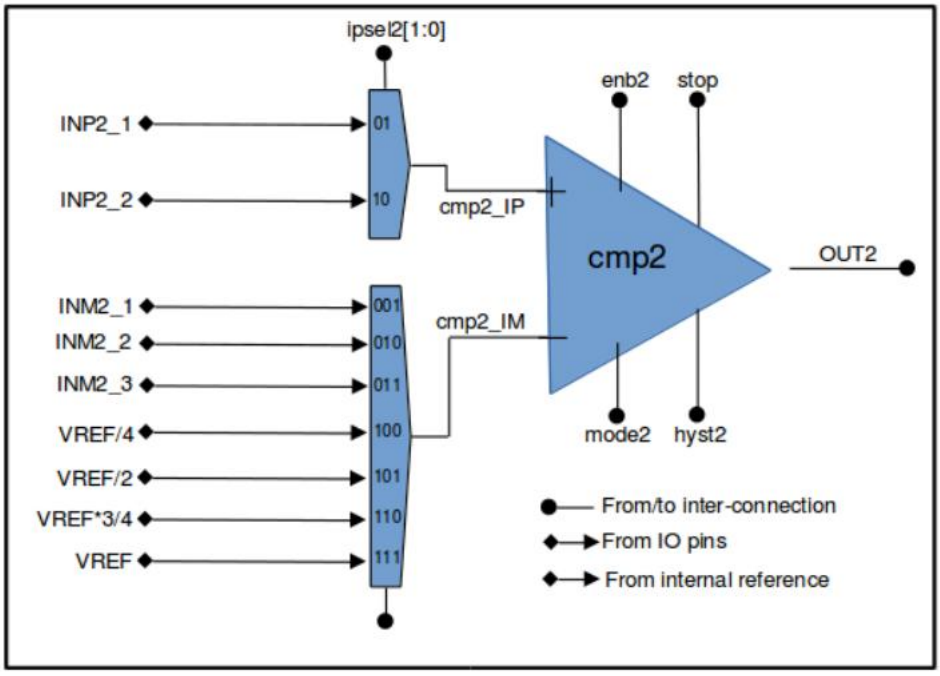
(5) The whole or sub-multiple values of internal reference voltage Window comparator

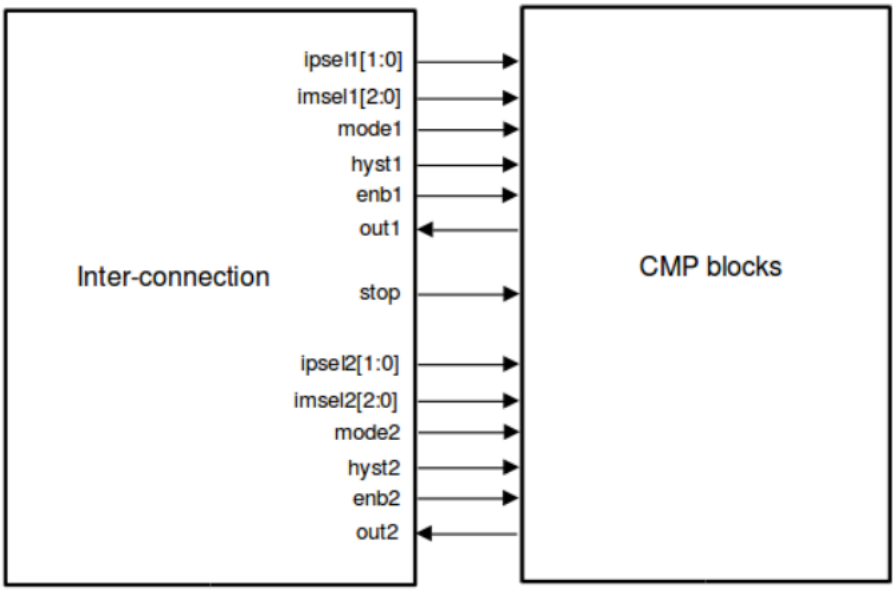
(6) Outputs to I/O

(7) Outputs to timers for triggering

CMP block diagram







14.3 CMP block pins

CMP block pins	Descriptions
INP1_1 INP1_2	Non-inverting input signals of cmp1
ipsel1[1:0]	cmp1_IP input selection 01: INP1_1 10: INP1_2
INM1_1 INM1_2 INM1_3	Inverting input signals of cmp1
imsel1[2:0]	cmp1_IM input selection 001: INM1_1 010: INM1_2 011: INM1_3 100: ¼ VREF 101: ½ VREF 110: ¾ VREF 111: VREF

mode1	Control the operating mode of the cmp1 adjust the speed /consumption. 0: High speed / full power 1: Low speed / low power
hyst1	Used to set the hysteresis of cmp1 0: no hysteresis 1: have hysteresis
enb1	Used to enable cmp1 0: cmp1 enable 1: cmp1 disable
out1	cmp1 output
INP2_1 INP2_2	Non-inverting input signals of cmp2
ipsel2[1:0]	cmp2_IP input selection 01: INP2_1 10: INP2_2
INM2_1 INM2_2 INM2_3	Inverting input signals of cmp2
imsel2[2:0]	cmp2_IM input selection 001: INM2_1 010: INM2_2 011: INM2_3 100: ¼ VREF 101: ½ VREF 110: ¾ VREF 111: VREF
mode2	Control the operating mode of the cmp2 adjust the speed /consumption. 0: High speed / full power 1: Low speed / low power
hyst2	Used to set the hysteresis of cmp2 0: no hysteresis 1: have hysteresis
enb2	Used to enable cmp2 0: cmp2 enable 1: cmp2 disable
out2	cmp2 output
stop	Stop mode enable 0: disable 1: enable, when stop enable, power down CMP block.

14.4 CMP input signals vs package pins

CMP IP input pins	Lqfp64
INP1_1	PIN_15 (ADC_IN1_CMP_PA1)
INP1_2	PIN_20(ADC_IN4_CMP_PA4_DAC0)
INM1_1	PIN_20(ADC_IN4_CMP_PA4_DAC0)
INM1_2	PIN_14(WKUP_ADC_IN0_CMP_PA0)
INM1_3	PIN_21(ADC_IN5_CMP_PA5_DAC1)
INP2_1	PIN_15 (ADC_IN1_CMP_PA1)
INP2_2	PIN_17(ADC_IN3_CMP_PA3)
INM2_1	PIN_20(ADC_IN4_CMP_PA4_DAC0)
INM2_2	PIN_16(ADC_IN2_CMP_PA2)
INM2_3	PIN_21(ADC_IN5_CMP_PA5_DAC1)
CMP IP input pins	Lqfp100
INP1_1	PIN_24 (ADC_IN1_CMP_PA1)
INP1_2	PIN_29(ADC_IN4_CMP_PA4_DAC0)
INM1_1	PIN_29(ADC_IN4_CMP_PA4_DAC0)
INM1_2	PIN_23(WKUP_ADC_IN0_CMP_PA0)
INM1_3	PIN_30(ADC_IN5_CMP_PA5_DAC1)
INP2_1	PIN_24 (ADC_IN1_CMP_PA1)
INP2_2	PIN_26(ADC_IN3_CMP_PA3)
INM2_1	PIN_29(ADC_IN4_CMP_PA4_DAC0)
INM2_2	PIN_25(ADC_IN2_CMP_PA2)
INM2_3	PIN_30(ADC_IN5_CMP_PA5_DAC1)

14.5 Comparator characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDDA	Analog supply voltage		3.0	3.3	3.3	V
VIN	Comparator input voltage range		0		VDDA	V
tstart	Comparator startup time	VDDA ≥ 3.0 V			10	us
tD	Propagation delay for full range step with 100 mV overdrive	VDDA ≥ 3.0 V			50	ns
VOFFSET	Comparator offset error	VDDA ≥ 3.0 V			±30	mV

15 Backup registers (BKP)

BKP introduction

The backup registers are sixteen two 16-bit registers for storing 32 bytes of user application data. They are implemented in the backup domain that remains powered on by VBAT when the VDD33 power is switched off. They are not reset when the device wakes up from Standby mode or by a system reset or power reset.

In addition, the BKP control registers are used to manage the RTC calibration.

After reset, access to the Backup registers and RTC is disabled and the Backup domain (BKP) is protected against possible parasitic write access.

To enable access to the Backup registers and the RTC, proceed as follows:

(1) enable the power and backup interface clocks by setting the PWREN and BKPEN bits in the RCC_APB1ENR register

(2) set the DBP bit the Power Control Register (PWR_CR) to enable access to the Backup registers and RTC.

BKP main features

(1) 32-byte data registers

(2) Calibration register for storing the RTC calibration value

(3) Possibility to output the RTC Calibration Clock, RTC Alarm pulse or Second pulse.

RTC calibration

For measurement purposes, the RTC clock with a frequency divided by 64 can be output on the pin. The clock can be slowed down by up to 121 ppm by configuring CAL[6:0] bits.

BKP registers

Backup data register x (BKP_DRx) (x = 1 ..17)

Address offset:

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D<15:0>															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0: D[15:0] Backup data

These bits can be written with user data.

Note: The BKP_DRx registers are not reset by a System reset or Power reset or when the device wakes up from Standby mode.

They are reset by a Backup Domain reset.

RTC clock calibration register (BKP_RTCCR)

Address offset:

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
						ASOS	ASOE	COO	CAL<6:0>						
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 9 ASOS: Alarm or second output selection

When the ASOE bit is set, the ASOS bit can be used to select whether the signal output on the PC13 pin is the RTC Second pulse signal or the Alarm pulse signal:

0: RTC Alarm pulse output selected

1: RTC Second pulse output selected

Note: This bit is reset only by a Backup domain reset.

Bit 8 ASOE: Alarm or second output enable

Setting this bit outputs either the RTC Alarm pulse signal or the Second pulse signal on the PC13 pin depending on the ASOS bit.

0: output disable

1: output enable

The output pulse duration is one RTC clock period. The PC13 pin must not be enabled while the ASOE bit is set.

Note: This bit is reset only by a Backup domain reset.

Bit 7 CCO: Calibration clock output

0: No effect

1: Setting this bit outputs the RTC clock with a frequency divided by 64 on the PC13 pin.

Note: This bit is reset when the VDD supply is powered off.

Bit 6:0 CAL[6:0]: Calibration value

This value indicates the number of clock pulses that will be ignored every 2^20 clock pulses.

This allows the calibration of the RTC, slowing down the clock by steps of 1000000/2^20PPM.

The clock of the RTC can be slowed down from 0 to 121PPM.

16 CRC(Cyclic redundancy check calculation unit)

16.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the Flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

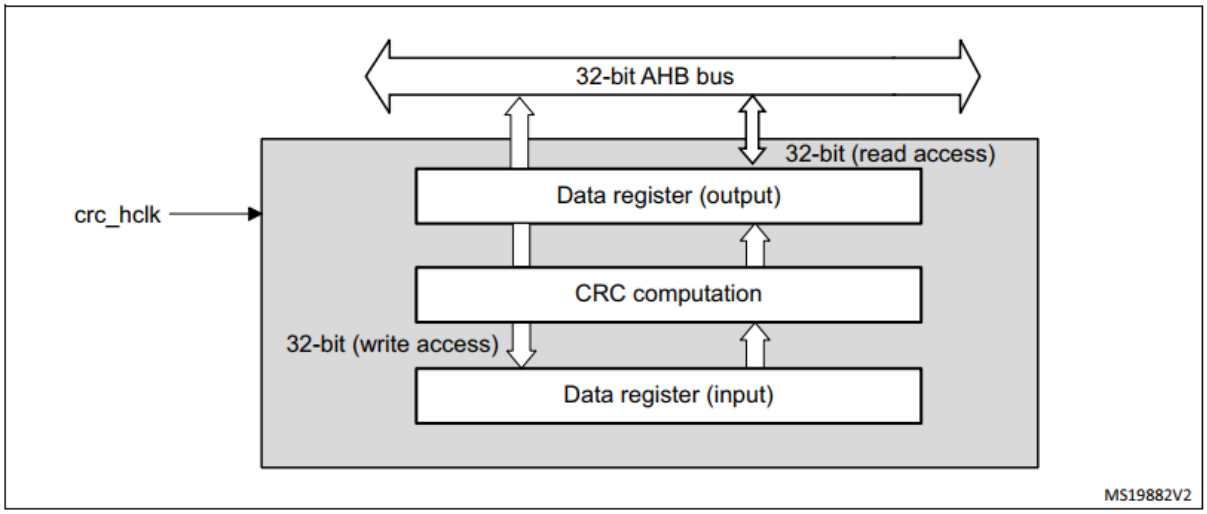
16.2 CRC main features

- Fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data

16.3 CRC functional description

16.3.1 CRC block diagram

Figure 7. CRC calculation unit block diagram



16.3.2 CRC internal signals

Table 15. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

16.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit access is allowed.

The duration of the computation depends on data width:

- 4 AHB clock cycles for 32-bit
- 2 AHB clock cycles for 16-bit
- 1 AHB clock cycles for 8-bit

An input buffer allows to immediately write a second data without waiting for any wait states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed, to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example: input data 0x1A2B3C4D is used for CRC calculation as:

0x58D43CB2 with bit-reversal done by byte

0xD458B23C with bit-reversal done by half-word

0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level: for example, output data 0x11223344 is converted into 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the

POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11D

16.4 CRC registers

16.4.1 Data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 DR[31:0]: Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

16.4.2 Independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
								rw							

Bits 31:8 Reserved, must be kept cleared.

Bits 7:0 IDR[7:0]: General-purpose 8-bit data register bits

These bits can be used as a temporary storage location for one byte.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

16.4.3 Control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
								rw	rw	rw	rw	rw			rs

Bits 31:8 Reserved, must be kept cleared.

Bit 7 **REV_OUT**: Reverse output data

This bit controls the reversal of the bit order of the output data. 0: Bit order not affected

1: Bit-reversed output format

Bits 6:5 **REV_IN[1:0]**: Reverse input data

These bits control the reversal of the bit order of the input data 00: Bit order not affected

01: Bit reversal done by byte

10: Bit reversal done by half-word 11: Bit reversal done by word

Bits 4:3 POLYSIZE[1:0]: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept cleared.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware

16.4.4 Initial CRC value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT***: Programmable initial CRC value*

This register is used to write the CRC initial value.

16.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C11DB7

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

Bits 31:0 **POL[31:0]***:* Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

16.4.6 CRC register map

Table 16. CRC register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REV_OUT	REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																									0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	Polynomial coefficients																															
	Reset value	0x04C11DB7																															

17 General-purpose input/outputs (GPIOs)

17.1 Overview

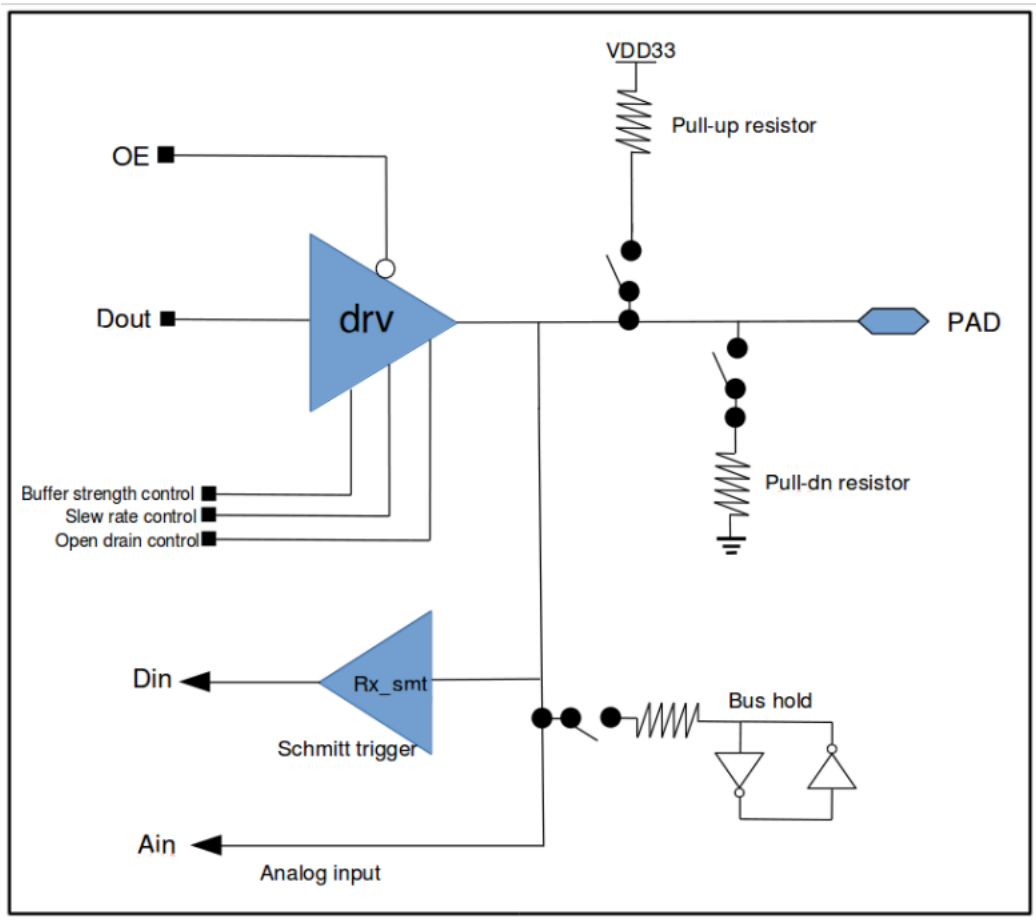
AG32 device provides up to 78 user I/O ports(GPIOs).

Each of the GPIO pins can be configured by software as output (push-pull or open-drain, with or without pull-up or pull-down), as input (floating, with or without pull-up or pull-down) or as peripheral alternate function. Most of the GPIO pins are shared with digital or analog alternate functions. All GPIOs are high-current-capable and have speed selection to better manage internal noise, power consumption and electromagnetic emission.

The I/O configuration can be locked if needed by following a specific sequence in order to avoid spurious writing to the I/Os registers.

17.2 Functional description

Basic structure of a user I/O



User IOs offers a range of programmable features for an I/O pin. These features increase the flexibility of I/O utilization and provide a way to reduce the usage of external discrete components, such as pull-up resistors.

Programmable Current Strength

The output buffer for each I/O pin has a programmable current strength control for certain I/O standards. The LVTTL, LVCMOS standards have several levels of current strength that you can control.

Slew Rate Control

The output buffer for each I/O pin provides optional programmable output slew-rate control. However, these fast transitions may introduce noise transients in the system. A slower slew rate reduces system noise, but adds a nominal delay to rising and falling edges. Because each I/O pin has an individual slew-rate control, you can specify the slew rate on a pin-by-pin basis. The slew-rate control affects both the rising and falling edges.

Open-Drain Output

Each I/O pin provide an optional open-drain (equivalent to an open-collector) output. This open-drain output enables the device to provide system-level control signals (for example, interrupt and write enable signals) that are asserted by multiple devices in your system.

Bus Hold

Each user I/O pin provides an optional bus-hold feature. The bus-hold circuitry holds the signal on an I/O pin at its last-driven state. Because the bus-hold feature holds the last-driven state of the pin until the next input signal is present, an external pull-up or pull-down resistor is not necessary to hold a signal level when the bus is tri-stated.

Programmable Pull-Up Resistor

Each I/O pin provides an optional programmable pull-up resistor while in user mode. If you enable this feature for an I/O pin, the pull-up resistor holds the output to the VDD level.

During and just after reset, the user IOs are configured in Input Floating mode.

The JTAG pins are in input PU/PD after reset:

- JTDI in PU
- JTCK in PD
- JTMS in PU
- NJTRST in PU

17.3 Register descriptions

17.3.1 Data register, GPIODATA

The GPIODATA register is the data register. In software control mode, values written in the GPIODATA register are transferred onto the GPOUT pins if the respective pins have been configured as outputs through the GPIODIR register.
In order to write to GPIODATA, the corresponding bits in the mask, resulting from the address bus, PADDR[9:2], must be HIGH. Otherwise the bit values remain unchanged by the write.
Similarly, the values read from this register are determined for each bit, by the mask bit derived from the address used to access the data register, PADDR[9:2]. Bits that are 1 in the address mask cause the corresponding bits in GPIODATA to be read, and bits that are 0 in the address mask cause the corresponding bits in GPIODATA to be read as 0, regardless of their value.
A read from GPIODATA returns the last bit value written if the respective pins are configured as output, or it returns the value on the corresponding input GPIN bit when these are configured as inputs. All bits are cleared by a reset.
Table below shows the bit assignment of the GPIODATA register.

Bits	Name	Type	Function
7:0	Data register	Read/ write	Input data Output data

17.3.2 Data direction register, GPIODIR

The GPIODIR register is the data direction register. Bits set to HIGH in the GPIODIR configure corresponding pin to be an output. Clearing a bit configures the pin to be input. All bits are cleared by a reset. Therefore, the GPIO pins are input by default.
Table below shows the bit assignment of the GPIODIR register

Bits	Name	Type	Function
7:0	Data direction register	Read/ write	Bits set, pins output Bits cleared, pins output

17.3.3 Interrupt sense register, GPIOIS

The GPIOIS register is the interrupt sense register. Bits set to HIGH in GPIOIS configure the corresponding pins to detect levels. Clearing a bit configures the pin to detect edges. All bits are cleared by a reset.
Table below shows the bit assignment of the GPIOIS register.

Bits	Name	Type	Function
7:0	Interrupt sense register	Read/ write	Bits clear, edge on corresponding pin is detected Bits set, level on corresponding pin is detected

17.3.4 Interrupt both-edges register, GPIOIBE

The GPIOIBE register is the interrupt both-edges register. When the corresponding bit in GPIOIS is set to detect edges, bits set to HIGH in GPIOIBE configure the corresponding pin to detect both rising and falling edges, regardless of the corresponding bit in the GPIOIEV (interrupt event register). Clearing a bit configures the pin to be controlled by GPIOIEV. All bits are cleared by a reset.
Table below shows the bit assignment of the GPIOIBE register.

Bits	Name	Type	Function
7:0	Interrupt both edges	Read/write	Bits set, both edges on corresponding pin trigger an interrupt.Bits cleared, interrupt generation event is controlled by GPIOIEV. Single edge, determined by corresponding bit in GPIOIEV register.

17.3.5 Interrupt event register, GPIOIEV

The GPIOIEV register is the interrupt event register. Bits set to HIGH in GPIOIEV configure the corresponding pin to detect rising edges or high levels, depending on the corresponding bit value in GPIOIS. Clearing a bit configures the pin to detect falling edges or low levels, depending on the corresponding bit value in GPIOIS. All bits are cleared by a reset.
Table below shows the bit assignment of the GPIOIEV register.

Bits	Name	Type	Function
7:0	Interrupt event register	Read/write	Bits set, rising edges, or high levels on corresponding pins trigger interrupts. Bits cleared, falling edges, or low levels on corresponding pin trigger interrupts.

17.3.6 Interrupt mask register, GPIOIE

The GPIOIE register is the interrupt mask register. Bits set to HIGH in GPIOIE allow the corresponding pins to trigger their individual interrupts and the combined GPIOINTR line. Clearing a bit disables interrupt triggering on that pin. All bits are cleared by a reset.
Table below shows the bit assignment of the GPIOIE register.

Bits	Name	Type	Function
7:0	Interrupt mask register	Read/write	Bits set, corresponding pin is not masked. Bits cleared, corresponding pin interrupt is masked.

17.3.7 Raw interrupt status register, GPIORIS

The GPIORIS register is the raw interrupt status register. Bits read HIGH in GPIORIS reflect the status of interrupts trigger conditions detected (raw, prior to masking), indicating that all the requirements have been met, before they are finally allowed to trigger by GPIOIE. Bits read as zero indicate that corresponding input pins have not initiated an interrupt. This register is read only, and bits are cleared by a reset.
Table below shows the bit assignment of the GPIORIS register.

Bits	Name	Type	Function
7:0	Raw interrupt status	Read	Reflect the status of interrupts trigger conditions detection on pins (raw, prior to masking). Bits set, requirements met by corresponding pins.Bits clear, requirements not met.

17.3.8 Masked interrupt status register, GPIOMIS

The GPIOMIS register is the masked interrupt status register. Bits read HIGH in GPIOMIS reflect the status of input lines triggering an interrupt. Bits read as LOW indicate that either no interrupt has been

generated, or the interrupt is masked. GPIOMIS is the state of the interrupt after masking. This register is read-only, and all bits are cleared by a reset.

The contents of this register are made available externally through the intra-chip (or on-chip) GPIOMIS[7:0] signals.

Table below shows the bit assignment of the GPIOMIS register.

Bits	Name	Type	Function
7:0	Masked interrupt status	Read	Masked value of interrupt due to corresponding pin. Bits clear, PrimeCell GPIO line interrupt not active. Bits set, PrimeCell GPIO line asserting interrupt.

17.3.9 Interrupt clear register, GPIOIC

The GPIOIC register is the interrupt clear register. Writing a 1 to a bit in this register clears the corresponding interrupt edge detection logic register. Writing a 0 has no effect. This register is write-only and all bits are cleared by a reset.

Table below shows the bit assignment of the GPIOIC register.

Bits	Name	Type	Function
7:0	Interrupt clear register	Write	Bit written as 1, clears edge detection logic. Bit written as 0, has no effect.

17.3.10 Mode control select register, GPIOAFSEL

The GPIOAFSEL register is the mode control select register. Writing a 1 to any bit in this register selects the hardware control for the corresponding PrimeCell GPIO line. All bits are cleared by a reset, therefore no PrimeCell GPIO line is set to hardware control by default.

Table below shows the bit assignment of the GPIOAFSEL register.

Bits	Name	Type	Function
7:0	Mode control select register	Read/ write	Bit set, enables hardware control mode. Bit cleared, enables software control mode.

18 Universal asynchronous receiver transmitter (UART)

18.1 UART Introduction

The UART is an AMBA slave module that connects to the Advanced Peripheral Bus (APB).

The UART (UART0 - UART4) are used to translate data between parallel and serial interfaces, provides a flexible full duplex data exchange using asynchronous transfer. It is also commonly used for RS-232 standard communication.

The UART includes a programmable baud rate generator which is capable of dividing the system clock to produce a dedicated clock for the UART transmitter and receiver. The UART also supports DMA function for high speed data communication except UART4.

Programmable parameters

The following key parameters are programmable:

- communication baud rate, integer, and fractional parts
- number of data bits
- number of stop bits
- parity mode
- FIFO enable (16 deep) or disable (1 deep)
- FIFO trigger levels selectable between 1/8, 1/4, 1/2, 3/4, and 7/8.
- internal nominal 1.8432MHz clock frequency (1.42 – 2.12MHz) to generate low-power mode shorter bit duration
- hardware flow control.

Additional test registers and modes are implemented for integration testing.

18.2 UART functional description

The UART performs:

- serial-to-parallel conversion on data received from a peripheral device
- parallel-to-serial conversion on data transmitted to the peripheral device.

The CPU reads and writes data and control/status information through the AMBA APB interface. The transmit and receive paths are buffered with internal FIFO memories enabling up to 16-bytes to be stored independently in both transmit and receive modes.

The UART:

- includes a programmable baud rate generator that generates a common transmit and receive internal clock from the UART internal reference clock input, **UARTCLK**

- offers similar functionality to the industry-standard 16C550 UART device
- supports baud rates of up to 460.8Kbits/s, subject to **UARTCLK** reference clock frequency

The UART operation and baud rate values are controlled by the line control register (UARTLCR_H) and the baud rate divisor registers (UARTIBRD and UARTFBRD).

The UART can generate:

•individually-maskable interrupts from the receive (including timeout), transmit, modem status and error conditions

- a single combined interrupt so that the output is asserted if any of the individual interrupts are asserted, and unmasked
- DMA request signals for interfacing with a Direct Memory Access (DMA) controller

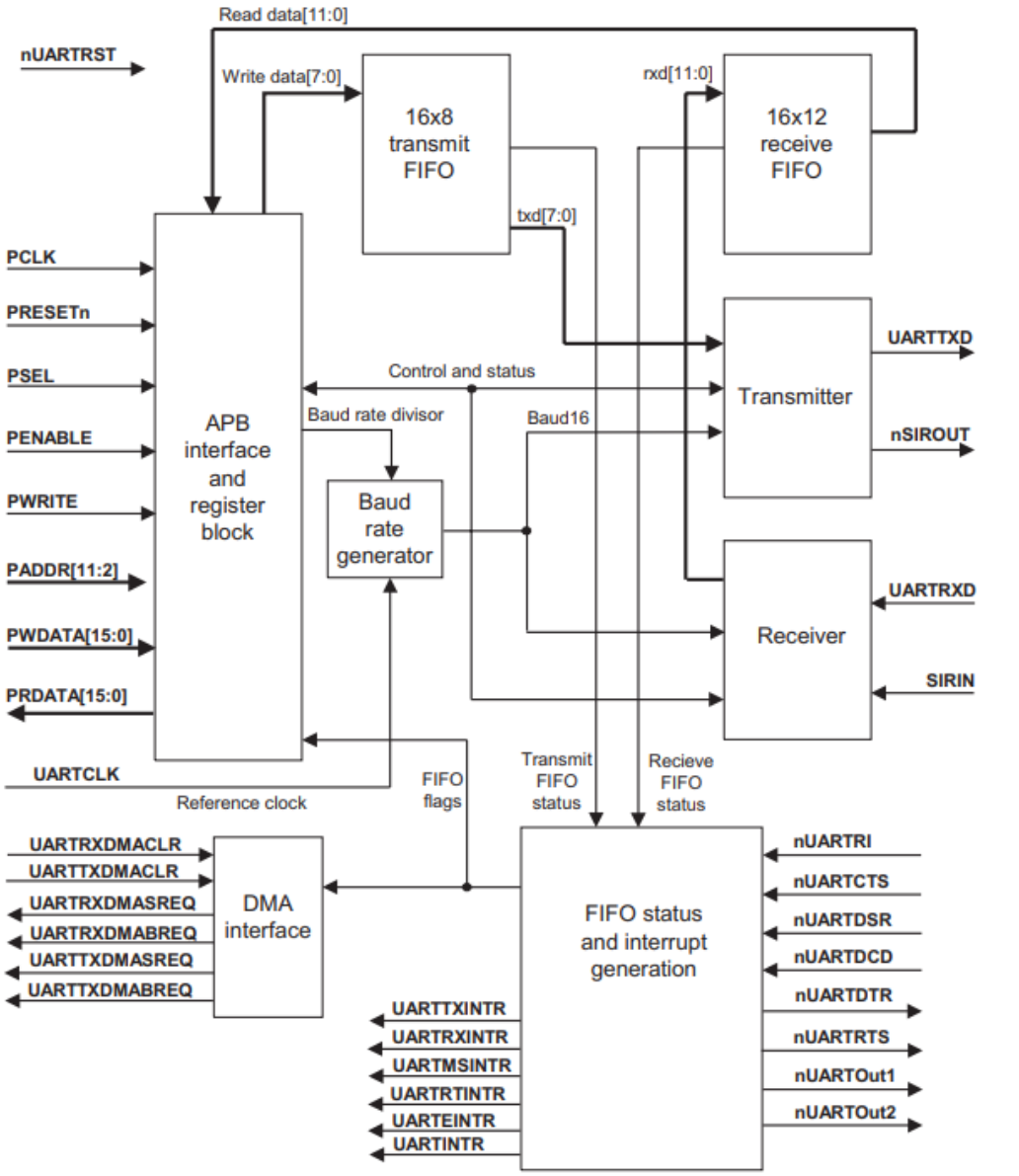
If a framing, parity, or break error occurs during reception, the appropriate error bit is set, and is stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately and FIFO data is prevented from being overwritten.

You can program the FIFOs to be 1-byte deep providing a conventional double-buffered UART interface.

The modem status input signals Clear To Send (CTS), Data Carrier Detect (DCD), Data Set Ready (DSR), and Ring Indicator (RI) are supported. The output modem control lines, Request To Send (RTS), and Data Terminal Ready (DTR) are also supported.

There is a programmable hardware flow control feature that uses the **nUARTCTS** input and the **nUARTRTS** output to automatically control the serial data flow.

Figure below shows a block diagram of the UART.



UART block diagram

18.3 Operation

18.3.1 Interface reset

The UART are reset by the global reset signal PRESETn and a block-specific reset signal nUARTRST. An external reset controller must use PRESETn to assert nUARTRST asynchronously and negate it

synchronously to UARTCLK. PRESETn must be asserted LOW for a period long enough to reset the slowest block in the on-chip system, and then be taken HIGH again. The UART requires PRESETn to be asserted LOW for at least one period of PCLK.

The values of the registers after reset are detailed in the next Chapter.

18.3.2 Clock signals

The frequency selected for **UARTCLK** must accommodate the desired range of baud rates:

$\mathbf{F_{UARTCLK}}$ (min) >= 16 x baud_rate (max)

$\mathbf{F_{UARTCLK}}$ (max) <= 16 x 65535 x baud_rate (min)

For example, for a range of baud rates from 110 baud to 460800 baud the **UARTCLK** frequency must be within the range 7.3728MHz to 115MHz.

The frequency of **UARTCLK** must also be within the required error limits for all baud rates to be used.

There is also a constraint on the ratio of clock frequencies for **PCLK** to **UARTCLK**. The frequency of **UARTCLK** must be no more than 5/3 times faster than the frequency of **PCLK**:

$\mathbf{F_{UARTCLK}}$ <= 5/3 x $\mathbf{F_{PCLK}}$

This allows sufficient time to write the received data to the receive FIFO.

18.3.3 UART operation

Control data is written to the UART line control register, UARTLCR_H. This register is 29 bits wide internally, but is externally accessed through the AMBA APB bus by three writes to register locations, UARTLCR_H, UARTIBRD, and UARTFBRD. UARTLCR_H defines:

- transmission parameters
- word length
- buffer mode
- number of transmitted stop bits
- parity mode
- break generation.

UARTIBRD and UARTFBRD together define the baud rate divisor

Fractional baud rate divider

The baud rate divisor is a 22-bit number consisting of a 16-bit integer and a 6-bit fractional part. This is used by the baud rate generator to determine the bit period. The fractional baud rate divider enables the use of any clock with a frequency >3.6864MHz to act as **UARTCLK**, while it is still possible to generate all the standard baud rates.

The 16-bit integer is loaded through the UARTIBRD register. The 6-bit fractional part is loaded into the UARTFBRD register. The Baud Rate Divisor has the following relationship to **UARTCLK**:

Baud Rate Divisor = **UARTCLK**/(16xBaud Rate) = BRD_I + BRD_F

where BRD_I is the integer part and BRD_F is the fractional part separated by a decimal point as shown in Figure below.

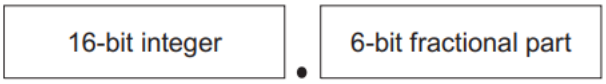


Figure: Baud rate divisor

You can calculate the 6-bit number (m) by taking the fractional part of the required baud rate divisor and multiplying it by 64 (that is, 2^n, where n is the width of the UARTFBRD register) and adding 0.5 to account for rounding errors:

$m = integer(BRD_F * 2^n + 0.5)$

An internal clock enable signal, **Baud16**, is generated, and is a stream of one **UARTCLK** wide pulses with an average frequency of 16 times the desired baud rate. This signal is then divided by 16 to give the transmit clock. A low number in the baud rate divisor gives a short bit period, and a high number in the baud rate divisor gives a long bit period.

Data transmission or reception

Data received or transmitted is stored in two 16-byte FIFOs, though the receive FIFO has an extra four bits per character for status information.

For transmission, data is written into the transmit FIFO. If the UART is enabled, it causes a data frame to start transmitting with the parameters indicated in UARTLCR_H. Data continues to be transmitted until there is no data left in the transmit FIFO. The **BUSY** signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. **BUSY** is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. **BUSY** can be asserted HIGH even though the UART might no longer be enabled.

For each sample of data, three readings are taken and the majority value is kept. In the following paragraphs the middle sampling point is defined, and one sample is taken either side of it.

When the receiver is idle (**UARTRXD** continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by **Baud16**, begins running and data is sampled on the eighth cycle of that counter in normal UART mode, or the fourth cycle of the counter in SIR mode to allow for the shorter logic 0 pulses (half way through a bit period).

The start bit is valid if **UARTRXD** is still LOW on the eighth cycle of **Baud16**, otherwise a false start bit is detected and it is ignored.

If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.

Lastly, a valid stop bit is confirmed if **UARTRXD** is HIGH, otherwise a framing error has occurred. When a full word is received, the data is stored in the receive FIFO, with any error bits associated with that word.

Error bits

Three error bits are stored in bits [10:8] of the receive FIFO, and are associated with a particular character. There is an additional error that indicates an overrun error and this is stored in bit 11 of the receive FIFO.

Overrun bit

The overrun bit is not associated with the character in the receive FIFO. The overrun error is set when the FIFO is full, and the next character is completely received in the shift register. The data in the shift

Table 2-1 Receive FIFO bit functions

FIFO bit	Function
11	Overrun indicator
10	Break error
9	Parity error
8	Framing error
7:0	Received data

register is overwritten, but it is not written into the FIFO. When an empty location is available in the receive FIFO, and another character is received, the state of the overrun bit is copied into the receive FIFO along with the received character. The overrun state is then cleared. Table below shows the bit functions of the receive FIFO.

Disabling the FIFOs

Additionally, you can disable the FIFOs. In this case, the transmit and receive sides of the UART have 1-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received, and the previous one was not yet read. In this implementation, the FIFOs are not physically disabled, but the flags are manipulated to give the illusion of a 1-byte register. When the FIFOs are disabled, a write to the data register bypasses the holding register unless the transmit shift register is already in use.

System and diagnostic loopback testing

You can perform loopback testing for UART data by setting the Loop Back Enable (LBE) bit to 1 in the control register UARTCR (bit 7).

Data transmitted on UARTTXD is received on the UARTRXD input.

18.3.4 UART character frame

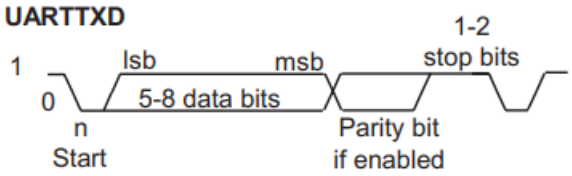


Figure 2-4 UART character frame

18.4 UART modem operation

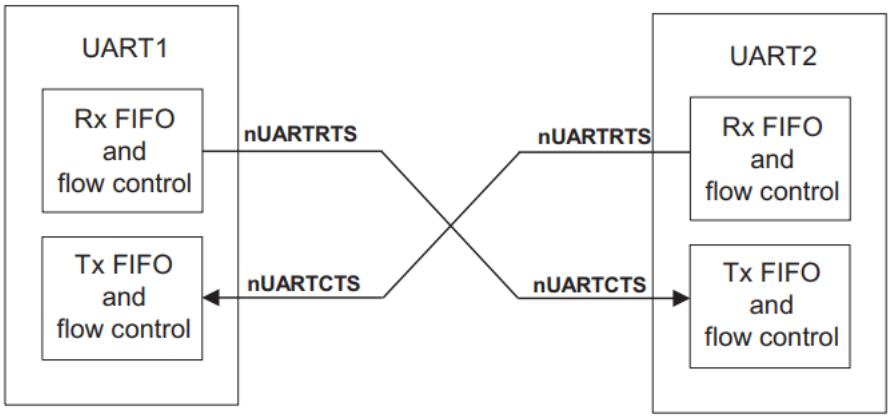
You can use the UART to support both the Data Terminal Equipment (DTE) and Data Communication Equipment (DCE) modes of operation. Figure on page 2-3 shows the modem signals in the DTE mode. For DCE mode, Table 2-2 shows the meaning of the signals.

Table 2-2 Meaning of modem input/output in DTE and DCE modes

Port Name	Meaning	
	DTE	DCE
nUARTCTS	Clear to send	Request to send
nUARTDSR	Data set ready	Data terminal ready
nUARTDCD	Data carrier detect	-
nUARTRI	Ring indicator	-
nUARTRTS	Request to send	Clear to send
nUARTDTR	Data terminal ready	Data set ready
nUARTOUT1	-	Data carrier detect
nUARTOUT2	-	Ring indicator

18.5 UART hardware flow control

The hardware flow control feature is fully selectable, and enables you to control the serial data flow by using the **nUARTRTS** output and **nUARTCTS** input signals. Figure below shows how two devices can communicate with each other using hardware flow control.



When the RTS flow control is enabled, the **nUARTRTS** signal is asserted until the receive FIFO is filled up to the programmed watermark level. When the CTS flow control is enabled, the transmitter can only transmit data when the **nUARTCTS** signal is asserted.

The hardware flow control is selectable through bits 14 (**RTSEn**) and 15 (**CTSEn**) in the UART control register (UARTCR). Table below shows how you must set the bits to enable RTS and CTS flow control both simultaneously, and independently.

Table 2-3 Control bits to enable and disable hardware flow control

CTSEn bit 15 in UARTCR	RTSEn bit 14 in UARTCR	Description
1	1	Both RTS and CTS flow control enabled
1	0	Only CTS flow control enabled
0	1	Only RTS flow control enabled
0	0	Both RTS and CTS flow control disabled

RTS flow control

The RTS flow control logic is linked to the programmable receive FIFO watermark levels. When RTS flow control is enabled, the **nUARTRTS** is asserted until the receive FIFO is filled up to the watermark level. When the receive FIFO watermark level is reached, the **nUARTRTS** signal is deasserted, indicating that there is no more room to receive any more data. The transmission of data is expected to cease after the current character has been transmitted.

The **nUARTRTS** signal is reasserted when data has been read out of the receive FIFO so that it is filled to less than the watermark level. If RTS flow control is disabled and the UART is still enabled, then data is received until the receive FIFO is full, or no more data is transmitted to it.

CTS flow control

If CTS flow control is enabled, then the transmitter checks the **nUARTCTS** signal before transmitting the next byte. If the **nUARTCTS** signal is asserted, it transmits the byte otherwise transmission does not occur.

The data continues to be transmitted while **nUARTCTS** is asserted, and the transmit FIFO is not empty. If the transmit FIFO is empty and the **nUARTCTS** signal is asserted no data is transmitted.

If the **nUARTCTS** signal is deasserted and CTS flow control is enabled, then the current character transmission is completed before stopping. If CTS flow control is disabled and the UART is enabled, then the data continues to be transmitted until the transmit FIFO is empty.

18.6 UART DMA interface

The UART provides an interface to connect to the DMA controller. The DMA operation of the UART is controlled through the UART DMA control register, UARTDMACR. The DMA interface includes the following signals:

For receive:

UARTRXDMASREQ

Single character DMA transfer request, asserted by the UART. For receive, one character consists of up to 12 bits. This signal is asserted when the receive FIFO contains at least one character.

UARTRXDMABREQ

Burst DMA transfer request, asserted by the UART. This signal is asserted when the receive FIFO contains more characters than the programmed watermark level. You can program the watermark level for each FIFO through the **UARTIFLS** register.

UARTRXDMACLR

DMA request clear, asserted by the DMA controller to clear the receive request signals. If DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

For transmit:

UARTTXDMASREQ

Single character DMA transfer request, asserted by the UART. For transmit one character consists of up to eight bits. This signal is asserted when there is at least one empty location in the transmit FIFO.

UARTTXDMABREQ

Burst DMA transfer request, asserted by the UART. This signal is asserted when the transmit FIFO contains less characters than the watermark level. You can program the watermark level for each FIFO through the **UARTIFLS** register.

UARTTXDMACLR

DMA request clear, asserted by the DMA controller to clear the transmit request signals. If DMA burst transfer is requested, the clear signal is asserted during the transfer of the last data in the burst.

The burst transfer and single transfer request signals are not mutually exclusive, they can both be asserted at the same time. For example, when there is more data than the watermark level in the receive FIFO, the burst transfer request and the single transfer request are asserted. When the amount of data left in the receive FIFO is less than the watermark level, the single request only is asserted. This is useful for situations where the number of characters left to be received in the stream is less than a burst.

For example, say 19 characters have to be received and the watermark level is programmed to be four. The DMA controller then transfers four bursts of four characters and three single transfers to complete the stream.

Each request signal remains asserted until the relevant **DMACLR** signal is asserted. After the request clear signal is deasserted, a request signal can become active again, depending on the conditions described above. All request signals are deasserted if the UART is disabled or the DMA enable signal is cleared.

When the UART is in the FIFO disabled mode, only the DMA single transfer mode can operate, since only one character can be transferred to, or from the FIFOs at any time. **UARTRXDMASREQ** and **UARTTXDMASREQ** are the only request signals that can be asserted. When the UART is in the FIFO enabled mode, data transfers can be made by either single or burst transfers depending on the programmed watermark level and the amount of data in the FIFO. Table below shows the trigger points for **DMABREQ** depending on the watermark level, for both the transmit and receive FIFOs.

Table 2-4 DMA trigger points for the transmit and receive FIFOs

	Burst length	
Watermark level	**Transmit (number of empty locations)**	**Receive (number of filled locations)**
1/8	14	2
1/4	12	4
1/2	8	8
3/4	4	12
7/8	2	14

In addition to the above, the DMAONERR bit in the DMA control register supports the use of the receive error interrupt, UARTEINTR. It enables the DMA receive request outputs, UARTRXDMASREQ or UARTRXDMABREQ, to be masked out when the UART error interrupt, UARTEINTR, is asserted. The DMA receive request outputs remain inactive until the UARTEINTR is cleared. The DMA transmit request outputs are unaffected.

Figure below shows the timing diagram for both a single transfer request and a burst transfer request with the appropriate DMA clear signal. The signals are all synchronous to PCLK. For the sake of clarity it is assumed that there is no synchronization of the request signals in the DMA controller.

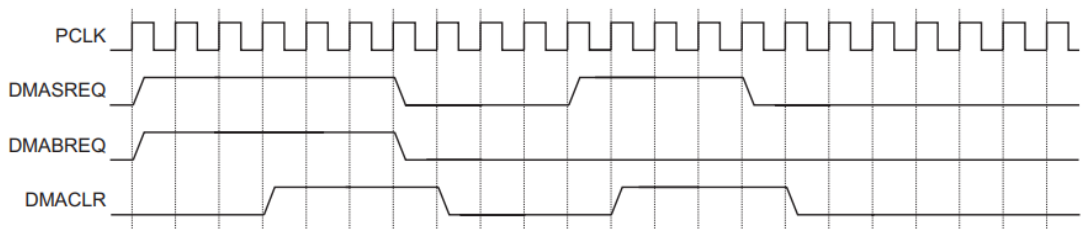


Figure 2-7 DMA transfer waveforms

18.7 Programmer's Model

18.7.1 Summary of registers

Offset	Type	Width	Reset value	Name	Description
0x000	RW	12/8	0x---	UARTDR	Data register
0x004	RW	4/0	0x0	UARTRSR/ UARTECR	Receive status register/error clear register
0x008-0x014	-	-	-	-	Reserved
0x018	RO	9	0b-10010---	UARTFR	Flag register
0x01C-0x020	-	-	-	-	Reserved
0x024	RW	16	0x0000	UARTIBRD	Integer baud rate register
0x028	RW	6	0x00	UARTFBRD	Fractional baud rate register
0x02C	RW	8	0x00	UARTLCR_H	Line control register
0x030	RW	16	0x0300	UARTCR	Control register
0x034	RW	6	0x12	UARTIFLS	Interrupt FIFO level select register
0x038	RW	11	0x000	UARTIMSC	Interrupt mask set/clear register
0x03C	RO	11	0x00-	UARTRIS	Raw interrupt status register
0x040	RO	11	0x00-	UARTMIS	Masked interrupt status register
0x044	WO	11	-	UARTICR	Interrupt clear register
0x048	RW	3	0x00	UARTDMACR	DMA control register

18.7.2 Register descriptions

18.7.2.1 Data register, UARTDR

The UARTDR register is the data register.

For words to be transmitted:

- if the FIFOs are enabled, data written to this location is pushed onto the transmit FIFO
- if the FIFOs are not enabled, data is stored in the transmitter holding register (the bottom word of the transmit FIFO).

The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted.

For received words:

- if the FIFOs are enabled, the data byte and the 4-bit status (break, frame, parity, and overrun) is pushed onto the 12-bit wide receive FIFO
- if the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom word of the receive FIFO).

The received data byte is read by performing reads from the UARTDR register along with the corresponding status information. The status information can also be read by a read of the UARTRSR/UARTECR register as shown in Table on the next page.

Table 3-2 UARTDR register

Bits	Name	Function
15:12	-	Reserved.
11	OE	Overrun error. This bit is set to 1 if data is received and the receive FIFO is already full. This is cleared to 0 once there is an empty space in the FIFO and a new character can be written to it.
10	BE	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits). In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state), and the next valid start bit is received.
9	PE	Parity error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the UARTLCR_H register. In FIFO mode, this error is associated with the character at the top of the FIFO.
8	FE	Framing error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). In FIFO mode, this error is associated with the character at the top of the FIFO.
7:0	DATA	Receive (read) data character. Transmit (write) data character.

Note:

You must disable the UART before any of the control registers are reprogrammed. When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

18.7.2.2 Receive status register/error clear register, UARTRSR/UARTECR

The UARTRSR/UARTECR register is the receive status register/error clear register.

Receive status can also be read from UARTRSR. If the status is read from this register, then the status information for break, framing and parity corresponds to the data character read from UARTDR prior to reading UARTRSR. The status information for overrun is set immediately when an overrun condition occurs.

A write to UARTECR clears the framing, parity, break, and overrun errors. All the bits are cleared to 0 on reset. Table below shows the bit assignment of the UARTRSR/UARTECR register.

Table 3-3 UARTRSR/UARTECR register

Bits	Name	Function
7:0	-	A write to this register clears the framing, parity, break, and overrun errors. The data value is not important.
7:4	-	Reserved, unpredictable when read.
3	OE	Overrun error. This bit is set to 1 if data is received and the FIFO is already full. This bit is cleared to 0 by a write to UARTECR. The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The CPU must now read the data in order to empty the FIFO.
2	BE	Break error. This bit is set to 1 if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity, and stop bits). This bit is cleared to 0 after a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to a 1 (marking state) and the next valid start bit is received.
1	PE	Parity error. When this bit is set to 1, it indicates that the parity of the received data character does not match the parity selected as defined by bits 2 and 7 of the UARTLCR_H register. This bit is cleared to 0 by a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO.
0	FE	Framing error. When this bit is set to 1, it indicates that the received character did not have a valid stop bit (a valid stop bit is 1). This bit is cleared to 0 by a write to UARTECR. In FIFO mode, this error is associated with the character at the top of the FIFO.

Note:

The received data character must be read first from UARTDR before reading the error status associated with that data character from UARTRSR. This read sequence cannot be reversed, because the status register UARTRSR is updated only when a read occurs from the data register UARTDR. However, the status information can also be obtained by reading the UARTDR register.

18.7.2.3 Flag register, UARTFR

The UARTFR register is the flag register. After reset TXFF, RXFF, and BUSY are 0, and TXFE and RXFE are 1. Table below shows the bit assignment of the UARTFR register.

Table 3-4 UARTFR register

Bits	Name	Function
15:9	-	Reserved, do not modify, read as zero.
8	RI	Ring indicator. This bit is the complement of the UART ring indicator (**nUARTRI**) modem status input. That is, the bit is 1 when the modem status input is 0.
7	TXFE	Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty. This bit does not indicate if there is data in the transmit shift register.
6	RXFF	Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.
5	TXFF	Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.
4	RXFE	Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UARTLCR_H register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.
3	BUSY	UART busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether the UART is enabled or not).
2	DCD	Data carrier detect. This bit is the complement of the UART data carrier detect (**nUARTDCD**) modem status input. That is, the bit is 1 when the modem status input is 0.
1	DSR	Data set ready. This bit is the complement of the UART data set ready (**nUARTDSR**) modem status input. That is, the bit is 1 when the modem status input is 0.
0	CTS	Clear to send. This bit is the complement of the UART clear to send (**nUARTCTS**) modem status input. That is, the bit is 1 when the modem status input is 0.

18.7.2.4 Integer baud rate register, UARTIBRD

The UARTIBRD register is the integer part of the baud rate divisor value. All the bits are cleared to 0 on reset. Table below shows the bit assignment of the UARTIBRD register.

Table 3-6 UARTIBRD register

Bits	Name	Function
15:0	BAUD DIVINT	The integer baud rate divisor. These bits are cleared to 0 on reset.

18.7.2.5 Fractional baud rate register, UARTFBRD

The UARTFBRD register is the fractional part of the baud rate divisor value. All the bits are cleared to 0 on reset. Table below shows the bit assignment of register of the UARTFBRD register.

The baud rate divisor is calculated as follows:

Table 3-7 UARTFBRD register

Bits	Name	Function
5:0	BAUD DIVFRAC	The fractional baud rate divisor. These bits are cleared to 0 on reset.

Baud rate divisor BAUDDIV = ($F_{UARTCLK}$/ {16 * Baud rate})

where $F_{UARTCLK}$ is the UART reference clock frequency.

The BAUDDIV is comprised of the integer value (BAUD DIVINT) and the fractional value (BAUD DIVFRAC).

Note:

The contents of the UARTIBRD and UARTFBRD registers are not updated until transmission or reception of the current character is complete.

The minimum divide ratio possible is 1 and the maximum is 65535(2^{16} - 1). That is, UARTIBRD = 0 is invalid and UARTFBRD is ignored when this is the case.

Similarly, when UARTIBRD = 65535 (that is 0xFFFF), then UARTFBRD must not be greater than zero. If this is exceeded it results in an aborted transmission or reception.

This is an example of how to calculate the divisor value.

If the required baud rate is 230400 and UARTCLK = 4MHz then:

Baud Rate Divisor = (4 * 106)/(16 * 230400) = 1.085

Therefore, BRDI = 1 and BRDF = 0.085,

Therefore, fractional part, m = integer((0.085 * 64) + 0.5) = 5

Generated baud rate divider = 1 + 5/64 = 1.078

Generated baud rate = (4 * 106)/(16 * 1.078) = 231911

Error = (231911 - 230400)/230400 * 100 = 0.656%

The maximum error using a 6-bit UARTFBRD register = 1/64 * 100 = 1.56%. This occurs when m = 1, and the error is cumulative over 64 clock ticks.

Table below shows some typical bit rates and their corresponding divisors, given the UART clock frequency of 7.3728MHz. These values do not use the fractional divider so the value in the UARTFBRD register is zero.

Table 3-8 Typical baud rates and divisors

Programmed integer divisor	Bit rate (bps)
0x1	460800
0x2	230400
0x4	115200
0x6	76800
0x8	57600
0xC	38400
0x18	19200
0x20	14400
0x30	9600
0xC0	2400
0x180	1200
0x105D	110

Table below shows some required bit rates and their corresponding integer and fractional divisor values and generated bit rates given a clock frequency of 4MHz.

Table 3-9 Typical baud rates and integer and fractional divisors

Programmed divisor (integer)	Programmed divisor (fraction)	Required bit rate in bps	Generated bit rate in bps	Error %
0x1	0x5	230400	231911	0.656
0x2	0xB	115200	115101	0.086
0x3	0x10	76800	76923	0.160
0x6	0x21	38400	38369	0.081
0x11	0x17	14400	14401	0.007
0x68	0xB	2400	2400	~0
0x8E0	0x2F	110	110	~0

18.7.2.6 Line control register, UARTLCR_H

The UARTLCR_H register is the line control register. This register accesses bits 29 to 22 of the UART bit rate and line control register, UARTLCR.

All the bits are cleared to 0 when reset. Table 3-10 shows the bit assignment of the UARTCR_H register.

Table 3-10 UARTLCR_H register

Bits	Name	Function
15:8	-	Reserved, do not modify, read as zero.
7	SPS	Stick parity select. When bits 1, 2, and 7 of the UARTLCR_H register are set, the parity bit is transmitted and checked as a 0. When bits 1 and 7 are set, and bit 2 is 0, the parity bit is transmitted and checked as a 1. When this bit is cleared stick parity is disabled. Refer to Table 3-11 on page 3-14 for a truth table showing the SPS, EPS and PEN bits.
6:5	WLEN	Word length. The select bits indicate the number of data bits transmitted or received in a frame as follows: 11 = 8 bits 10 = 7 bits 01 = 6 bits 00 = 5 bits.
4	FEN	Enable FIFOs. If this bit is set to 1, transmit and receive FIFO buffers are enabled (FIFO mode). When cleared to 0 the FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers.
3	STP2	Two stop bits select. If this bit is set to 1, two stop bits are transmitted at the end of the frame. The receive logic does not check for two stop bits being received.
2	EPS	Even parity select. If this bit is set to 1, even parity generation and checking is performed during transmission and reception, which checks for an even number of 1s in data and parity bits. When cleared to 0 then odd parity is performed which checks for an odd number of 1s. This bit has no effect when parity is disabled by **Parity Enable** (bit 1) being cleared to 0. Refer to Table 3-11 on page 3-14 for a truth table showing the SPS, EPS and PEN bits.
1	PEN	Parity enable. If this bit is set to 1, parity checking and generation is enabled, else parity is disabled and no parity bit added to the data frame. Refer to Table 3-11 on page 3-14 for a truth table showing the SPS, EPS and PEN bits.
0	BRK	Send break. If this bit is set to 1, a low-level is continually output on the UARTTXD output, after completing transmission of the current character. For the proper execution of the break command, the software must set this bit for at least two complete frames. For normal use, this bit must be cleared to 0.

UARTLCR_H, UARTIBRD and UARTFBRD form a single 30-bit wide register (UARTLCR) which is updated on a single write strobe generated by a UARTLCR_H write. So, in order to internally update the contents of UARTIBRD or UARTFBRD, a UARTLCR_H write must always be performed at the end.

Note:

To update the three registers there are two possible sequences:

- UARTIBRD write, UARTFBRD write and UARTLCR_H write
- UARTFBRD write, UARTIBRD write and UARTLCR_H write.

To update UARTIBRD or UARTFBRD only:

- UARTIBRD write (or UARTFBRD write) and UARTLCR_H write.

Table below is a truth table for the Stick Parity Select (SPS), Even Parity Select (EPS), and Parity ENable (PEN) bits of the UARTLCR_H register.

Table 3-11 Truth table

PEN	EPS	SPS	Parity bit (transmitted or checked)
0	x	x	Not transmitted or checked
1	1	0	Even parity
1	0	0	Odd parity
1	0	1	1
1	1	1	0

Note:

The baud rate and line control registers must not be changed:

- when the UART is enabled
- when completing a transmission or a reception when it has been programmed to become disabled.

The FIFO integrity is not guaranteed under the following conditions:

- after the BRK bit has been initiated
- if the software disables the UART in the middle of a transmission with data in the FIFO, and then re-enables it.

18.7.2.7 Control register, UARTCR

The UARTCR register is the control register. All the bits are cleared to 0 on reset except for bits 9 and 8 which are set to 1. Table 3-12 shows the bit assignment of the UARTCR register.

Note:

Table 3-12 UARTCR register

Bits	Name	Function
15	CTSEn	CTS hardware flow control enable. If this bit is set to 1, CTS hardware flow control is enabled. Data is only transmitted when the **nUARTCTS** signal is asserted.
14	RTSEn	RTS hardware flow control enable. If this bit is set to 1, RTS hardware flow control is enabled. Data is only requested when there is space in the receive FIFO for it to be received.
13	Out2	This bit is the complement of the UART Out2 (nUARTOut2) modem status output. That is, when the bit is programmed to a 1, the output is 0. For DTE this can be used as *Ring Indicator* (RI).
12	Out1	This bit is the complement of the UART Out1 (nUARTOut1) modem status output. That is, when the bit is programmed to a 1 the output is 0. For DTE this can be used as *Data Carrier Detect* (DCD).
11	RTS	Request to send. This bit is the complement of the UART request to send (**nUARTRTS**) modem status output. That is, when the bit is programmed to a 1, the output is 0.
10	DTR	Data transmit ready. This bit is the complement of the UART data transmit ready (**nUARTDTR**) modem status output. That is, when the bit is programmed to a 1, the output is 0.
9	RXE	Receive enable. If this bit is set to 1, the receive section of the UART is enabled. Data reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (bit 1). When the UART is disabled in the middle of reception, it completes the current character before stopping.
8	TXE	Transmit enable. If this bit is set to 1, the transmit section of the UART is enabled. Data transmission occurs for either UART signals, or SIR signals according to the setting of SIR Enable (bit 1). When the UART is disabled in the middle of transmission, it completes the current character before stopping.
7	LBE	Loop back enable. If this bit is set to 1 and the SIR Enable bit is set to 1 and the test register UARTTCR bit 2 (SIRTEST) is set to 1, then the **nSIROUT** path is inverted, and fed through to the **SIRIN** path. The SIRTEST bit in the test register must be set to 1 to override the normal half-duplex SIR operation. This must be the requirement for accessing the test registers during normal operation, and SIRTEST must be cleared to 0 when loopback testing is finished.This feature reduces the amount of external coupling required during system test. If this bit is set to 1, and the SIRTEST bit is set to 0, the UARTTXD path is fed through to the UARTRXD path. In either SIR mode or normal mode, when this bit is set, the modem outputs are also fed through to the modem inputs. This bit is cleared to 0 on reset, which disables the loopback mode.

Table 3-12 UARTCR register (continued)

Bits	Name	Function
6:3	-	Reserved, do not modify, read as zero.
2	SIRLP	IrDA SIR low power mode. This bit selects the IrDA encoding mode. If this bit is cleared to 0, low-level bits are transmitted as an active high pulse with a width of $^3/_{16}$th of the bit period. If this bit is set to 1, low-level bits are transmitted with a pulse width which is 3 times the period of the **IrLPBaud16** input signal, regardless of the selected bit rate. Setting this bit uses less power, but might reduce transmission distances.
1	SIREN	SIR enable. If this bit is set to 1, the IrDA SIR ENDEC is enabled. This bit has no effect if the UART is not enabled by bit 0 being set to 1. When the IrDA SIR ENDEC is enabled, data is transmitted and received on **nSIROUT** and **SIRIN**. **UARTTXD** remains in the marking state (set to 1). Signal transitions on **UARTRXD** or modem status inputs have no effect. When the IrDA SIR ENDEC is disabled, **nSIROUT** remains cleared to 0 (no light pulse generated), and signal transitions on **SIRIN** have no effect.
0	UARTEN	UART enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (bit 1). When the UART is disabled in the middle of transmission or reception, it completes the current character before stopping.

To enable transmission, both TXE, bit 8, and UARTEN, bit 0, must be set. Similarly, to enable reception, RXE, bit 9, and UARTEN, bit 0, must be set.

Note:

Program the control registers as follows:

1. Disable the UART.
2. Wait for the end of transmission or reception of the current character.
3. Flush the transmit FIFO by disabling bit 4 (FEN) in the line control register (UARTCLR_H).
4. Reprogram the control register.
5. Enable the UART

18.7.2.8 Interrupt FIFO level select register, UARTIFLS

The UARTIFLS register is the interrupt FIFO level select register. You can use the UARTIFLS register to define the FIFO level at which the **UARTTXINTR** and **UARTRXINTR** are triggered.

The interrupts are generated based on a transition through a level rather than being based on the level. That is, the design is such that the interrupts are generated when the fill level progresses through the trigger level.

The bits are reset so that the trigger level is when the FIFOs are at the half-way mark. Table below shows the bit assignment of the UARTIFLS register.

Table 3-13 UARTIFLS register

Bits	Name	Function
15:6	-	Reserved, do not modify, read as zero.
5:3	RXIFLSEL	Receive interrupt FIFO level select. The trigger points for the receive interrupt are as follows: 000 = Receive FIFO becomes >= 1/8 full 001 = Receive FIFO becomes >= 1/4 full 010 = Receive FIFO becomes >= 1/2 full 011 = Receive FIFO becomes >= 3/4 full 100 = Receive FIFO becomes >= 7/8 full 101:111 = reserved.
2:0	TXIFLSEL	Transmit interrupt FIFO level select. The trigger points for the transmit interrupt are as follows: 000 = Transmit FIFO becomes <= 1/8 full 001 = Transmit FIFO becomes <= 1/4 full 010 = Transmit FIFO becomes <= 1/2 full 011 = Transmit FIFO becomes <= 3/4 full 100 = Transmit FIFO becomes <= 7/8 full 101:111 = reserved.

18.7.2.9 Interrupt mask set/clear register, UARTIMSC

The UARTIMSC register is the interrupt mask set/clear register.

It is a read/write register. On a read this register gives the current value of the mask on the relevant interrupt. On a write of 1 to the particular bit, it sets the corresponding mask of that interrupt. A write of 0 clears the corresponding mask.

All the bits are cleared to 0 when reset. Table below shows the bit assignment of the UARTIMSC register.

Table 3-14 UARTIMSC register

Bits	Name	Function
15:11	-	Reserved, read as zero, do not modify.
10	OEIM	Overrun error interrupt mask. On a read, the current mask for the **OEIM** interrupt is returned. On a write of 1, the mask of the **OEIM** interrupt is set. A write of 0 clears the mask.
9	BEIM	Break error interrupt mask. On a read the current mask for the **BEIM** interrupt is returned. On a write of 1, the mask of the **BEIM** interrupt is set. A write of 0 clears the mask.
8	PEIM	Parity error interrupt mask. On a read the current mask for the **PEIM** interrupt is returned. On a write of 1, the mask of the **PEIM** interrupt is set. A write of 0 clears the mask.
7	FEIM	Framing error interrupt mask. On a read the current mask for the **FEIM** interrupt is returned. On a write of 1, the mask of the **FEIM** interrupt is set. A write of 0 clears the mask.
6	RTIM	Receive timeout interrupt mask. On a read the current mask for the **RTIM** interrupt is returned. On a write of 1, the mask of the **RTIM** interrupt is set. A write of 0 clears the mask.
5	TXIM	Transmit interrupt mask. On a read the current mask for the **TXIM** interrupt is returned. On a write of 1, the mask of the **TXIM** interrupt is set. A write of 0 clears the mask.
4	RXIM	Receive interrupt mask. On a read the current mask for the **RXIM** interrupt is returned. On a write of 1, the mask of the **RXIM** interrupt is set. A write of 0 clears the mask.
3	DSRMIM	**nUARTDSR** modem interrupt mask. On a read the current mask for the **DSRMIM** interrupt is returned. On a write of 1, the mask of the **DSRMIM** interrupt is set. A write of 0 clears the mask.
2	DCDMIM	**nUARTDCD** modem interrupt mask. On a read the current mask for the **DCDMIM** interrupt is returned. On a write of 1, the mask of the **DCDMIM** interrupt is set. A write of 0 clears the mask.
1	CTSMIM	**nUARTCTS** modem interrupt mask. On a read the current mask for the **CTSMIM** interrupt is returned. On a write of 1, the mask of the **CTSMIM** interrupt is set. A write of 0 clears the mask.
0	RIMIM	**nUARTRI** modem interrupt mask. On a read the current mask for the **RIMIM** interrupt is returned. On a write of 1, the mask of the RIMIM interrupt is set. A write of 0 clears the mask.

18.7.2.10 Raw interrupt status register, UARTRIS

The UARTRIS register is the raw interrupt status register. It is a read-only register. On a read this register gives the current raw status value of the corresponding interrupt. A write has no effect.

Caution: All the bits, except for the modem status interrupt bits (bits 3 to 0), are cleared to 0 when reset. The modem status interrupt bits are undefined after reset.

Table below shows the bit assignment of the UARTRIS register.

Table 3-15 UARTRIS register

Bits	Name	Function
15:11	-	Reserved, read as zero, do not modify
10	OERIS	Overrun error interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTOEINTR** interrupt
9	BERIS	Break error interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTBEINTR** interrupt
8	PERIS	Parity error interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTPEINTR** interrupt
7	FERIS	Framing error interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTFEINTR** interrupt
6	RTRIS	Receive timeout interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTRTINTR** interrupt[a]
5	TXRIS	Transmit interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTTXINTR** interrupt
4	RXRIS	Receive interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTRXINTR** interrupt
3	DSRRMIS	**nUARTDSR** modem interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTDSRINTR** interrupt
2	DCDRMIS	**nUARTDCD** modem interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTDCDINTR** interrupt
1	CTSRMIS	**nUARTCTS** modem interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTCTSINTR** interrupt
0	RIRMIS	**nUARTRI** modem interrupt status. Gives the raw interrupt state (prior to masking) of the **UARTRIINTR** interrupt

18.7.2.11 Masked interrupt status register, UARTMIS

The UARTMIS register is the masked interrupt status register. It is a read-only register. On a read this register gives the current masked status value of the corresponding interrupt. A write has no effect.

All the bits except for the modem status interrupt bits (bits 3 to 0) are cleared to 0 when reset. The modem status interrupt bits are undefined after reset. Table below shows the bit assignment of the UARTMIS register.

Table 3-16 UARTMIS register

Bits	Name	Function
15:11	-	Reserved, read as zero, do not modify
10	OEMIS	Overrun error masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTOEINTR** interrupt
9	BEMIS	Break error masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTBEINTR** interrupt
8	PEMIS	Parity error masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTPEINTR** interrupt
7	FEMIS	Framing error masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTFEINTR** interrupt
6	RTMIS	Receive timeout masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTRTINTR** interrupt
5	TXMIS	Transmit masked interrupt status. Gives the masked interrupt state (after masking) of the UARTTXINTR interrupt
Bits	**Name**	**Function**
4	RXMIS	Receive masked interrupt status. Gives the masked interrupt state (after masking) of the UARTRXINTR interrupt
3	DSRMMIS	**nUARTDSR** modem masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTDSRINTR** interrupt
2	DCDMMIS	**nUARTDCD** modem masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTDCDINTR** interrupt
1	CTSMMIS	**nUARTCTS** modem masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTCTSINTR** interrupt
0	RIMMIS	**nUARTRI** modem masked interrupt status. Gives the masked interrupt state (after masking) of the **UARTRIINTR** interrupt

18.7.2.12 Interrupt clear register, UARTICR

The UARTICR register is the interrupt clear register and is write-only. On a write of 1, the corresponding interrupt is cleared. A write of 0 has no effect. Table below shows the bit assignment of the UARTICR register.

Table 3-17 UARTICR register

Bits	Name	Function
15:11	Reserved	Reserved, read as zero, do not modify
10	OEIC	Overrun error interrupt clear. Clears the **UARTOEINTR** interrupt
9	BEIC	Break error interrupt clear. Clears the **UARTBEINTR** interrupt
8	PEIC	Parity error interrupt clear. Clears the **UARTPEINTR** interrupt
7	FEIC	Framing error interrupt clear. Clears the **UARTFEINTR** interrupt
6	RTIC	Receive timeout interrupt clear. Clears the **UARTRTINTR** interrupt
5	TXIC	Transmit interrupt clear. Clears the **UARTTXINTR** interrupt
4	RXIC	Receive interrupt clear. Clears the **UARTRXINTR** interrupt
3	DSRMIC	**nUARTDSR** modem interrupt clear. Clears the **UARTDSRINTR** interrupt
2	DCDMIC	**nUARTDCD** modem interrupt clear. Clears the **UARTDCDINTR** interrupt
1	CTSMIC	**nUARTCTS** modem interrupt clear. Clears the **UARTCTSINTR** interrupt
0	RIMIC	**nUARTRI** modem interrupt clear. Clears the **UARTRIINTR** interrupt

18.7.2.13 DMA control register, UARTDMACR

The UARTDMACR register is the DMA control register. It is a read/write register. All the bits are cleared to 0 on reset. Table below shows the bit assignment of the UARTDMACR register.

Table 3-18 UARTDMACR register

Bits	Name	Function
15:3	-	Reserved, read as zero, do not modify.
2	DMAONERR	DMA on error. If this bit is set to 1, the DMA receive request outputs, **UARTRXDMASREQ** or **UARTRXDMABREQ**, are disabled when the UART error interrupt is asserted.
1	TXDMAE	Transmit DMA enable. If this bit is set to 1, DMA for the transmit FIFO is enabled.
0	RXDMAE	Receive DMA enable. If this bit is set to 1, DMA for the receive FIFO is enabled.

19 Inter-integrated circuit(I2C)

19.1 I2C introduction

The I2C interface is an internal circuit allowing communication with an external I2C interface which is an industry standard two line serial interface used for connection to external hardware.

These two serial lines are known as a serial data line (SDA) and a serial clock line (SCL). The I2C module provides several data transfer rates of up to 100 kHz in standard mode, up to 400 kHz in the fast mode and up to 1 MHz in the fast mode plus .

The I2C module also has an arbitration detect function to prevent the situation where more than one master attempts to transmit data to the I2C bus at the same time. A CRC-8 calculator is also provided in I2C interface to perform packet error checking for I2C data.

AG32 device provide:

- Up to two I2C bus interfaces can support master mode with a frequency up to 1 MHz (Fast mode plus)
- Provide arbitration function, optional PEC (packet error checking) generation and checking
- Supports 7-bit and 10-bit addressing mode and general call addressing mode

19.2 Architecture

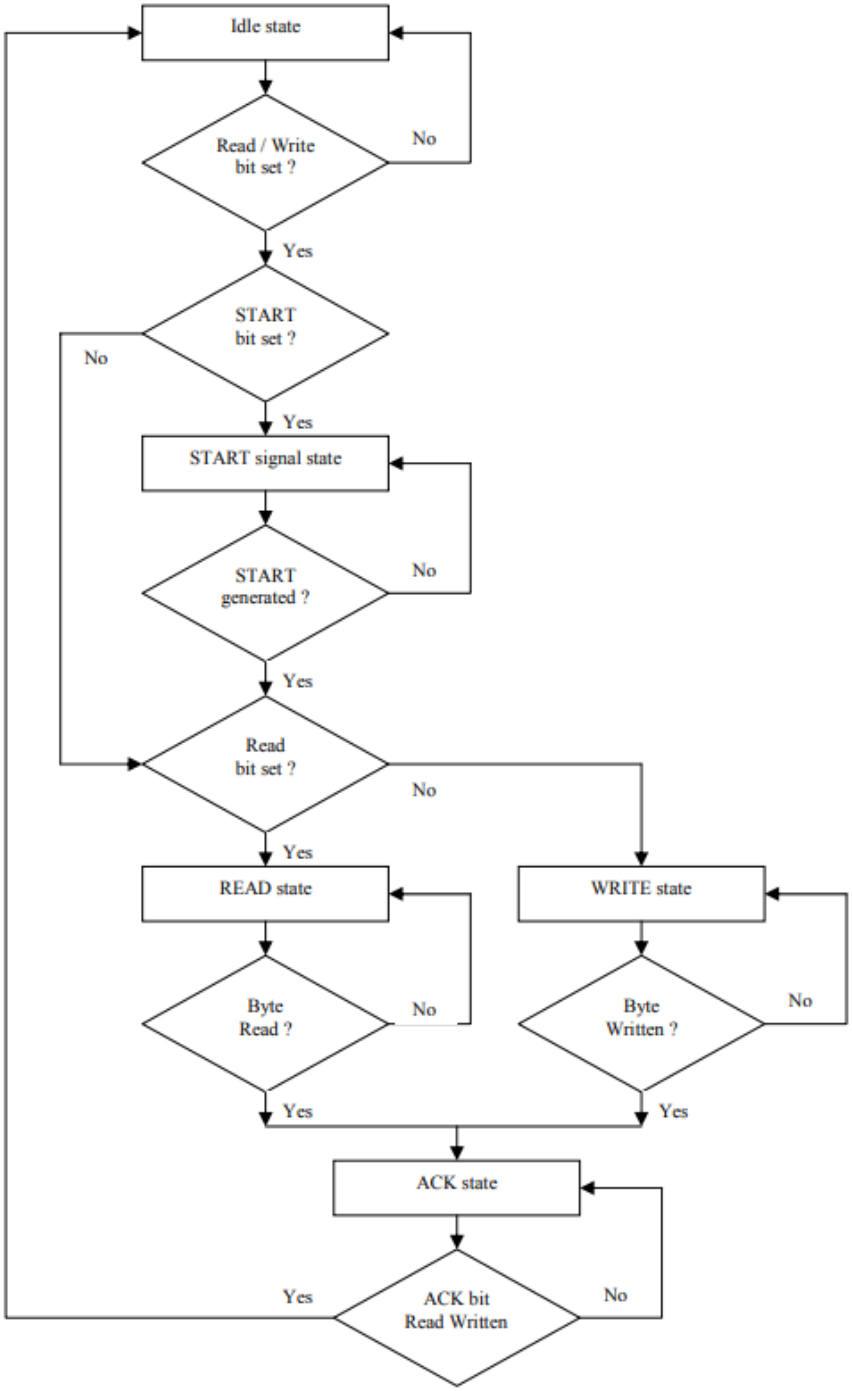
The I^2C core is built around four primary blocks; the Clock Generator, the Byte Command Controller, the Bit Command Controller and the DataIO Shift Register. All other blocks are used for interfacing or for storing temporary values.

Clock Generator

The Clock Generator generates an internal 4*Fscl clock enable signal that triggers all synchronous elements in the Bit Command Controller. It also handles clock stretching needed by some slaves.

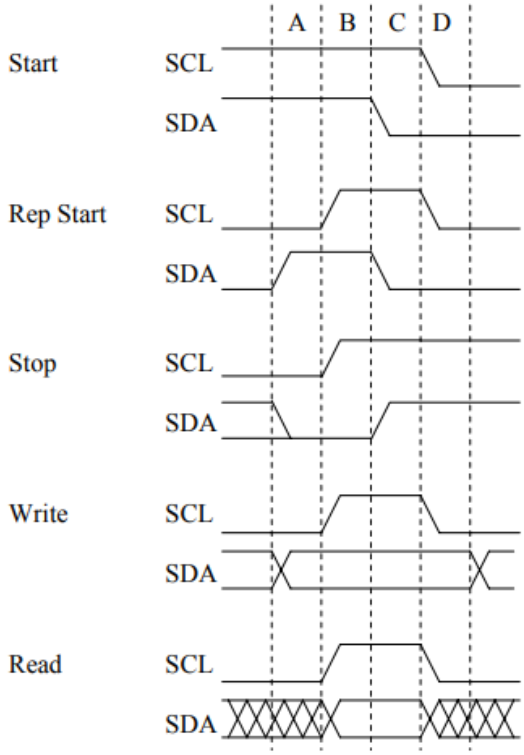
Byte Command Controller

The Byte Command Controller handles I^2C traffic at the byte level. It takes data from the Command Register and translates it into sequences based on the transmission of a single byte. By setting the START, STOP, and READ bit in the Command Register, for example, the Byte Command Controller generates a sequence that results in the generation of a START signal, the reading of a byte from the slave device, and the generation of a STOP signal. It does this by dividing each byte operation into separate bit-operations, which are then sent to the Bit Command Controller.



Bit Command Controller

The Bit Command Controller handles the actual transmission of data and the generation of the specific levels for START, Repeated START, and STOP signals by controlling the SCL and SDA lines. The Byte Command Controller tells the Bit Command Controller which operation has to be performed. For a single byte read, the Bit Command Controller receives 8 separate read commands. Each bit-operation is divided into 5 pieces (idle and A, B, C, and D), except for a STOP operation which is divided into 4 pieces (idle and A, B, and C).



DataIO Shift Register

The DataIO Shift Register contains the data associated with the current transfer. During a read action, data is shifted in from the SDA line. After a byte has been read the contents are copied into the Receive Register. During a write action, the Transmit Register’s contents are copied into the DataIO Shift Register and are then transmitted onto the SDA line.

19.3 Operation

19.3.1 System Configuration

The I^2C system uses a serial data line (SDA) and a serial clock line (SCL) for data transfers. All devices connected to these two signals must have open drain or open collector outputs. The logic AND function

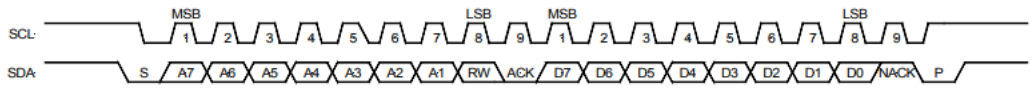
is exercised on both lines with external pull-up resistors.

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line on a byte-by-byte basis. Each data byte is 8 bits long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (see START and STOP signals).

19.3.2 I²C Protocol

Normally, a standard communication consists of four parts:

1) START signal generation
2) Slave address transfer
3) Data transfer
4) STOP signal generation



START signal

When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the S-bit, is defined as a high-to-low transition of SDA while SCL is high. The START signal denotes the beginning of a new data transfer. A Repeated START is a START signal without first generating a STOP signal. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus. The core generates a START signal when the STA-bit in the Command Register is set and the RD or WR bits are set. Depending on the current status of the SCL line, a START or Repeated START is generated.

Slave Address Transfer

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a seven-bits calling address followed by a RW bit. The RW bit signals the slave the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

Note: The core supports 10bit slave addresses by generating two address transfers. See the Philips I²C specifications for more details.

The core treats a Slave Address Transfer as any other write action. Store the slave device’s address in the Transmit Register and set the WR bit. The core will then transfer the slave address on the bus.

Data Transfer

Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a No Acknowledge, the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as the receiving device, does not acknowledge the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

To write data to a slave, store the data to be transmitted in the Transmit Register and set the WR bit. To read data from a slave, set the RD bit. During a transfer the core set the TIP flag, indicating that a Transfer is In Progress. When the transfer is done the TIP flag is reset, the IF flag set and, when enabled, an interrupt generated. The Receive Register contains valid data after the IF flag has been set. The user may issue a new write or read command when the TIP flag is reset.

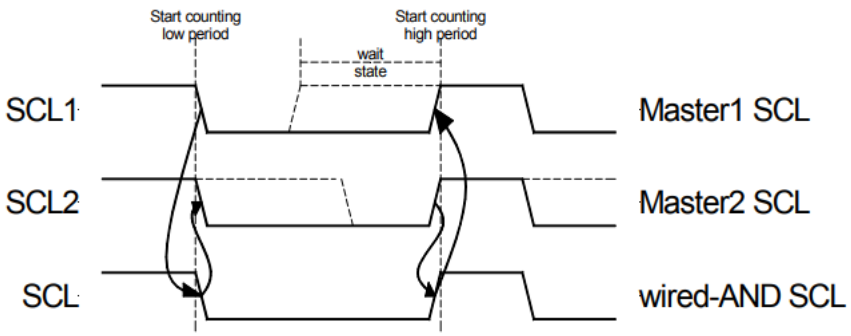
STOP signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the P-bit, is defined as a low-to-high transition of SDA while SCL is at logical ‘1’.

19.3.3 Arbitration Procedure

Clock Synchronization

The I^2C bus is a true multimaster bus that allows more than one master to be connected on it. If two or more masters simultaneously try to control the bus, a clock synchronization procedure determines the bus clock. Because of the wired-AND connection of the I^2C signals a high to low transition affects all devices connected to the bus. Therefore a high to low transition on the SCL line causes all concerned devices to count off their low period. Once a device clock has gone low it will hold the SCL line in that state until the clock high state is reached. Due to the wired-AND connection the SCL line will therefore be held low by the device with the longest low period, and held high by the device with the shortest high period.



Clock Stretching

Slave devices can use the clock synchronization mechanism to slow down the transfer bit rate. After

the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave’s SCL low period is greater than the master’s SCL low period, the resulting SCL bus signal low period is stretched, thus inserting wait-states.

19.4 Registers

19.4.1 Registers list

Name	Address	Width	Access	Description
PRERlo	0x00	8	RW	Clock Prescale register lo-byte
PRERhi	0x01	8	RW	Clock Prescale register hi-byte
CTR	0x02	8	RW	Control register
TXR	0x03	8	W	Transmit register
RXR	0x03	8	R	Receive register
CR	0x04	8	W	Command register
SR	0x04	8	R	Status register

19.4.2 Register description

Prescale Register

This register is used to prescale the SCL clock line. Due to the structure of the I^2C interface, the core uses a 5*SCL clock internally. The prescale register must be programmed to this 5*SCL frequency (minus 1). Change the value of the prescale register only when the ‘EN’ bit is cleared.

Example: wb_clk_i = 32MHz, desired SCL = 100KHz

$$prescale = \frac{32\ MHz}{5 * 100\ KHz} - 1 = 63\ (dec) = 3F\ (hex)$$

Reset value: 0xFFFF

Control register

Bit #	Access	Description
7	RW	EN, I^2C core enable bit. When set to ‘1’, the core is enabled. When set to ‘0’, the core is disabled.
6	RW	IEN, I^2C core interrupt enable bit. When set to ‘1’, interrupt is enabled. When set to ‘0’, interrupt is disabled.
5:0	RW	*Reserved*

Reset Value: 0x00

The core responds to new commands only when the ‘EN’ bit is set. Pending commands are

finished. Clear the ‘EN’ bit only when no transfer is in progress, i.e. after a STOP command, or when the command register has the STO bit set. When halted during a transfer, the core can hang the I2 C bus.

Transmit register

Bit #	Access	Description
7:1	W	Next byte to transmit via I^2C
0	W	In case of a data transfer this bit represent the data’s LSB. In case of a slave address transfer this bit represents the RW bit. ‘1’ = reading from slave ‘0’ = writing to slave

Reset value: 0x00

Receive register

Bit #	Access	Description
7:0	R	Last byte received via I^2C

Reset value: 0x00

Command register

Bit #	Access	Description
7	W	STA, generate (repeated) start condition
6	W	STO, generate stop condition
5	W	RD, read from slave
4	W	WR, write to slave
3	W	ACK, when a receiver, sent ACK (ACK = ‘0’) or NACK (ACK = ‘1’)
2:1	W	*Reserved*
0	W	IACK, Interrupt acknowledge. When set, clears a pending interrupt.

Reset Value: 0x00

The STA, STO, RD, WR, and IACK bits are cleared automatically. These bits are always read as zeros.

Status register

Bit #	Access	Description
7	R	RxACK, Received acknowledge from slave. This flag represents acknowledge from the addressed slave. '1' = No acknowledge received '0' = Acknowledge received
6	R	Busy, I²C bus busy '1' after START signal detected '0' after STOP signal detected
5	R	AL, Arbitration lost This bit is set when the core lost arbitration. Arbitration is lost when: • a STOP signal is detected, but non requested • The master drives SDA high, but SDA is low. See *bus-arbitration* section for more information.
4:2	R	*Reserved*
1	R	TIP, Transfer in progress. '1' when transferring data '0' when transfer complete
0	R	IF, Interrupt Flag. This bit is set when an interrupt is pending, which will cause a processor interrupt request if the IEN bit is set. The Interrupt Flag is set when: • one byte transfer has been completed • arbitration is lost

Reset Value: 0x00

Please note that all reserved bits are read as zeros. To ensure forward compatibility, they should be written as zeros.

20 Controller area network (CAN)

20.1 Overview

AG32 device provides:

- One CAN2.0B interface with communication frequency up to 1 Mbit/s
- Internal main PLL for CAN CLK compliantly

Controller area network (CAN) is a method for enabling serial communication in field bus. The CAN protocol has been used extensively in industrial automation and automotive applications. It can receive and transmit standard frames with 11-bit identifiers as well as extended frames with 29-bit identifiers. The CAN has three mailboxes for transmission and two FIFOs of three message deep for reception. It also provides 14 scalable/configurable identifier filter banks for selecting the incoming messages needed and discarding the others.

AG32 support The Inventra™ MCAN2 standard.

The MCAN2 has two main modes of operation: an Operating Mode in which data may be transmitted and received, and a Reset Mode in which bus timing parameters and message acceptance filters can be set. Reset Mode also allows the Receive and Transmit Error Counters and the Error Warning Limit to be changed.

Reset Mode is selected either by executing a hardware reset or by setting the Reset Mode bit in the Mode Register (MOD.0) to '1'. The MCAN2 is returned to Operating Mode by clearing the MOD.0 bit.

The MCAN2 also supports a Listen Only Mode and a Self Test Mode, selectable through the Mode Register in either Operating Mode or Reset Mode.

In Listen Only Mode, the MCAN2 is only able to receive data: no transmission is possible. The MCAN2 does not even transmit any acknowledgement of data being successfully received. It is also forced to be 'error passive'.

In Self Test Mode, the MCAN2 sends and receives a message using the MCAN2's Self Reception feature without looking for any acknowledgement from any remote node.

The device also offers a Clock Output Mode, only selectable within Reset Mode, in which TX1 is used to output the Transmit clock rather than a second copy of the transmission data.

20.2 Operation

20.2.1 Configuration

The way in which the MCAN2 is configured to operate is set within its Reset Mode, into which it is placed both immediately following a hardware reset (NRST taken low) and as a result of setting the Reset Mode bit in the Mode register (MOD.0) to 1 (software reset).

The details of the register settings following both these events are given in an Appendix at the end of this document.

While in Reset Mode, you may wish to set the following aspects of the MCAN2's operation:

- The bus timing parameters to be applied (These select the baud rate used on the CAN bus)
- The acceptance filters to be applied to received messages
- The required interrupts
- The desired error warning limit
- The required output mode – with either a copy of the transmission bit stream or the Transmit clock on TX1
- The relationship of the CLKOUT signal to the input clock.

20.2.2 Bus Timing Parameters

The bus timing parameters configure the MCAN2 for the bit rate used on the CAN bus and set the point within each bit period at which the received bit stream is to be sampled. They also specify the degree to which the MCAN2 may compensate for variations in the bit rates generated by other nodes by re-synchronizing to the bit stream.

To cater for variations in the bit rate generated by other nodes and for physical delay times both on the bus and within the CAN nodes, the bit period is seen as being composed of a Synchronization segment, a Propagation segment and two Phase Buffers. The Synchronization segment represents the part of the bit period in which the bit edge is expected to arrive. The Propagation segment represents the part of the bit time that is allowed to compensate for physical delay times. The two Phase Buffers surround the sampling point and are shortened or lengthened as necessary to re-synchronize to the incoming bit stream when the bit edge arrives outside of the Synchronization segment.

The length of each of these segments is defined as a number of 'Time Quanta' (TQ). The Synchronization segment is always 1 TQ, the Propagation segment may be 1 – 8 TQ, and the two Phase Buffers may be 1 – 8 TQ. The maximum amount by which the Phase Buffers can be lengthened or shortened is also defined – as the Synchronization Jump Width. This is limited to 1 – 4 TQ and is also required to not be longer than either of the two Phase Buffers.

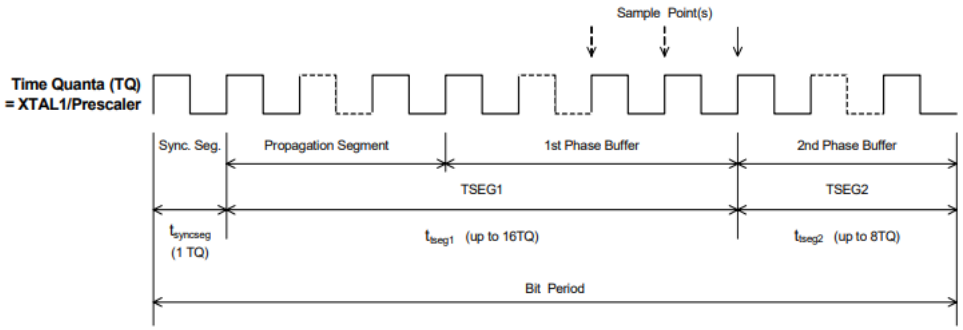
The timing parameters used on the MCAN2 are selected through the two Bus Timing Registers: BTR0 and BRT1.

BTR0 defines the time quantum to be used in terms of periods of the XTAL1 input clock, together with the Synchronization Jump Width (in time quanta). Time quanta between 2 x the XTAL1 clock period and 128 x the XTAL1 clock period are supported.

BTR1 defines the lengths in time quanta of two time segments, TSEG1 and TSEG2, and the number of samples made of each bit period – 1 or 3 (1 is recommended for high-speed (class C) buses; 3 is

recommended for low/medium (class A or B) buses). TSEG1 represents the time between the Synchronization segment and the sample point (i.e. the Propagation segment plus the first Phase Buffer). TSEG2 represents the time between the sample point and the end of the bit period (i.e. the second Phase Buffer).

General Structure of a Bit Period



TSEG1 can be between 1 and 16 TQ long while TSEG2 can be between 1 and 8 TQ long. In theory, bit periods can therefore be defined between 3 and 25 TQ in length. In practice, however, they are required to be in the range 8 and 25 TQ picked out by the BOSCH standard.

20.2.3 Acceptance Filters

Within a CAN network, all nodes receive all messages transmitted on the bus.

To allow a node to ignore messages that are not relevant to it, the MCAN2 provides a 4-byte Acceptance Filter, which can be used to pick out only those messages with an appropriate identifier. Any message that does not pass though this filter can be discarded as not applicable to the receiving CAN node.

Normally message filtering is based upon the whole identifier, which can be 11 or 29 bits long depending if the received message is a standard or extended frame format. However, in the MCAN2, optional mask registers allow groups of identifiers to be received and placed into the Receive FIFO by setting particular identifier bits to be ‘don’t care’.

The filter can be applied either as a single 4-byte filter or as two shorter filters. The selection is made through the AFM bit of the Mode register (bit 3). If AFM = ‘1’, a single filter will be applied; if AFM = ‘0’, two filters will be applied. Where two filters are used, the incoming message is accepted if its identifier matches either filter.

The filters applied are defined in a set of Acceptance Code Registers ACR0 – 3, used in conjunction with a corresponding set of Acceptance Mask Registers AMR0 – 3 (see Sections 10.1 and 10.2). The bit pattern against which the message identifier is matched is recorded in the ACR registers, masked by the values recorded in the AMR registers. ‘0’s in AMR0 – 3 identify the bits at the corresponding positions in ACR0 – 3 which must be matched in the message identifier, ‘1’s identify the corresponding bits as ‘don’t care’. Both groups of registers are set to zero by a hardware reset (i.e. set to accept only messages with a zero identifier) but are left unchanged by a software reset.

The way in which the bit patterns defined by ACR0 – 3 are applied further depend on whether the incoming message is in Standard Frame Format (SFF) or Extended Frame Format (EFF).

20.2.4 Interrupts

The MCAN2 supports the generation of an interrupt for any of the following conditions:

• Bus activity while the MCAN2 is in Sleep Mode (Wake-Up Interrupt)

• Receipt of a message (Receive Interrupt)

• Completion of the current transmission (Transmit Interrupt)

• Loss of Received data through the FIFO being full (Data Overrun Interrupt)

• Loss of Arbitration on the CAN bus* (Arbitration Loss Interrupt)

• Error on the CAN bus* (Bus Error Interrupt)

• MCAN2 coming out of 'Error Passive' state (Error Passive Interrupt)

• The number of errors either exceeding the Error Warning Limit or causing the device to go into Bus Off state (Error Warning Interrupt)

Following a hardware reset, these interrupts are disabled. The user therefore needs to enable the ones they require in the Interrupt Enable Register. The selection of interrupts that are enabled is not however affected by a software reset.

20.2.5 Error Warning Limit

The Error Warning Limit (EWL) represents the number of errors in either reception or transmission at which a warning should be generated. When either the Transmit Error Counter or the Receive Error Counter passes this value, the Error Status bit in the Status Register (SR.6) is set and an Error Warning Interrupt is generated (if enabled).

The value for the EWL is recorded in the Error Warning Limit Register (described in Section 10.9). The value selected following a hardware reset is 96 which, if reached, would indicate a seriously disturbed bus.

The current setting is left unchanged by a software reset.

20.2.6 Output Mode

The MCAN2 supports two possible output driver configurations: 'Normal Output' and 'Clock Output'. Note: The additional driver configurations available in the SJA1000 through this register are not supported by the MCAN2.

In Normal Output Mode, the bit sequence (TXD) is sent to TX0 with an inverse copy sent to TX1. In Clock Output Mode, the bit sequence is output on the TX0 signal as in Normal Output Mode but the data stream to TX1 is replaced by a copy of the Transmit clock (TXCLK), the rising edge of which marks the beginning of a bit period.

Normal Output Mode is automatically selected following a hardware reset. If Clock Output Mode is required, it may be selected through the Output Control Register.

The selected mode is left unchanged by a software reset.

20.2.7 CLKOUT Signal

The CLKOUT signal is derived from the XTAL1 input clock.

The relationship between the CLKOUT signal and the XTAL1 clock is defined by the Clock Divider Register. The bottom three bits of this register specify a divisor for the XTAL1 clock between 2 and 14, while bit 3 of the register enables or disables the CLKOUT signal as required.

After a hardware reset, the Clock Divider Register is set so that the CLKOUT signal is enabled and equal to XTAL1 divided by 2. The current setting is left unchanged by a software reset.

20.2.8 Example Configuration Steps

Address	Register	Action
1Fh	Clock Divider	Set CLKOUT frequency, derived from XTAL1 input.
08h	Output Control Register	Configure the output driver for outputs TX0 and TX1
04h	Interrupt Enable	Enable required interrupts
00h	Mode	Select message filter type
10h-13h	Acceptance Code Registers	Set to select required range of identifiers
14h-17h	Acceptance Mask Registers	
06h	Bus Timing Register:0	Set SJW and clocks per Time Quanta
07h	Bus Timing Register:1	Set Tseg1 and Tseg2, and therefore sample point of bit period
00'h	Mode	Release Reset Mode

20.3 Interrupt Handling

When the CPU is interrupted (NINT going low), it needs to read the interrupt register to determine which type of event caused the interrupt.

The possible interrupts are (in the order they appear in the Interrupt register):

- Receive Interrupt (IR.0 set)
- Transmit Interrupt (IR.1 set)
- Error Warning Interrupt (IR.2 set)
- Data Overrun Interrupt (IR.3 set)
- Wake-Up Interrupt (IR.4 set)
- Error Passive Interrupt (IR.5 set)
- Arbitration Loss Interrupt (IR.6 set)
- Bus Error Interrupt (IR.7 set)

The following sections describe the actions to be taken in response to each type of interrupt.

20.3.1 Receive Interrupt

The generation of a Receive Interrupt indicates the availability of a message to be read in the Receive FIFO.

The message is read through a 13-byte window onto the Receive FIFO referred the Receive Buffer, which is located at CAN addresses 10h – 1Ch.

Once the message currently accessible through the Receive Buffer has been read, the CPU needs to release the window it currently has on the FIFO by issuing a Release Receive Buffer command (CMR.2 = '1'). The RX FIFO Read Pointer (and hence the Receive Buffer Start Address) then moves to the position in the Receive FIFO at which the next message will start.

If there is an unread message at this position, this becomes immediately available to read through the Receive Buffer. If no message is available, the Receive Interrupt (IR.0) and Receive Buffer Status (SR.0) bits will be cleared.

20.3.2 Transmit Interrupt

The generation of a Transmit Interrupt indicates the readiness of the Transmit Buffer to receive another message for transmission. The response made to this interrupt simply depends on whether there is further data to be sent. If there is, the transmission procedure outlined in Section 4 needs to be repeated. If not, the interrupt may be ignored.

20.3.3 Error Warning Interrupt

The generation of an Error Warning interrupt indicates either that the count of transmission errors or the count of reception errors has passed the EWL value recorded in the Error Warning Limit register, or that the MCAN2 has been put into Bus Off state because the number of transmission errors has exceeded 255.

The count of reception errors is recorded in the RXERR register, the count of transmission errors is recorded in the TXERR register.

If the MCAN2 has been placed in Bus Off state, the Bus Status bit (SR.7) will be set to '1' (Bus Off). In addition, the Reset Mode bit (MOD.0) will have been set, causing a software reset and placing the MCAN2 in Reset Mode where it will then stay until the host CPU clears the Reset Mode bit in the Mode Register (MOD.0).

Furthermore, on its return to Operating Mode, the MCAN2 will wait for 128 occurrences of the Bus Free sequence of 11 successive recessive bits (the minimum time defined by the CAN protocol) before becoming 'Bus On' again. Note: During this period, the progress that is being made towards Bus On can be monitored by reading the TXERR register. On leaving Reset Mode, this is initially set to 127. It then counts down through the required number of Bus Free sequences to become zero at the point when the device is allowed to become Bus On again.

If the interrupt has been generated as a result of the EWL value being exceeded, it is up to the programmer what action is taken in response to the generation of this interrupt.

20.3.4 Data Overrun Interrupt

A Data Overrun Interrupt is only generated when the required storage space for the received message is greater than the number of free bytes in the Receive FIFO. The Data Overrun Status bit (SR.1) will also be set.

The required storage space is determined from the RTR, FF and DLC bits of the received message which respectively define whether the message is a Remote Transmission Request, whether it is a standard frame format or extended frame format message, and the number of bytes included in the message.

The assessment of the space required is made after the message has been received. If insufficient space is available to store the message, the message will be lost.

The recovery that can be made when messages are lost will depend on the system design. However, experiencing significant numbers of Data Overrun events would suggest that the volume of data traffic has been under-estimated and that the system would benefit from a larger memory buffer for incoming messages.

20.3.5 Wake-up Interrupt

A Wake-Up Interrupt is generated when the MCAN2 is awakened from Sleep Mode.

Any of the following events will cause the MCAN2 to 'wake up' from Sleep Mode:

- Clearing the Sleep Mode bit (MOD.4)
- A low on NINT_IN
- Activity on the CAN bus input (RX0)

It is up to the CPU to identify why the device has been awoken, for example by first reading the Mode register then testing the level of NINT_IN.

20.3.6 Error Passive Interrupt

The Receive Error (RXERR) and Transmit Error (TXERR) counters are respectively automatically incremented by one each time a Receive error or Transmit error occurs, and decremented by one by each successful reception or transmission.

If the accumulated total of either Receive or Transmit Errors goes over 127, the MCAN2 goes into state in which further errors continue to be counted but individual interrupts are no longer generated. This state is described as 'Error Passive' and an Error Passive Interrupt is generated (if enabled) to signal that the Error Passive state has been entered.

The MCAN2 remains in Error Passive state while either error count remains over 127. The Transmission Error count continues to be incremented and decremented while it remains over 127. The Receive Error count, however, is automatically reduced to a value between 119 and 127 by each message that is successfully received, potentially taking the MCAN2 out of Error Passive state.

20.3.7 Arbitration Loss Interrupt

The generation of an Arbitration Loss Interrupt indicates that the MCAN2 has lost control of the CAN bus while it was in the process of transmitting a message.

Normally, there is no need for any special action to be taken as the MCAN2 will automatically try again to transmit the current message. The fact that arbitration has been lost may however be of importance if the option of a One-Shot transmission has been taken .

The bit position at which arbitration was lost will be recorded in the Arbitration Lost Capture (ALC) Register. For details of the way in which this bit position is recorded.

20.3.8 Bus Error Interrupt

The generation of a Bus Error Interrupt indicates the occurrence of a transmission error on the CAN bus.

Normally, there is no need for any special action to be taken as the MCAN2 will automatically discard any incoming message in which bus errors have occurred and it will automatically try to send again any transmit message that experienced bus errors. However, should additional information on a bus error be required, the type of error (bit/form/stuff/other) and the location of the each error are captured in an Error Code Capture Register (described in Section 10.8) where they remain until this register is read.

Experiencing significant numbers of such errors may however indicate that corrective action should be taken, so the MCAN2 maintains two error counters – one for reception errors (RXERR) and one for transmission errors (TXERR) – which are automatically incremented whenever an error occurs. Should either counter exceed the value recorded in the Error Warning Limit register, an Error Warning interrupt is generated (if enabled) while if either counter exceeds a count of 127, an Error Passive interrupt is generated (if enabled). An Error Warning interrupt will also be generated if the MCAN2 goes into Bus Off state as a result of the count of transmission errors exceeding 255.

20.4 Sleep Mode

When there is no bus activity and no interrupts are pending, power can be saved by putting the MCAN2 into a Sleep Mode in which XTAL1_IN is turned off. This is selected by setting the Sleep Mode bit in the Mode Register (MOD.4) to ‘1’.

Any of the following events will cause the MCAN2 to ‘wake up’ from Sleep Mode:

- Setting the Sleep Mode bit to ‘0’
- Activity on the CAN bus input (RX0)
- A low on NINT_IN

On waking up, the MCAN2 will generate a Wake-Up Interrupt.

Note: If the MCAN2 is awakened by bus activity, it cannot receive any message until after it has detected a Bus-Free sequence of 11 recessive bits on the bus. You should also note that it is not possible to select Sleep Mode while the MCAN2 is in Reset Mode.

20.5 Register Description

The registers used in the MCAN2 are listed below, with detailed information about the individual registers given in the following sections (referenced in the table). Note: Different Read/Write permissions apply depending on whether the MCAN2 is in Operating Mode or Reset Mode.

Address	Register Name		See Section	Operating Mode	Reset Mode	Comment
00h	MOD	Mode	10.12	Read/Write	Read/Write	
01h	CMR	Command	10.7	Write only	Write only	Returns 00h when read.
02h	SR	Status	10.18	Read only	Read only	
03h	IR	Interrupt	10.10	Read only	Read only	
04h	IER	Interrupt enable	10.11	Read/Write	Read/Write	
05h		*Reserved*		N/A	N/A	Returns 00h when read.
06h	BTR0	Bus Timing 0	10.4	Read only	Read/Write	
07h	BTR1	Bus Timing 1	10.5	Read only	Read/Write	
08h	OCR	Output Control Register	10.13	Read only	Read/Write	
09h		*Reserved*		N/A	N/A	
0Ah		*Reserved*		N/A	N/A	Returns 00h when read.
0Bh	ALC	Arbitration Lost Capture	10.3	Read only	Read only	
0Ch	ECC	Error Code Capture	10.8	Read only	Read only	
0Dh	EWLR	Error Warning Limit	10.9	Read only	Read/Write	
0Eh	RXERR	Receive Error Counter	10.16	Read only	Read/Write	
0Fh	TXERR	Transmit Error Counter	10.20	Read only	Read/Write	
10h	Transmit Buffer	Transmit Frame Information	10.19	Write	(see below)	Read back from 60h.
11h – 1Ch	Transmit Buffer	Transmit Data Information	10.19	Write	"	Read back from 61h – 6Ch.
10h	Receive Window	Receive Frame Information	10.14	Read	"	
11h – 1Ch	Receive Window	Receive Data Information	10.14	Read	"	
10h – 13h	ACR0–3	Acceptance Code Registers 0 – 3	10.1	(see above)	Read/Write	***Note:*** *18h – 1Ch reserved in Reset Mode (return 00h when read)*
14h – 17h	AMR0–3	Acceptance Mask Registers 0 – 3	10.2	"	Read/Write	
1Dh	RMC	Receive Message Counter	10.17	Read only	Read only	
1Eh	RBSA	Receive Buffer Start Address	10.15	Read only	Read/Write	
1Fh	CDR	Clock Divider	0	Read/Write	Read/Write	
20h – 5Fh		Receive FIFO	10.14	Read only	Read/Write	
60h – 6Ch		Transmit Buffer	10.19	Read only	Read only	
6Dh – 7Fh		*Reserved*		N/A	N/A	Return 00h when read.

20.5.1 Acceptance Code Registers (ACR0 – ACR3): ADDRESS 10h – 13h

These 8-bit registers record the bit patterns used by the Acceptance Filter in conjunction with the masks provided by AMR0 – AMR3 in filtering received data.

The way in which these bit patterns are applied depends on whether a single filter or dual filters are being used and on whether the data is in Standard Frame Format (SFF) or Extended Frame Format (EFF).

These registers can only be accessed in Reset Mode.

20.5.2 Acceptance Mask Registers (AMR0 – AMR3): ADDRESS 14h – 17h

These 8-bit registers record the mask patterns applied by the Acceptance Filter in filtering the data received. '0's in these registers identify the bits of the incoming data bytes that are required to match the bit values in the corresponding Acceptance Code Registers. '1's mark individual bits as 'don't care'.

The bits of the incoming data picked out by these masks depends on whether a single filter or dual filters are being used and on whether the data is in Standard Frame Format (SFF) or Extended Frame Format (EFF).

The registers can only be accessed in Reset Mode.

20.5.3 Arbitration Lost Capture Register (ALC): ADDRESS 0Bh

This read-only register records the bit position at which arbitration was lost.

When bus arbitration lost, an Arbitration Lost Interrupt is generated (if enabled) and the current position of the Bit Processor is captured into this Arbitration Lost Capture Register. The contents of this register are then maintained until the register has been read by the user's software. The capture mechanism is then activated again.

BIT	COMMENT
ALC[7:5]	*Reserved. Return zero when read.*
ALC[4:0]	See table below.

ALC[4:0]	DECIMAL VALUE	FUNCTION
0 0 0 0 0	00	Arbitration lost in 1st bit of identifier (ID.28).
0 0 0 0 1	01	Arbitration lost in 2nd bit of identifier (ID.27).
0 0 0 1 0	02	Arbitration lost in 3rd bit of identifier (ID.26).
0 0 0 1 1	03	Arbitration lost in 4th bit of identifier (ID.25).
0 0 1 0 0	04	Arbitration lost in 5th bit of identifier (ID.24).
0 0 1 0 1	05	Arbitration lost in 6th bit of identifier (ID.23).
0 0 1 1 0	06	Arbitration lost in 7th bit of identifier (ID.22).
0 0 1 1 1	07	Arbitration lost in 8th bit of identifier (ID.21).
0 1 0 0 0	08	Arbitration lost in 9th bit of identifier (ID.20).
0 1 0 0 1	09	Arbitration lost in 10th bit of identifier (ID.19).
0 1 0 1 0	10	Arbitration lost in 11th bit of identifier (ID.18).
0 1 0 1 1	11	Arbitration lost in SRTR bit [1].
0 1 1 0 0	12	Arbitration lost in IDE bit.

20.5.4 Bus Timing Register 0 (BTR0): ADDRESS 06h

Bus Timing Register 0 defines the values of the Synchronization Jump Width (SJW) and the Baud Rate Prescaler (BRP).

BTR0.7	BTR0.6	BTR0.5	BTR0.4	BTR0.3	BTR0.2	BTR0.1	BTR0.0
SJW.1	SJW.0	BRP.5	BRP.4	BRP.3	BRP.2	BRP.1	BRP.0

SYNCHRONIZATION JUMP WIDTH (SJW): BTR0[7:6].

The Synchronization Jump Width defines the maximum number of time quanta by which a bit period may be shortened or lengthened in attempting to re-synchronize on the relevant signal edge (recessive to dominant) of the current transmission.

BAUD RATE PRESCALER (BRP): BTR0[5:0]

The Baud Rate Prescaler defines the 'time quantum' TQ of the CAN clock as a multiple of the XTAL1

ALC[4:0]	DECIMAL VALUE	FUNCTION
0 1 1 0 1	13	Arbitration lost in 12th bit of identifier (ID.17)
0 1 1 1 0	14	Arbitration lost in 13th bit of identifier (ID.16)
0 1 1 1 1	15	Arbitration lost in 14th bit of identifier (ID.15)
1 0 0 0 0	16	Arbitration lost in 15th bit of identifier (ID.14)
1 0 0 0 1	17	Arbitration lost in 16th bit of identifier (ID.13)
1 0 0 1 0	18	Arbitration lost in 17th bit of identifier (ID.12)
1 0 0 1 1	19	Arbitration lost in 18th bit of identifier (ID.11)
1 0 1 0 0	20	Arbitration lost in 19th bit of identifier (ID.10)
1 0 1 0 1	21	Arbitration lost in 20th bit of identifier (ID.9)
1 0 1 1 0	22	Arbitration lost in 21st bit of identifier (ID.8)
1 0 1 1 1	23	Arbitration lost in 22nd bit of identifier (ID.7)
1 1 0 0 0	24	Arbitration lost in 23rd bit of identifier (ID.6)
1 1 0 0 1	25	Arbitration lost in 24th bit of identifier (ID.5)
1 1 0 1 0	26	Arbitration lost in 25th bit of identifier (ID.4)
1 1 0 1 1	27	Arbitration lost in 26th bit of identifier (ID.3)
1 1 1 0 0	28	Arbitration lost in 27th bit of identifier (ID.2)
1 1 1 0 1	29	Arbitration lost in 28th bit of identifier (ID.1)
1 1 1 1 0	30	Arbitration lost in 29th bit of identifier (ID.0)
1 1 1 1 1	31	Arbitration lost in RTR bit

Extended Frame Format messages only

input clock period. The time quantum of the CAN clock is given by:

TQ =2 x t_{clk} x (32 x BRP.5 + 16 x BRP.4 + 8 x BRP.3 + 4 x BRP.2 + 2 x BRP.1 + BRP.0 + 1) where t_{clk} = time period of the XTAL1 frequency = 1/f_{xtal1}

20.5.5 Bus Timing Register 1 (BTR1): ADDRESS 07h

Bus Timing Register 1 defines the length of the bit period, the location of the sample point and the number of samples to be taken at each sample point.

BTR1.7	BTR1.6	BTR1.5	BTR1.4	BTR1.3	BTR1.2	BTR1.1	BTR1.0
SAM	TSEG2.2	TSEG2.1	TSEG2.0	TSEG1.3	TSEG1.2	TSEG1.1	TSEG1.0

SAMPLING (SAM): BTR1.7

BIT	VALUE	FUNCTION
SAM	1	The bus will be sampled three times. (This is recommended for low/medium speed buses (class A or B).)
	0	The bus will be sampled once. (This is recommended for high speed buses (SAE class C).)

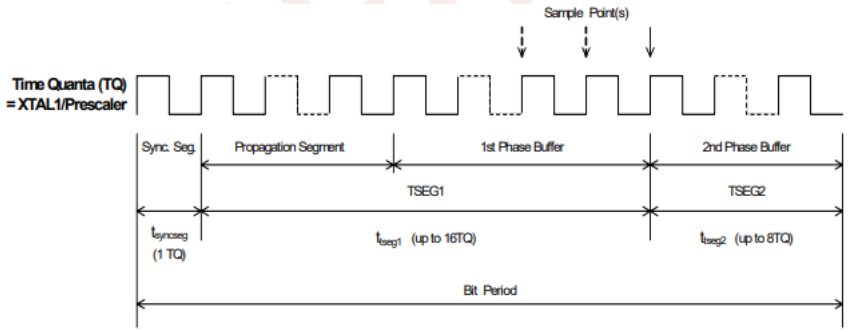
TSEG1 AND TSEG2: BTR1[3:0], BTR1[6:4]

TSEG1 and TSEG2 define the length of the bit period by giving the number of time quanta up to and after the point(s) at which incoming data will be sampled. In terms of TSEG1 and TSEG2, the parameters tsyncseg, ttseg1 and ttseg2 shown in the diagram are:

$t_{syncseg}$ =1 x TQ

t_{tseg1} = TQ x (8 x TSEG1.3 + 4 x TSEG1.2 + 2 x TSEG1.1 + TSEG1.0 + 1)

t_{tseg2}= TQ x (4 x TSEG2.2 + 2 x TSEG2.1 + TSEG2.0 + 1)



20.5.6 Clcck Divider Register (CDR): ADDRESS 1Fh

The Clock Divider Register controls the CLKOUT signal. The default state of the register after a hardware reset is 11000000 (divide by 2 and CLKOUT signal enabled). The register is not changed by a software reset.

CDR.7	CDR.6	CDR.5	CDR.4	CDR.3	CDR.2	CDR.1	CDR.0
1	1	0	0	Clock Off	CDR.2	CDR.1	CDR.0

CDR[2:0]

The bits CD.2 to CD.0 define the frequency at the external CLKOUT pin as shown in the following table (fosc is the frequency of the external oscillator (XTAL1)). These bits may be accessed from either Reset Mode or Operating Mode.

CD[2:0]	CLKOUT FREQUENCY	CD[2:0]	CLKOUT FREQUENCY
0 0 0	$f_{osc}/2$	1 0 0	$f_{osc}/10$
0 0 1	$f_{osc}/4$	1 0 1	$f_{osc}/12$
0 1 0	$f_{osc}/6$	1 1 0	$f_{osc}/14$
0 1 1	$f_{osc}/8$	1 1 1	f_{osc}

CLOCKOFF (CDR.3)

Setting this bit allows the external CLKOUT signal to be disabled.

20.5.7 Command Register (CMR): ADDRESS 01h

Setting one or more bits within the Command Register initiates an action within the transfer layer of the CAN controller.

Note: This register is write only. When read, all bits return '0'. You should also note that there must be at least one external clock cycle between consecutive commands.

BIT	SYMBOL	NAME	FUNCTION
CMR[7:5]	–	–	*Reserved*
CMR.4	SRR	Self Reception Request	Set to '1' when a message is to be transmitted and received simultaneously.
CMR.3	CDO	Clear Data Overrun	Set to '1' to clear the data overrun condition signaled by the Data Overrun Status bit (SR.1). *Note:* No further Data Overrun Interrupt will be generated while the Data Overrun Status bit remains set.
CMR.2	RRB	Release Receive Buffer	Set to '1' to release the Receive Buffer – see Section 5.
CMR.1	AT	Abort Transmission	Set to '1' to cancel the next transmission request, provided this is not already in progress.
CMR.0	TR	Transmission Request	Set to '1' when a message is to be transmitted – see Section 4.

20.5.8 Error Code Capture Register (ECC): ADDRESS 0CH

This read-only register may be used to obtain detailed information about the type and location of bus errors.

When a bus error occurs, a Bus Error Interrupt is generated (if enabled) and the current bit position of the Bit Processor is captured into this Error Code Capture Register. The contents of this register are then maintained until the register has been read by the user's software. The capture mechanism is then activated again.

BIT	NAME
ECC[7:6]	Error Code
ECC.5	Direction
ECC[4:0]	Segment Code

20.5.9 Error Warning Limit Register (EWLR): ADDRESS 0Dh

This register defines the number of errors after which an Error Warning Interrupt should be generated (if enabled).

This register is read only in Operating Mode but may be written in Reset Mode. You should note that changes made within Reset Mode are only put into effect on return to Operating Mode.

The default value of this register (after hardware reset) is 0110000 (i.e. 96). An error count of this level suggests a significantly disturbed bus, the causes of which should be investigated.

EWLR.7	EWLR.6	EWLR.5	EWLR.4	EWLR.3	EWLR.2	EWLR.1	EWLR.0
EWL.7	EWL.6	EWL.5	EWL.4	EWL.3	EWL.2	EWL.1	EWL.0

20.5.10 Interrupt Register (IR): ADDRESS 03h

The Interrupt Register allows the source of an interrupt to be identified. When one or more bits of this register are set, the MCAN2 sends an interrupt to the CPU. The way the different interrupts should be handled.

Note: The Interrupt Register is read-only. After the register has been read by the CPU, all except the Receive Interrupt bit are reset.

BIT	SYMBOL	NAME	FUNCTION
IR.7	BEI	Bus Error Interrupt	Set when the MCAN2 detects an error on the CAN-bus – provided the BEIE bit (IER.7) is set within the Interrupt Enable Register.
IR.6	ALI	Arbitration Lost Interrupt	Set when the MCAN2 loses arbitration and becomes a receiver – provided the ALIE bit (IER.6) is set within the Interrupt Enable Register.
IR.5	EPI	Error Passive Interrupt	Set when the MCAN2 re-enters error active state after being in error passive state or when at least one error counter exceeds the protocol-defined level of 127 – provided the EPIE bit (IER.5) is set within the Interrupt Enable Register.
IR.4	WUI	Wake-Up Interrupt	Set when bus activity is detected while the CAN controller is sleeping – provided the WUIE bit (IER.4) is set within the Interrupt Enable Register. [1]
IR.3	DOI	Data Overrun Interrupt	Set on a '0-to-1' transition of the Data Overrun Status bit (SR.1) – provided the DOIE bit (IER.3) is set within the Interrupt Enable Register.
IR.2	EI	Error Warning Interrupt	Set on every change (set or clear) of either the Bus Status or Error Status bits (SR.7,SR.6) – provided the EIE bit (IER.2) is set within the Interrupt Enable Register.
IR.1	TI	Transmit Interrupt	Set whenever the Transmit Buffer Status (SR.2) changes from '0-to-1' (released) – provided the TIE bit (IER.1) is set within the Interrupt Enable Register.
IR.0	RI	Receive Interrupt [2]	Set whenever the Receive Buffer contains one or more messages – provided the RIE bit (IER.0) is set within the Interrupt Enable Register. Cleared when the release Receive Buffer command (CMR. 2) is issued provided there is no further data to read in the Receive Buffer.

20.5.11 Interrupt Enable Register(IER): ADDRESS 04h

This read/write register is used to select the events that are indicated to the CPU through an interrupt being generated.

BIT	SYMBOL	NAME	FUNCTION
IER.7	BEIE	Bus Error Interrupt Enable	When set to '1', an interrupt will be generated when a bus error has been detected. When set to '0', the interrupt is disabled.
IER.6	ALIE	Arbitration Lost Interrupt Enable	When set to '1', an interrupt will be generated when the MCAN2 loses arbitration. When set to '0', the interrupt is disabled.
IER.5	EPIE	Error Passive Interrupt Enable	When set to '1', an interrupt will be generated when the error status of the MCAN2 changes from error active to error passive or vice versa. When set to '0', the interrupt is disabled.
IER.4	WUIE	Wake-Up Interrupt Enable	When set to '1', an interrupt will be generated when the sleeping MCAN2 wakes up. When set to '0', the interrupt is disabled.
IER.3	DOIE	Data Overrun Interrupt Enable	When set to '1', an interrupt will be generated when the Data Overrun Status bit (SR.1) is set. When set to '0', the interrupt is disabled.
IER.2	EIE	Error Warning Interrupt Enable	When set to '1', an interrupt will be generated when the bus status or error status bits (SR.7, SR.6) change. When set to '0', the interrupt is disabled.
IER.1	TIE	Transmit Interrupt Enable	When set to '1', an interrupt will be generated when a message has been successfully transmitted or the Transmit Buffer is accessible again. When set to '0', the interrupt is disabled.
IER.0	RIE	Receive Interrupt Enable[1]	When set to '1', an interrupt will be generated when the Receive Buffer Status (SR.0) goes from '0' to '1' ('full'). When set to '0', the interrupt is disabled.

20.5.12 Mode Register (MOD): ADDRESS 00h

This read/write register is used to set the behavior of the CAN controller.

BIT	SYMBOL	NAME	VALUE	FUNCTION
MOD[7:5]	–	–	–	*Reserved. Return zero when read.*
MOD.4	SM	Sleep Mode *(Can only be written in Operating Mode)*	1	Sleep. The MCAN2 enters its Sleep mode provided no CAN interrupt is pending and there is no bus activity. (If there is bus activity or an interrupt is pending, the Wake-Up procedure is executed.)
			0	Wake-up (normal operation). If sleeping, the MCAN2 wakes up.
MOD.3	AFM	Acceptance Filter Mode[1]	1	Single Filter. Receive data filtered using one 4-byte filter.
			0	Dual Filter. Receive data filtered using two shorter filters.
MOD.2	STM	Self Test Mode[1]	1	Self Test enabled. In this mode, a full node test is possible without any other active node on the bus using the self reception request command. The MCAN2 will perform a successful transmission, even if no acknowledge is received.
			0	Normal operation. An acknowledge is required for successful transmission.
MOD.1	LOM	Listen Only Mode[1]	1	Listen Only enabled. In this mode, the MCAN2 does not send an acknowledge to the CAN bus, even when a message is received successfully.
			0	Normal operation. The error counters are stopped at the current value.
MOD.0	RM	Reset Mode	1	Reset Mode selected. Any message currently being transmitted or received is aborted and Reset Mode is entered.
			0	Normal operation. The MCAN2 returns to Operating Mode on the '1-to-0' transition of this bit.

20.5.13 Output Control Register (OCR): ADDRESS 08h

The Output Control Register allows the selection of two possible output driver configurations: 'Normal Output' and 'Clock Output'.

In Normal Output Mode, the bit sequence (TXD) is sent to TX0 with the inverse sent to TX1.

In Clock Output Mode, the bit sequence is output on the TX0 signal as in normal output mode but the data stream on TX1 is replaced by a copy of the Transmit clock (TXCLK), the rising edge of which marks the beginning of a bit period. The pulse width of this clock is one Time Quantum (TQ).

Note: The additional driver configurations available in the SJA1000 through this register are not supported by the MCAN2.

OCR.7	OCR.6	OCR.5	OCR.4	OCR.3	OCR.2	OCR.1	OCR.0
Reserved	*Reserved*	*Reserved*	*Reserved*	*Reserved*	*Reserved*	OCMODE1	OCMODE0

Interpretation of OCMODE bits

OCMODE1	OCMODE0	DESCRIPTION
0	X	*Reserved*
1	0	Normal Output Mode
1	1	Clock Output Mode

Note: The Output Control Register may only be written in Reset Mode. In Operating Mode, this register is read only. The Reserved bits return '0' when read.

20.5.14 Receive Buffer (10h – 1Ch)

The Receive Buffer provides the window through which the CPU accesses the Receive FIFO. Like the Transmit Buffer, the Receive Buffer has a length of 13 bytes (enough to accommodate one Receive message of up to eight data bytes).

Read-only access to the Receive Buffer is provided in Operating Mode using CAN addresses 10h – 1Ch

The layout of the Receive Buffer is similar to the Transmit Buffer described in the previous section. Indeed, the configuration used was chosen specifically to be compatible with the layout of the Transmit Buffer. Again, it is important to distinguish between Standard Frame Format (SFF) messages and the Extended Frame Format (EFF) messages.

Receive Buffer Layout

The Receive Buffer is subdivided into descriptor and data fields. The first byte of the descriptor field holds frame information. It describes the frame format (SFF or EFF), specifies remote or data frame and gives the data length. This is then followed by either two identifier bytes for SFF or four bytes for EFF messages. The data field contains up to eight data bytes.

Standard Frame Format (SFF)		Extended Frame Format (EFF)	
CAN Address	Field	CAN Address	Field
10h	RX Frame Information	10h	RX Frame Information
11h	RX Identifier 1	11h	RX Identifier 1
12h	RX Identifier 2	12h	RX Identifier 2
13h	RX Data Byte 1	13h	RX Identifier 3
14h	RX Data Byte 2	14h	RX Identifier 4
15h	RX Data Byte 3	15h	RX Data Byte 1
16h	RX Data Byte 4	16h	RX Data Byte 2
17h	RX Data Byte 5	17h	RX Data Byte 3
18h	RX Data Byte 6	18h	RX Data Byte 4
19h	RX Data Byte 7	19h	RX Data Byte 5
1Ah	RX Data Byte 8	1Ah	RX Data Byte 6
1Bh	(Unused)	1Bh	RX Data Byte 7
1Ch	(Unused)	1Ch	RX Data Byte 8

20.5.15 Receive Buffer Start Address (RBSA): ADDRESS 1Eh

The Receive Buffer Start Address register records the current location of the RX FIFO Read Pointer within the 64-byte Receive FIFO as a value between 0 and 63. Location 0 maps to CAN address 20h: Location 63 maps to CAN address 5Fh.

This register is reset to 00h by a hardware reset but is left unchanged by a software reset (which also does not change the FIFO contents). However, the software reset sets the RX FIFO Write Pointer to the value of the RX FIFO Read Pointer with the result that the data currently accessed by the Receive Buffer following a software reset will be overwritten by the next message to be recorded in the Receive FIFO.

Note: It is only possible to write to this register in Reset Mode.

RBSA.7	RBSA.6	RBSA.5	RBSA.4	RBSA.3	RBSA.2	RBSA.1	RBSA.0
–	–	RBSA.5	RBSA.4	RBSA.3	RBSA.2	RBSA.1	RBSA.0

20.5.16 Receive Error Counter Register (RXERR): ADDRESS 0Eh

The Receive Error Counter Register records the current value of the Receive Error Counter. This counter is incremented when errors are experienced in the Receive bit stream and decremented when messages are received without error, in line with the rules given in the CAN 2.0 specification. Together with the associated Transmit Error Counter (see Section 10.20), it provides an indication of the quality of transmission being experienced on the CAN bus.

An outline of the rules by which the counter is incremented and decremented is given in the table below. For full details, you should refer to the CAN 2.0 specification.

Two levels of the counter trigger specific events.

• When the counter reaches the level set in the Error Warning Limit register (see Section 10.9), an Error Warning Interrupt is generated (if enabled) unless this has previously been triggered by the Transmit Error Counter.

• When the counter goes over 127, the device is put into Error Passive state in accordance with the CAN 2.0 specification(unless previously triggered by the Transmit Error Counter) and an Active error is sent. An Error Passive Interrupt is also generated (if enabled).

After a hardware reset or when a Bus Off event occurs (see Transmit Error Counter – see Section 10.20), the counter is automatically set to ‘0’.

The register is read only in Operating Mode but may be written in Reset Mode. You should note, however, that writing to this register has no effect when the MCAN2 is in Bus Off state and that any change made within Reset Mode will in any case only come into effect on return to Operating Mode.

ERROR EVENT	ACTION TAKEN
Receiver detects an error	RXERR incremented by 1
Receiver detects dominant bit as the first bit after sending an Error flag	RXERR incremented by 8
Receiver detects a bit error while sending an Active error or an Overload error	RXERR incremented by 8
Message successfully received	RXERR decremented by 1
Message is received successfully when the count had previously been above the Error Passive trigger level of 127	RXERR automatically set to a value between 119 and 127
14th consecutive dominant bit received after sending an Active error or an Overload error	RXERR incremented by 8 and TXERR incremented by 8 both at this point and after each additional sequence of 8 consecutive dominant bits
8th consecutive dominant bit received after sending a Passive error	
Transmitter sends an error	TXERR incremented by 8
Transmitter detects a bit error while sending an Active error or an Overload error	TXERR incremented by 8
Transmitter successfully transmits message	TXERR decremented by 1

20.5.17 Receive Message Counter (RMC): ADDRESS 1Dh

The Receive Message Counter register records the number of messages currently available in the Receive FIFO. It is automatically incremented by each Receive event and decremented by each Release Receive Buffer command. It is available for Read only access in both Operating Mode and Reset Mode.

The register is reset to 00h by either a hardware or a software reset.

RMC.7	RMC.6	RMC.5	RMC.4	RMC.3	RMC.2	RMC.1	RMC.0
0	0	0	RMC.4	RMC.3	RMC.2	RMC.1	RMC.0

20.5.18 Status Register(SR): ADDRESS 02h

This read-only register reflects the status of the MCAN2 controller.

BIT	SYMBOL	NAME	VALUE	FUNCTION
SR.7	BS	Bus Status	1	The MCAN2 is in 'Bus Off' state and is not involved in bus activities.
			0	The MCAN2 is involved in bus activities.
SR.6	ES	Error Status	1	At least one of the error counters has reached or exceeded the CPU warning limit defined by the Error Warning Limit Register (EWL).
			0	Both error counters are below the warning limit.
SR.5	TS	Transmit Status [1]	1	The MCAN2 is in the process of transmitting a message.
			0	No message is being transmitted.
SR.4	RS	Receive Status [1]	1	The MCAN2 is in the process of receiving a message.
			0	Nothing is currently being received.
SR.3	TCS	Transmission Complete Status	1	The last requested transmission has been successfully completed.
			0	The last requested transmission has not been completed yet.
SR.2	TBS	Transmit Buffer Status	1	Transmit Buffer Released. The CPU may write a message to the Transmit Buffer.
			0	Transmit Buffer Locked. The CPU cannot access the Transmit Buffer because a message is either waiting for transmission or is in the process of being transmitted.
SR.1	DOS	Data Overrun Status	1	Data Overrun. A message has been lost because there was not enough space for that message in the Receive FIFO. [2]
			0	No data overrun has occurred since the last Clear Data Overrun command was given.
SR.0	RBS	Receive Buffer Status	1	Receive Buffer Full. One or more complete messages are available to be read from the Receive FIFO via the Receive Buffer.
			0	Receive Buffer Empty. No message currently available to be read.

20.5.19 Transmit Buffer (Write: 10h – 1Ch; Read: 60h – 6Ch)

The Transmit Buffer has a length of 13 bytes. It accommodates one Transmit message of up to eight data bytes.

Access to the Transmit Buffer in Operating Mode is write-only and is provided using CAN addresses 10h – 1Ch.

The global layout of the Transmit Buffer is shown below. It is important to distinguish between Standard Frame Format (SFF) messages and the Extended Frame Format (EFF) messages.

Note: Read access to the Transmit Buffer is possible using CAN addresses 60h – 6Ch.

Transmit Buffer Layout

The Transmit Buffer is subdivided into descriptor and data fields. The first byte of the descriptor field holds frame information. It describes the frame format (SFF or EFF), remote or data frame and the data length. This is then followed by either two identifier bytes for SFF or four bytes for EFF messages. The data field contains up to eight data bytes.

Standard Frame Format (SFF)		Extended Frame Format (EFF)	
CAN Address	Field	CAN Address	Field
10h	TX Frame Information	10h	TX Frame Information
11h	TX Identifier 1	11h	TX Identifier 1
12h	TX Identifier 2	12h	TX Identifier 2
13h	TX Data Byte 1	13h	TX Identifier 3
14h	TX Data Byte 2	14h	TX Identifier 4
15h	TX Data Byte 3	15h	TX Data Byte 1
16h	TX Data Byte 4	16h	TX Data Byte 2
17h	TX Data Byte 5	17h	TX Data Byte 3
18h	TX Data Byte 6	18h	TX Data Byte 4
19h	TX Data Byte 7	19h	TX Data Byte 5
1Ah	TX Data Byte 8	1Ah	TX Data Byte 6
1Bh	(Unused)	1Bh	TX Data Byte 7
1Ch	(Unused)	1Ch	TX Data Byte 8

20.5.20 Transmit Error Counter Register (TXERR): ADDRESS 0Fh

The Transmit Error Counter Register records the current value of the Transmit Error Counter. This counter is incremented when Transmission errors are experienced and decremented when messages are transmitted without error, in line with the rules given in the CAN 2.0 specification. Together with the associated Receive Error Counter (see Section 10.16), it provides an indication of the quality of transmission being experienced on the CAN bus.

An outline of the rules by which the counter is incremented and decremented is given in the table in Section 10.16. For full details, you should refer to the CAN 2.0 specification.

Three levels of the counter trigger specific events.

• When the counter reaches the level set in the Error Warning Limit register (see Section 10.9), an Error Warning Interrupt is generated (if enabled) unless this has previously been triggered by the Receive Error Counter.

• When the counter goes over 127, the device is put into Error Passive state in accordance with the CAN 2.0 specification (unless previously triggered by the Receive Error Counter), an Active error is sent and an Error Passive Interrupt is generated (if enabled).

• When the counter goes over 255, the device is put into Bus Off state in accordance with the CAN 2.0 specification and is automatically put into Reset mode (except during start-up when there is only one node on the CAN bus). An Error Warning Interrupt is also generated (if enabled).

After a hardware reset, the Transmit Error Counter is automatically set to '0'.

After a 'Bus Off' event, the register is initialized to 127 in order to count the minimum protocol-defined time before the MCAN2 can take part in further transmission on the CAN bus (128 occurrences of the 'Bus-Free' sequence of 11 consecutive recessive bits). Reading the Transmit Error Counter during this time will give the status of the Bus Off recovery. Note: If the Reset Mode is re-entered before the Bus Off recovery has been completed (TXERR > 0), Bus Off will stay active with TXERR frozen until the MCAN2 is taken back into Operating Mode.

It is possible to write to this register but only in Reset Mode. In Operating Mode, this register appears as read only memory to the CPU.

While in Bus Off state, writing a value in the range from 0 to 254 to TXERR clears the Bus Off flag. The MCAN2 will then wait for just one Bus Free sequence after the Reset Mode has been cleared.

Writing 255 to TXERR in Reset Mode initiates a CPU-driven Bus Off event. No error or bus status change happens in response to the new TXERR value until the MCAN2 is taken back into Operating Mode when a Bus Off event will be performed exactly as if it had been forced by a bus error. This means Reset Mode is entered again, the Transmit Error Counter is initialized to 127, the Receive counter is cleared and the relevant Status and Interrupt register bits are set. Clearing Reset Mode now will perform the protocol-defined Bus Off recovery sequence (waiting for 128 occurrences of the bus-free signal).

21 Flash-SPI control

21.1 Overview

21.1.1 Characteristics of this spi controller

The FlashSpi module is an spi master controller that can be configured through AHB bus. It is suitable for wifi chips based on AHB bus architecture like S902, and is used to read and write off-chip flash chips such as S25FL116K of Spansion, W25Q20CL of Winbond or similar off-chip flash chips that provide spi slave interface. This module has the following features:

1) provide a set of AHB slave interfaces, a set of DMA Single request interfaces and an interrupt request in the chip, and provide spi master interfaces outside the chip.

2) It can provide spi clock with the fastest half of the system clock frequency.

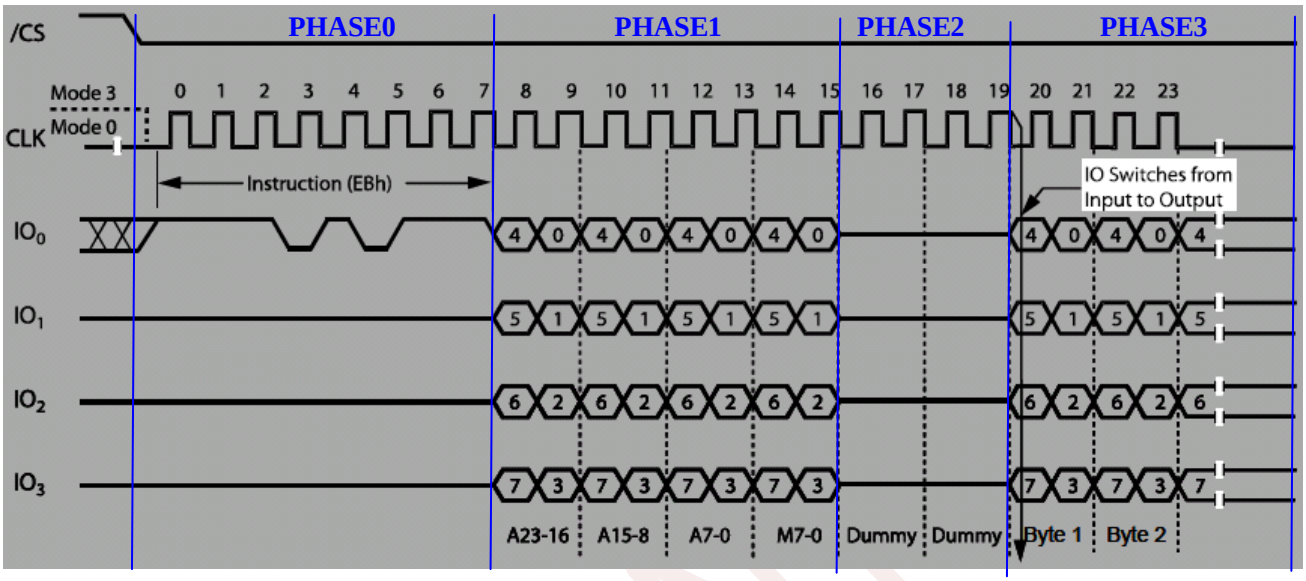
3) For a communication, it starts automatically after the register is configured. After the communication is completely finished, the register flag bit is displayed, and the completion interrupt can also be generated.

4) One communication can contain up to 8 independently configurable phase, which is enough to flexibly correspond to various situations contained in one spi communication of flash chip.

5) The whole module adopts synchronous clock design, and all signals belong to CLK clock domain.

21.1.2 The concept of PHASE

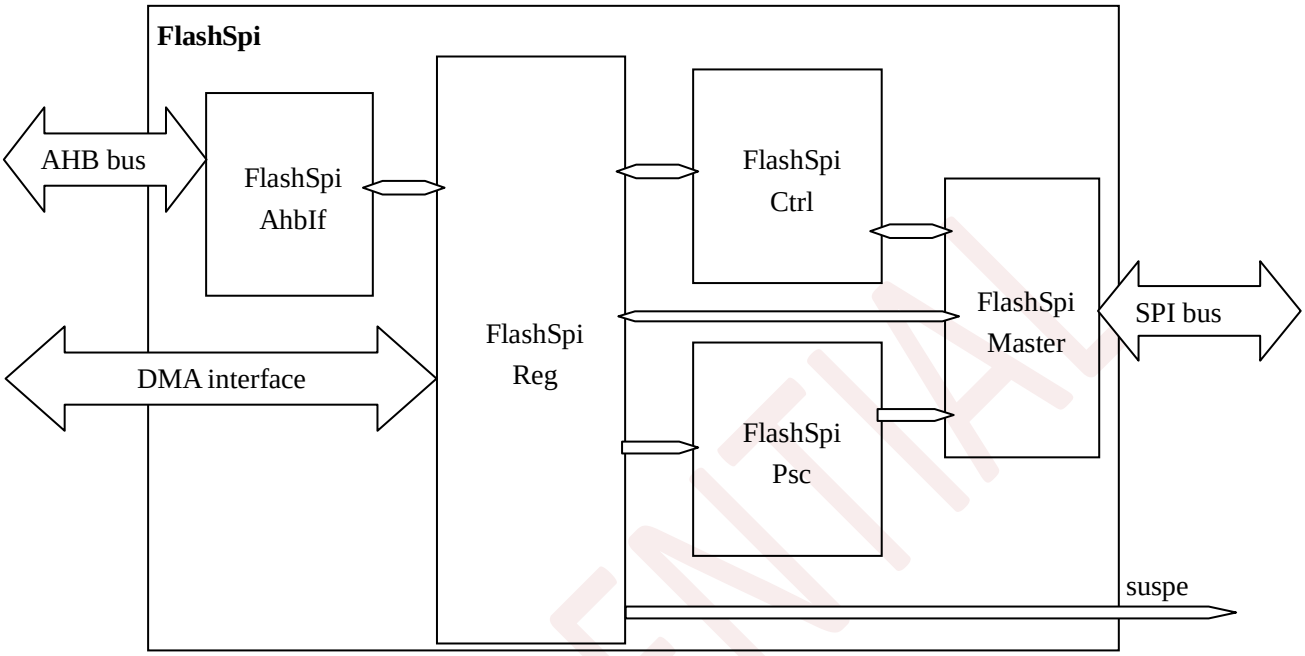
The following figure is a typical timing diagram of reading FLASH data through SPI.



The process from CSn falling to CSn rising is called an SPI communication. In the SPI communication shown above, Instruction, A, M and Dummy should be sent in turn, and then data should be received. And Instruction only uses one line, while the subsequent processes all use four lines. Aiming at this kind of timing, this module puts forward the concept of PHASE. A communication can contain up to 8 PHASEs, and each phase can independently set parameters such as read-write operation, 1/2/4 line mode, communication data volume, etc. After the first SPI communication starts, this module will finish the tasks set by each PHASE in the order of PHASE0→PHASE7, and then end the SPI communication. For example, for the timing of the above figure, it can be divided into 4 PHASE.

21.1.3 Module block diagram

The following figure is a block diagram of this spi controller:



FlashSpiAhbIf: an AHB Slave interface controller, which is responsible for converting the AHB bus signal into the internal read-write signal of the module and interacting with the register module FlashSpiReg to complete the read-write operation.

Flaspireg: register module of SPI controller, in which all registers are located. At the same time, the generation and processing of DMA interface signals and the generation of interrupts are also in this module.

Flaspictrl: the core control module of SPI controller. When it is detected that the SPI_START bit in FlashSpiReg is written as 1, the phase set in the register is analyzed to generate control and data for FlashSpiDataPump.

FlashSpiPsc：spi sck clock controller, used to control the frequency of sck clock.

FlashSpimaster: Spimaster interface controller, which controls the spi bus to send and receive according to the control signal sent by FlashSpiCtrl.

21.1.4 Top port

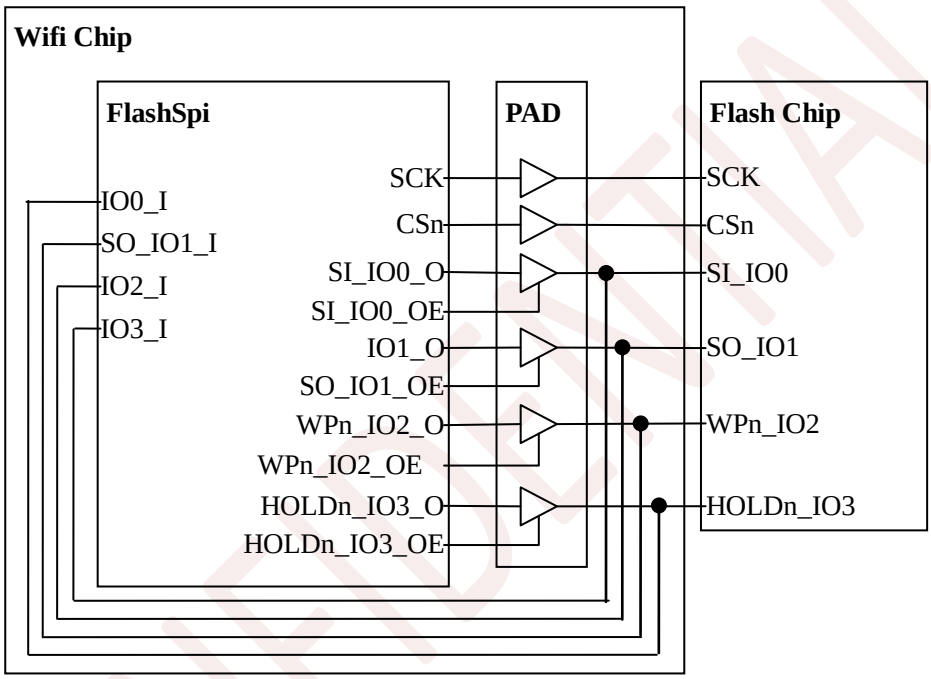
The following is a list of top-level ports and functions of this module:

Signal name	direction	bit wide	Connecting objects	explain
CLK	I	1	ClockGen	Module clock
RST_n	I	1	ResetGen	Module reset
HSEL	I	1	BusMatrix	AHB Slave bus signal
HWRITE	I	1	BusMatrix	AHB Slave bus signal
HADDR	I	32	BusMatrix	AHB Slave bus signal
HTRANS	I	2	BusMatrix	AHB Slave bus signal
HSIZE	I	3	BusMatrix	AHB Slave bus signal
HWDATA	I	32	BusMatrix	AHB Slave bus signal
HREADYIN	I	1	BusMatrix	AHB Slave bus signal
DMA_TX_SREQ_CLR	I	1	Dma	Clear TX_DMA request signal
DMA_RX_SREQ_CLR	I	1	Dma	Clear RX_DMA request signal
IO0_I	I	1	Pad	Spiio0 input of bus
SO_IO1_I	I	1	Pad	Spiso _ io1 input of bus
IO2_I	I	1	Pad	Spiio2 input of bus
IO3_I	I	1	Pad	Spiio3 input of bus
HREADYOUT	O	1	BusMatrix	AHB Slave bus signal
HRDATA	O	32	BusMatrix	AHB Slave bus signal
HRESP	O	2	BusMatrix	AHB Slave bus signal
DMA_TX_SREQ	O	1	Dma	TX_DMA request signal
DMA_RX_SREQ	O	1	Dma	RX_DMA request signal
SPI_DONE_INT	O	1	Cpu	SPI completion interrupt
SCK	O	1	Pad	Spisck signal of bus
CSn	O	1	Pad	Spicsn bus signal
SI_IO0_O	O	1	Pad	SPI bus SI_IO0 output data
SI_IO0_OE	O	1	Pad	SPI bus SI_IO0 output enable
IO1_O	O	1	Pad	Spiio1 bus output data
SO_IO1_OE	O	1	Pad	Spiio1 output enable for bus
WPn_IO2_O	O	1	Pad	Spiwpn _ io2 bus output data
WPn_IO2_OE	O	1	Pad	Spiwpn _ io2 bus output enable
HOLDn_IO3_O	O	1	Pad	Spiholdn _ io3 bus output data
HOLDn_IO3_OE	O	1	Pad	Spiholdn _ io3 bus output enable

21.2 Instructions for use of the module

21.2.1 System integration method

The integration of AHB bus, DMA interface and interrupt interface is relatively simple, which will not be described here. The integration of SPI bus should be equivalent to the following logic:

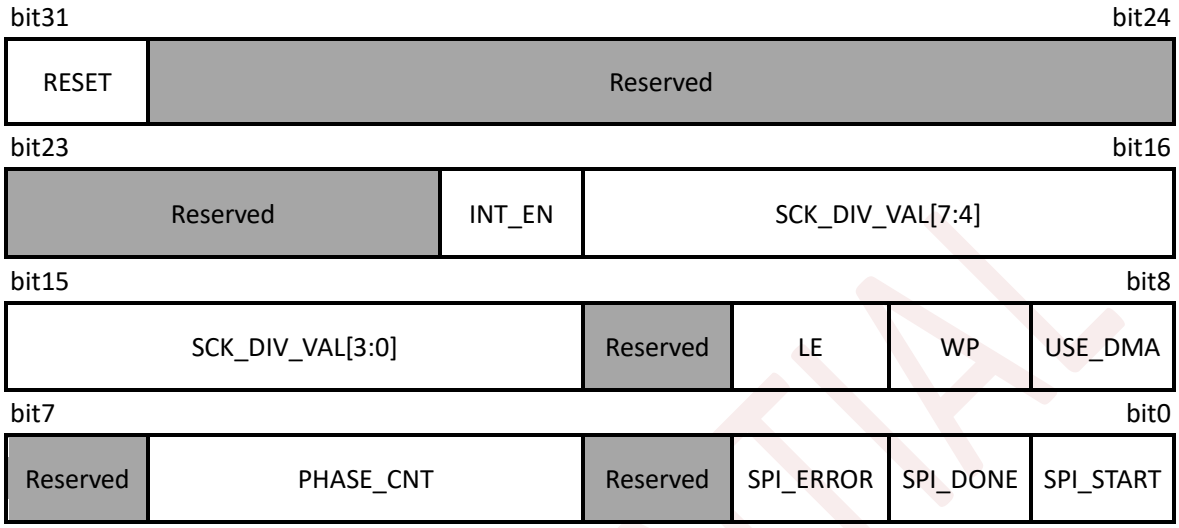


In addition, in order to prevent the input from floating, it is best to add a pull up resistor between Wifi Chip and Flash Chip.

21.2.2 register description

SPCR register (address: BASE_ADDR+8'h00)

SPCR is the global control register of SPI communication, and its bit configuration is as follows:

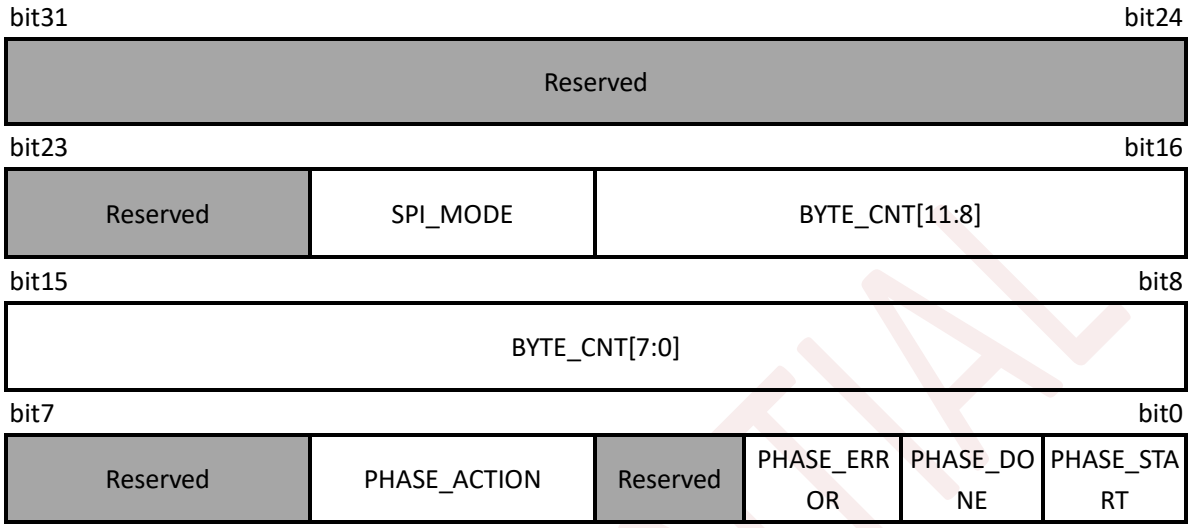


The function of each bit is defined in the following table:

bit	Bitname	initial value	read and write	describe
31	RESET	1'b0	W/R	Software reset of SPI controller. 1, all registers and all internal circuits except this bit are reset.
30-21	Reserved	10'h000	R	Keep.
20	INT_EN	1'b0	W/R	Interrupt enable. When set to 1, if SPI_DONE is 1, an interrupt will be sent to the CPU.
19-12	SCK_DIV_VAL	8'h08	W/R	SPI communication frequency setting. SPI communication rate is the frequency division of this value of the system clock. Even number must be filled in. Fill in 8'h00 to represent 256 frequency division.
11	Reserved	1'b0	R	Keep.
10	LE	1'b0	W/R	Small start. Because the data register is 32 bits, and SPI sends one byte at a time, when LE is set to 1, SPI will first send and receive [7:0] bits of the data register, then [15:8] until [31: 24]; When LE is set to 0, the order is reversed.
9	WP	1'b1	W/R	When SPI communication is in single or dual mode, WPn is valid when WP is 1, that is, 0; WPn is 1 when WP is 0.
8	USE_DMA	1'b0	W/R	Use DMA transfer for data of the last phase. Note: If there is only one phase in one spi communication, it is forbidden to set USE_DMA to 1.
7	Reserved	1'b0	R	Keep.
6-4	PHASE_CNT	3'h0	W/R	Number of PHASE included in one spi communication. 0: contains 1 phase. 1: contains 2 phase. …… 7: contains 8 phase.
3	Reserved	1'b0	R	Keep.
2	SPI_ERROR	1'b0	W0/R	When a communication is over, if any of the phase is wrong, this bit is set to one. Clear condition: software writes 0, or software writes SPI_START to 1 (that is, it is automatically cleared when the next communication starts).
1	SPI_DONE	1'b0	W0/R	SPI transmission and DMA transmission of the first communication have all ended. Clear condition: software writes 0, or software writes SPI_START to 1 (that is, it is automatically cleared when the next communication starts).
0	SPI_START	1'b0	W/R	Communication begins. Please set the ratio close to 1 after all the phase configurations of one communication. Clear condition: automatically clear after SPI communication ends. Please do not write 0 in the software.

phase _ ctrl0 ~ phase _ ctrl7 registers (address: base _ addr+8' h10 ~ 8' h2c)

The PHASE_CTRL register is used to individually configure each phase in an SPI communication. There are eight PHASE_CTRL registers, phase _ ctrl0 ~ phase _ ctrl7, which control each phase in turn. The bit configuration of the PHASE_CTRL register is as follows:



The function of each bit is defined in the following table:

bit	Bitname	initial value	read and write	describe
31-22	Reserved	10'h000	R	Keep.
21-20	SPI_MODE	2'h0	W/R	Current SPI bus mode of PHASE: 2' H0: Single mode 2' H1: Dual mode 2' H2: quad mode 2'h3: setting is prohibited.
19-8	BYTE_CNT	12'h000	W/R	Number of data byte in current PHASE communication. The value is invalid when PHASE_ACTION is set to POLL.
7-6	Reserved	1'b0	R	Keep.
5-4	PHASE_ACTION	2'h0	W/R	Action of current PHASE SPI: 2'h0：TX 2'h1：DUMMY TX 2'h2：RX 2'h3：POLL Please refer to section 2.3 for details.
3	Reserved	1'b0	R	Keep.
2	PHASE_ERROR	1'b0	R	PHASE_ACTION is POLL, and it exceeds the number of attempts that need to be unread. Clear condition: SPCR writes SPI_START to 1 (that is, it is automatically cleared when the next communication starts).
1	PHASE_DONE	1'b0	R	The current PHASE has been completed.

				Clear condition: SPCR writes SPI_START to 1 (that is, it is automatically cleared when the next communication starts).
0	PHASE_START	1'b0	R	PHASE is currently in progress. The hardware is automatically set and cleared according to the running situation.

phase _ data0 ~ phase _ data7 registers (address: base _ addr+8' h30 ~ 8' h4c)

Data register for each PHASE. The bit definition of the PHASE_DATA register is related to the PHASE _ action set by the current phase.

The details are as follows:

When PHASE_ACTION is POLL, it is used to save the configuration related to POLL:

bit31	bit23	bit15	bit7	bit0
POLL_LIMIT	POLL_MASK	POLL_EXPECT	POLL_READ	

When (POLL _ read & poll _ mask) = = poll _ expect, and the number of attempts is less than POLL_LIMIT, poll succeeds; Otherwise, the POLL fails and the PHASE_ERROR is set to one.

When the PHASE_ACTION is not POLL, the PHASE_DATA register is used to store the sent/received data, with a maximum of 4 byte.

If SPCR.LE is 0, the functions are as follows:

bit31	bit23	bit15	bit7	bit0
data byte 0	data byte 1	data byte 2	data byte 3	

If SPCR.LE is 1, the functions are as follows:

bit31	bit23	bit15	bit7	bit0
data byte 3	data byte 2	data byte 1	data byte 0	

That is, if sending data, SPI will first send data byte0, then data byte 1, until data byte 3. On the contrary, when receiving data, the position of data byte 0 will be written first until data byte 3.

21.2.3 Description of PHASE_ACTION

When SPCR.SPI_START=1, the hardware will automatically install PHASE0 → phase1 ... phase7 to execute the transactions in each phase until the number specified in SPCR.PHASE_CNT is completed. According to the functions of common flash chips, there are four kinds of ACTION that SPI needs to perform in this module: TX, DUMMY TX, RX and POLL. Here's a detailed description of the definition of each operation.

TX

When PHASE_ACTION is TX, SPI will circularly send the data in PHASE_DATA in the order of data byte0 ~ data byte3 according to the bus mode configured by SPI_MODE until all the specified phase _ byte _ CNTs are sent. Therefore, when the PHASE_BYTE_CNT is less than 5, the software can directly match the data into the PHASE_DATA；. Otherwise, it is recommended to configure DMA, so that SPI will automatically call DMA to write new data into PHASE_DATA at the end of a cycle.

DUMMY TX

DUMMY TX is similar to TX except that SPI will no longer send data in PHASE_DATA, but will send 8'hFF. Therefore, even if the PHASE_BYTE_CNT is greater than 4, there is no need to configure DMA.

RX

When PHASE_ACTION is RX, SPI will receive data according to the bus mode configured by SPI_MODE, and write it into PHASE_DATA circularly in the order of data byte0 ~ data byte3. Therefore, when the PHASE_BYTE_CNT is less than 5, the software can read the data in the PHASE_DATA after the SPI communication. Otherwise, it is recommended to configure DMA, so that SPI will automatically call DMA to send away the data in PHASE_DATA at the end of a cycle to avoid being overwritten by new data.

POLL

When PHASE_ACTION is POLL, SPI will continue to receive data according to the bus mode configured by SPI_MODE, and do the comparison operation of (poll _ read & poll _ mask) = = poll _ expect. If it is successful within the number of times specified in POLL_LIMIT, the POLL operation is completed; Otherwise, after the specified number of times of POLL_LIMIT is reached, the POLL operation is forcibly completed, and the PHASE_ERROR is set to one.

Note: If the POLL_LIMIT is set to 8'h00, the infinite POLL mode will be entered. SPI will perform the POLL operation indefinitely until the comparison is successful, otherwise it cannot be stopped. The only way to stop is SOFT_RESET.

Pay attention.

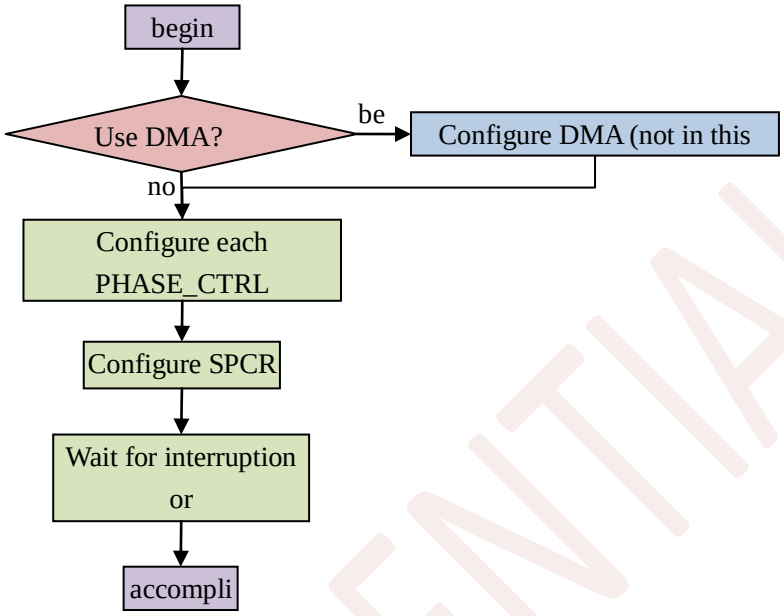
1)RX and POLL must be the last PHASE of an SPI communication and not the first PHASE.

2) If the PHASE is set to use DMA, its PHASE_ACTION cannot be POLL.

3) If it is set to the PHASE using DMA, and its PHASE_ACTION is RX, it may happen that SPI communication has been completed, but DMA has not yet been completed. At this time, SPCR.SPI_START will be cleared, but SPCR.SPI_DONE will not be set. SPCR.SPI_DONE will not be set until the DMA transfer is complete.

21.2.1 Software configuration sequence

The software can configure the registers in the following order during each SPI communication:

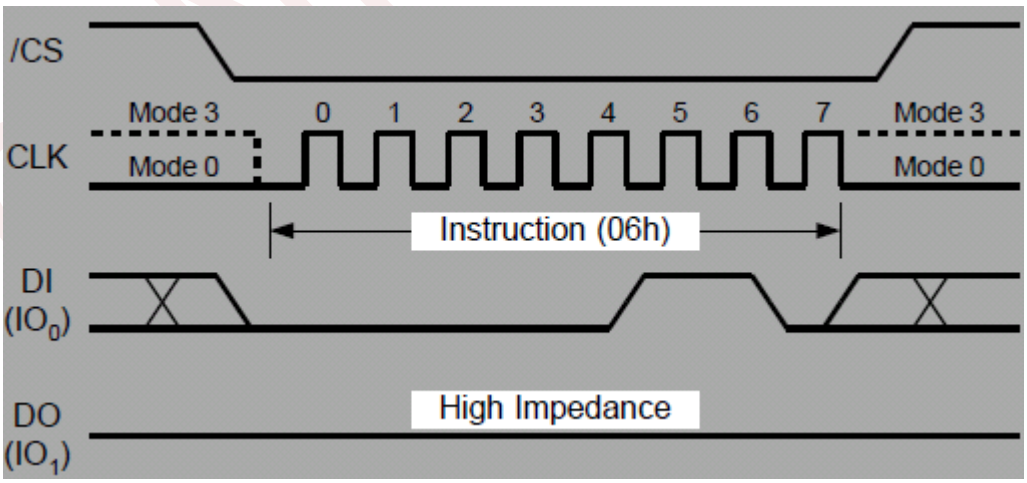


Example of software configuration

This section will introduce how to configure this module in combination with the common operations of common FLASH chips.

Send the Write Enable(06h) command.

The format of this command is as follows:



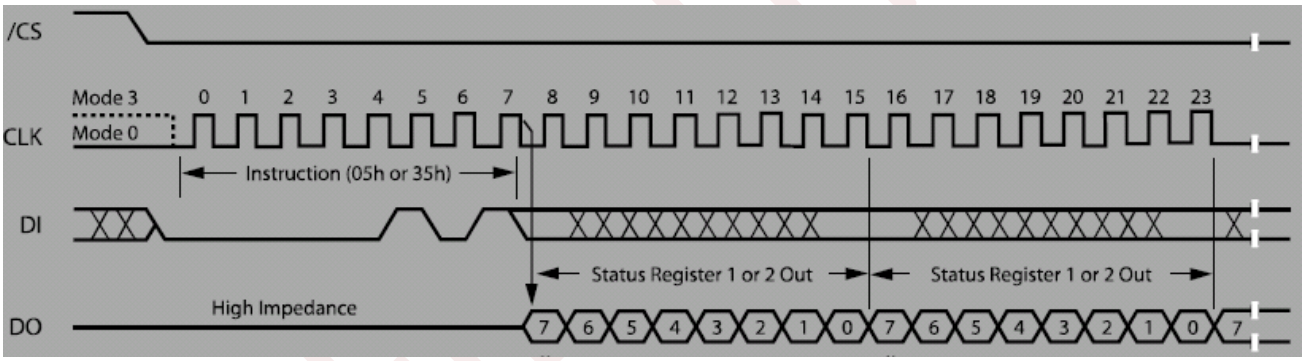
The following configuration is recommended:

sequence	Register name	Configuration value
one	PHASE_CTRL0	SPI_MODE=2'h0 BYTE_CNT=12'h001 PHASE_ACTION=2'h0
2	PHASE_DATA0	32'h0600_0000(SPCR.LE=0)/32'h0000_0006(SPCR.LE=1)
3	SPCR	USE_DMA=1'b0 PHASE_CNT=3'h0 WP=1'b0 SPI_START=1'b1

After configuration, SPI will start communication, and software can wait for interrupt or SPCR.SPI_START=0.

Send the Read Status Register-1(05h) command.

The sequence of this command is as follows:



Until CSn becomes 1, the value of Read Status Register-1 will be read repeatedly.

Example: read the value of Read Status Register-1 four times.

The following configuration is recommended:

sequence	Register name	Configuration value
one	PHASE_CTRL0	SPI_MODE=2'h0 BYTE_CNT=12'h001 PHASE_ACTION=2'h0
2	PHASE_DATA0	32'h0500_0000(SPCR.LE=0)/32'h0000_0005(SPCR.LE=1)
3	PHASE_CTRL1	SPI_MODE=2'h0 BYTE_CNT=12'h004 PHASE_ACTION=2'h2
4	SPCR	USE_DMA=1'b0 PHASE_CNT=3'h1 WP=1'b1 SPI_START=1'b1

After configuration, SPI will first run PHASE0, send 05h, then run PHASE1, read 4 byte, and

store the read value in the PHASE_DATA1 register. The software can read the value in PHASE_DATA1 after waiting for SPI_START=0.

Example: wait for the 0th bit of Read Status Register-1 to be 0, but try to read it 100 times at most.

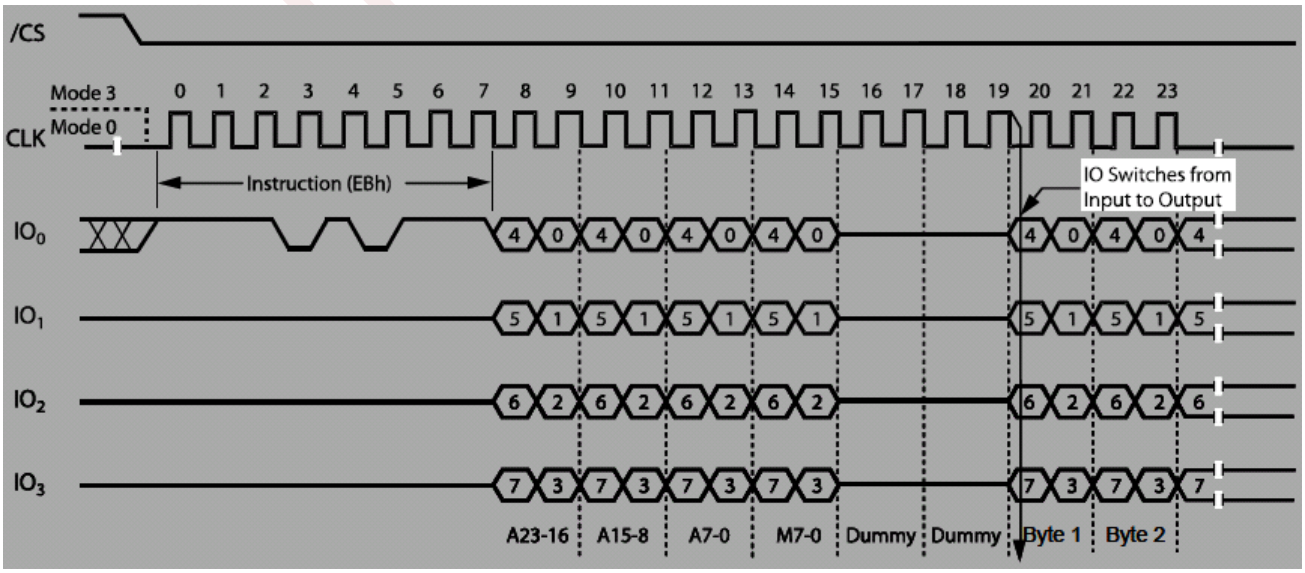
The following configuration is recommended:

sequence	Register name	Configuration value
1	PHASE_CTRL0	SPI_MODE=2'h0 BYTE_CNT=12'h001 PHASE_ACTION=2'h0
2	PHASE_DATA0	32'h0500_0000(SPCR.LE=0)/32'h0000_0005(SPCR.LE=1)
3	PHASE_CTRL1	SPI_MODE=2'h0 PHASE_ACTION=2'h3
4	PHASE_DATA1	32'h64010000
5	SPCR	USE_DMA=1'b0 PHASE_CNT=3'h1 WP=1'b1 SPI_START=1'b1

After the configuration is completed, spi will first run PHASE0, send 05h, then run phase1, constantly read data through SPI, compare the result with 0x00 and then compare it with 0x00 until the comparison is successful or exceeds 0x64 times. After waiting for SPI_START=0, the software can judge whether SPI_ERROR is 1 or not until the waiting is successful.

Read data with Fast Read Quad IO(EBh) command.

The timing diagram of this command is as follows:



Suppose the address is A=24'h123456 and M=78, and 100 data are read.

The recommended configuration is as follows:

sequence	Register name	Configuration value
1	DMA source address	PHASE_DATA3
2	PHASE_CTRL0	SPI_MODE=2'h0 BYTE_CNT=12'h001 PHASE_ACTION=2'h0
3	PHASE_DATA0	32'hEB00_0000(SPCR.LE=0)/32'h0000_00EB(SPCR.LE=1)
4	PHASE_CTRL1	SPI_MODE=2'h2 BYTE_CNT=12'h004 PHASE_ACTION=2'h0
5	PHASE_DATA1	32'h1234_5678(SPCR.LE=0)/32'h7856_3412(SPCR.LE=1)
6	PHASE_CTRL2	SPI_MODE=2'h2 BYTE_CNT=12'h002 PHASE_ACTION=2'h1
7	PHASE_CTRL3	SPI_MODE=2'h2 BYTE_CNT=12'h064 PHASE_ACTION=2'h2
8	SPCR	USE_DMA=1'b1 PHASE_CNT=3'h3 WP=1'b1 SPI_START=1'b1

After the communication starts, SPI first communicates with PHASE0 and sends 0xEB. Then send 0x12345678 of PHASE1. When configuring here, send A and M in one PHASE, because for SPI, A and M are the same sending data. Then send two dummy byte of PHASE2. At last, the one running PHASE3 charges 100 byte, and since SPCR.USE_DMA is set to 1, when PHASE_DATA3 is written, a DMA request will be sent to read the data in PHASE_DATA3.

22 Other Interfaces

22.1 Universal serial bus full-speed device interface (USBD)

- One full-speed USB Interface with frequency up to 12 Mbit/s
- Internal 60 MHz oscillator support crystal-less operation
- Internal main PLL for USB CLK compliantly

The Universal Serial Bus (USB) is a 4-wire bus with 4 bidirectional endpoints. The device controller enables 12 Mbit/s data exchange with integrated transceivers. Transaction formatting is performed by the hardware, including CRC generation and checking. It supports device modes. Transaction formatting is performed by the hardware, including CRC generation and checking.

The status of a completed USB transfer or error condition is indicated by status registers. An interrupt is also generated if enabled. The required precise 48 MHz clock which can be generated from the internal main PLL (the clock source must use an HXTAL crystal oscillator) or by the internal 48 MHz oscillator in automatic trimming mode that allows crystal-less operation.

AG32 has been integrated with tinyUSB in the project and can be used independently. The pins used by USB are fixed pins and cannot be changed in VE. In the routine, USB is enumerated as both cdc and msc (also supports HID and MIDI).

In the routine, the USB descriptor, callback, and configuration (CDC, HID, MSC, MIDI) have all been opened through the interface in. c.h under the src path. Users can customize or modify according to their own needs. For a detailed explanation of the configuration section and the use of the USB interface, please refer to the file description under the tinyUSB path under sdk, or refer to tinyUSB

22.2 Ethernet MAC interface with dedicated DMA and IEEE 1588 support

Peripheral available only on the AG32 devices.

The AG32 devices provide an IEEE-802.3-2002-compliant media access controller (MAC) for ethernet LAN communications through an industry-standard medium-independent interface (MII) or a reduced medium-independent interface (RMII). The AG32 requires an external physical interface device (PHY) to connect to the physical LAN bus (twisted-pair, fiber, etc.). the PHY is connected to the AG32 MII port using 17 signals for MII or 9 signals for RMII, and can be clocked using the 25 MHz (MII) from the AG32.

AG32 supports MAC modules. Supports RMII/MII interfaces. Currently, Lwip2.1.0 version is integrated into the SDK. In the example, the server-side functionality was used.

22.3 Debug mode

- Serial wire JTAG debug port (SWJ-DP)

The SWJ-DP Interface is embedded and is a combined JTAG and serial wire debug port that enables either a serial wire debug or a JTAG probe to be connected to the target.

23 Electrical characteristics

● Absolute maximum ratings

The maximum ratings are the limits to which the device can be subjected without permanently damaging the device. Note that the device is not guaranteed to operate properly at the maximum ratings. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

Table 1. Absolute maximum rating

Symbol	Parameter	Min	Max	Unit
VDD	External voltage range	VSS - 0.3	VSS + 3.6	V
VDDA	External analog supply voltage	VSSA - 0.3	VSSA + 3.6	V
VBAT	External battery supply voltage	VSS - 0.3	VSS + 3.6	V
VIN	Input voltage on I/O	VSS - 0.3	VSS + 3.6	V
Iio	Maximum current for GPIO pins	—	25	mA
Iinj	Injected current on I/O	—	±5	mA
TA	Operating temperature range	-40	+85	℃
TSTG	Storage temperature range	-55	+150	℃
TJ	Maximum junction temperature	—	125	℃

● Recommended DC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDD	Supply voltage	—	3.0	3.3	3.6	V
VDDA	Analog Supply voltage	—	3.0	3.3	3.6	V
VBAT	Battery supply voltage	—	2.2	—	3.6	V

Electrostatic discharge(ESD) & Latch-up

Parameter	Description	Conditions	Maximum value	Unit
VESD(HBM)	Maximum ESD	Electrostatic discharge voltage (human body model)	2000	V
VESD(CDM)	Maximum ESD	Electrostatic discharge voltage (charge device model)	500	V
LU	Latch-up		≥100	mA

Power consumption

The power measurements specified in the tables represent that code with data executing from on- chip Flash with the following specifications.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Idd	Supply current (Run mode)	VDD=VBAT=3.3V, HSE=8MHz, System clock=108 MHz, All peripherals enabled	—		—	mA
		VDD=VBAT=3.3V, HSE=8MHz, System clock =108 MHz, All peripherals disabled	—		—	mA
		VDD=VBAT=3.3V, HSE=8MHz, System clock =72MHz, All peripherals enabled	—		—	mA
		VDD=VBAT=3.3V, HSE=8MHz, System Clock =72 MHz, All peripherals disabled	—		—	mA
	Supply current (Sleep mode)	VDD=VBAT=3.3V, HSE=8MHz, CPU clock off, All peripherals enabled	—		—	mA
		VDD=VBAT=3.3V, HSE=8MHz, CPU clock off, All peripherals disabled	—		—	mA
	Supply current (Deep-Sleep mode)	VDD=VBAT=3.3V, All clock off, LSI on, RTC on, All IOs analog mode	—		——	mA
	Supply current (Standby mode)	VDD=VBAT=3.3V, LDO off, LSE off, LSI on, RTC on	—		—	μA
Ibat	Battery supply current (Standby mode)	VDD not available, VBAT=3.3V, LDO off, LSE on, LSI off, RTC on	—		—	μA
		VDD not available, VBAT=3.3 V, LDO off, LSE off, LSI on, RTC on	—		—	μA

Power up/down

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vpor	Power on reset threshold		2.0	2.2	2.4	V
Vpdr	power down reset threshold		1.8	2.0	2.2	V
Vhyst	PDR hysteresis		—	0.2	—	V
Trsttemp	Reset temporization		—	4	—	ms

External clock characteristics

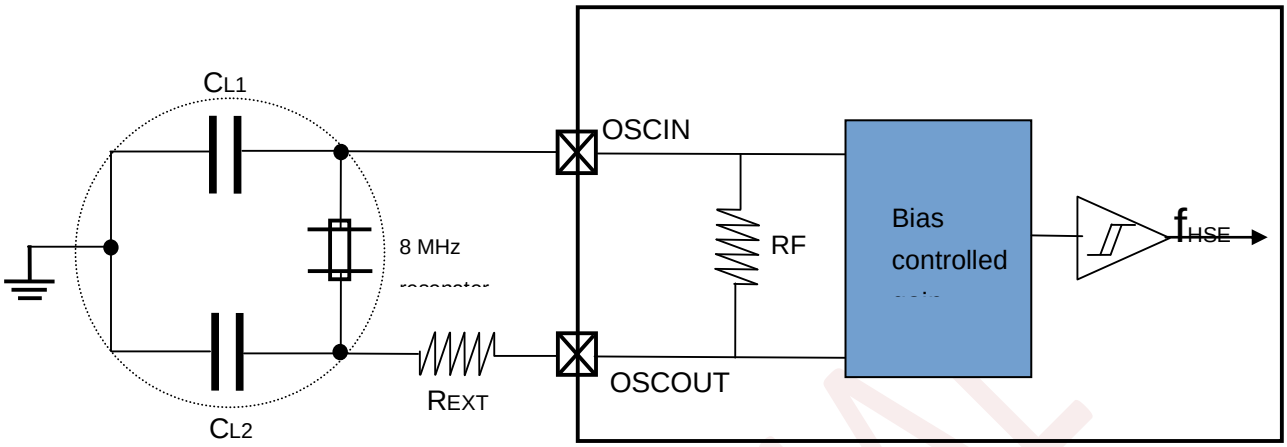
High-speed external clock generated from a crystal/ceramic resonator. The high-speed external (HSE) clock can be supplied with a 4 to 16 MHz crystal/ceramic resonator oscillator. All the information given in this paragraph are based on characterization results obtained with typical external components specified in Table. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. Refer to the crystal resonator manufacturer for more details on the resonator characteristics (frequency, package, accuracy).

HSE 4-26 MHz oscillator characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_OSC_IN	Oscillator frequency	VDD=3.3V	4	8	26	MHz
RF	Feedback resistor	—	—	1	—	MΩ
C	Recommended load capacitance on OSC_IN and OSC_OUT	—	—	20	30	pF
gm	Oscillator transconductance	—	25	—	—	mA/V
Dosc_out	Oscillator oscillator duty cycle	—	45	50	55	%
T_su_hse	startup time	VDD is stabilized	—	2	—	mS

For CL1 and CL2, it is recommended to use high-quality external ceramic capacitors in the 5pF to 25pF range (typ.), designed for high-frequency applications, and selected to match the requirements of the crystal or resonator (see Figure 24). CL1 and CL2 are usually the same size. The crystal manufacturer typically specifies a load capacitance which is the series combination of CL1 and CL2. PCB and MCU pin capacitance must be included (10pF can be used as a rough estimate of the combined pin and board capacitance) when sizing CL1 and CL2.

Typical application with an 8 MHz crystal

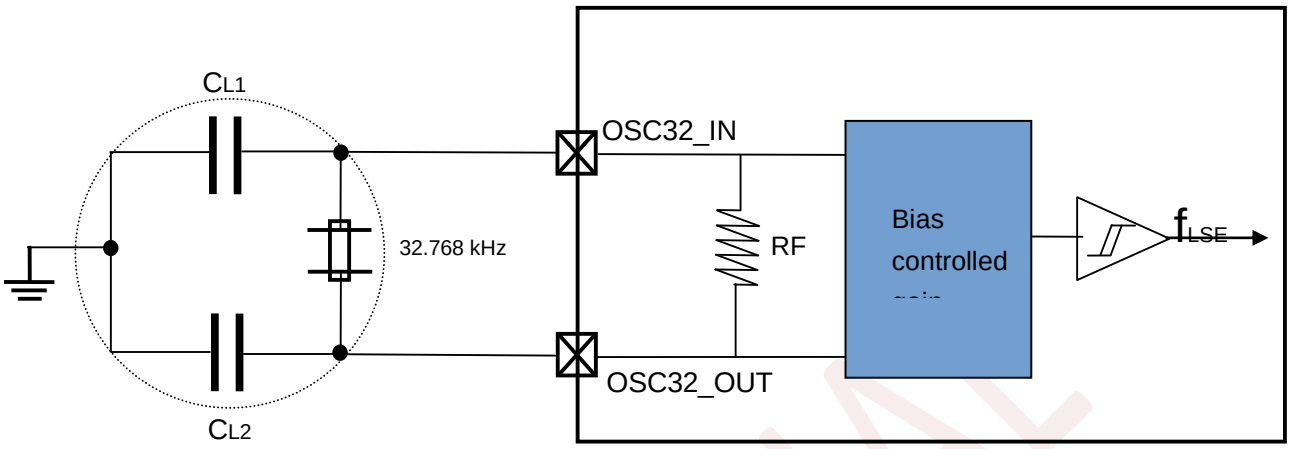


REXT value depends on the crystal characteristics.

Low-speed external clock generated from a crystal/ceramic resonator The low-speed external (LSE) clock can be supplied with a 32.768 kHz crystal/ceramic resonator oscillator. All the information given in this paragraph are based on characterization results obtained with typical external components specified in Table. In the application, the resonator and the load capacitors have to be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. Refer to the crystal resonator manufacturer for more details on the resonator characteristics (frequency, package, accuracy).

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_LSE	Oscillator frequency	VDD=VBAT=3.3V		32.768	1000	KHz
RF	Feedback resistor	—	—	10	—	MΩ
C	Recommended load capacitance on OSC32_IN and OSC32_OUT	—	—	—	15	pF
gm	Oscillator transconductance	—	10	—	—	uA/V
Dosc_out	Oscillator oscillator duty cycle	—	45	50	55	%
T_su_lse	startup time	VDD is stabilized	—	3	—	S

Typical application with a 32.768 kHz crystal



Internal clock source characteristics

High-speed internal (HSI) RC oscillator

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{HSI}	Oscillator frequency	VDD=3.3V	10	20	40	MHz
D_{uty_HSI}	Duty cycle		45	50	55	%
T_{su_HSI}	HSI oscillator startup time		1	—	2	μs

PLL characteristics

Symbol	Parameter	Min	Typ	Max	Unit
f_{PLL_IN}	PLL input clock	4	20	50	MHz
	PLL input clock duty cycle	40	50	60	%
f_{PLL_OUT}	PLL multiplier output clock	2	200	300	MHz
t_{LOCK}	PLL lock time	—	—	400	μs
Jitter	Cycle-to-cycle jitter	—	—	400	ps

Memory characteristics

Flash memory characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
PE_{CYC}	Number of guaranteed program /erase cycles before failure (Endurance)	TA=-40 ℃ ~ +85 ℃	100	—	—	kcycles
t_{RET}	Data retention time	TA=125 ℃	20	—	—	years
t_{PROG}	Word programming time	TA=-40 ℃ ~ +85 ℃	—	2	3	ms
t_{ERASE}	Page erase time	TA=-40 ℃ ~ +85 ℃	—	8	20	ms
t_{MERASE}	Mass erase time	TA=-40 ℃ ~ +85 ℃	—	8	20	ms

IO characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IL}	Standard IO Low level input voltage	VDD ≥ 3.0V	-0.3	—	0.8	V
V_{IH}	Standard IO High level input voltage	VDD ≥ 3.0V	1.5	—	3.6	V
V_{OL}	Low level output voltage	VDD ≥ 3.0V	—	—	0.2	V
V_{OH}	High level output voltage	VDD ≥ 3.0V	2.8	—	—	V
R_{PU}	Internal pull-up resistor	VIN=VSS	30	40	50	kΩ
R_{PD}	Internal pull-down resistor	VIN=VDD	30	40	50	kΩ

ADC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDDA	Operating voltage		3.0	3.3	3.6	V
VIN	ADC input voltage range		0	—	VREFP	V
f_{ADC}	ADC clock		0.5	—	13	MHz
f_S	Sampling rate		—	—	1	MHz
t_{conv}	ADC conversion time		1	—	20	μs
R_{ADC}	Input sampling switch resistance		—	—	0.5	kΩ
C_{ADC}	Input sampling capacitance		—	8	—	pF
tsu	Startup time		—	—	2	μs

DAC characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDDA	Operating voltage		3.0	3.3	3.6	V
VREFP	Reference supply voltage	VREFP should always below VDDA	3.0	3.3	3.6	V
R_{LOAD}	Load resistance	Resistive load vs. VSSA with buffer ON	5	—	—	kΩ
C_{LOAD}	Load capacitance	No pin/pad capacitance included	—	—	50	pF

Comparator characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VDDA	Analog supply voltage	—	3.0	3.3	3.6	V
VIN	Comparator input voltage range	—	0	—	VDDA	V
t_{start}	Comparator startup time	VDDA ≥ 3.0 V	—	—	10	μs
t_D	Propagation delay for full range step with 100 mV overdrive	VDDA ≥ 3.0 V	—	—	40	ns
V_{OFFSET}	Comparator offset error	VDDA ≥ 3.0 V	—	—	±25	mV

I^2C characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
f_{SCL}	SCL clock frequency	—	0	—	100	KHz
$t_{SCL(H)}$	SCL clock high time	—	4.0	—	0.6	ns
$t_{SCL(L)}$	SCL clock low time	—	4.7	—	1.3	ns

SPI characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
fSCK	SCK clock frequency	—	—	—	18	MHz
tSCK(H)	SCK clock high time	—	19	—	—	ns
tSCK(L)	SCK clock low time	—	19	—	—	ns
SPI master mode						
tV(MO)	Data output valid time	—	—	—	25	ns
tH(MO)	Data output hold time	—	2	—	—	ns
tSU(MI)	Data input setup time	—	5	—	—	ns
tH(MI)	Data input hold time	—	5	—	—	ns
SPI slave mode						
tSU(NSS)	NSS enable setup time	fPCLK=54MHz	74	—	—	ns
tH(NSS)	NSS enable hold time	fPCLK=54MHz	37	—	—	ns
tA(SO)	Data output access time	fPCLK=54MHz	0	—	55	ns
tDIS(SO)	Data output disable time	—	3	—	10	ns
tV(SO)	Data output valid time	—	—	—	25	ns
tH(SO)	Data output hold time	—	15	—	—	ns
tSU(SI)	Data input setup time	—	5	—	—	ns
tH(SI)	Data input hold time	—	4	—	—	ns

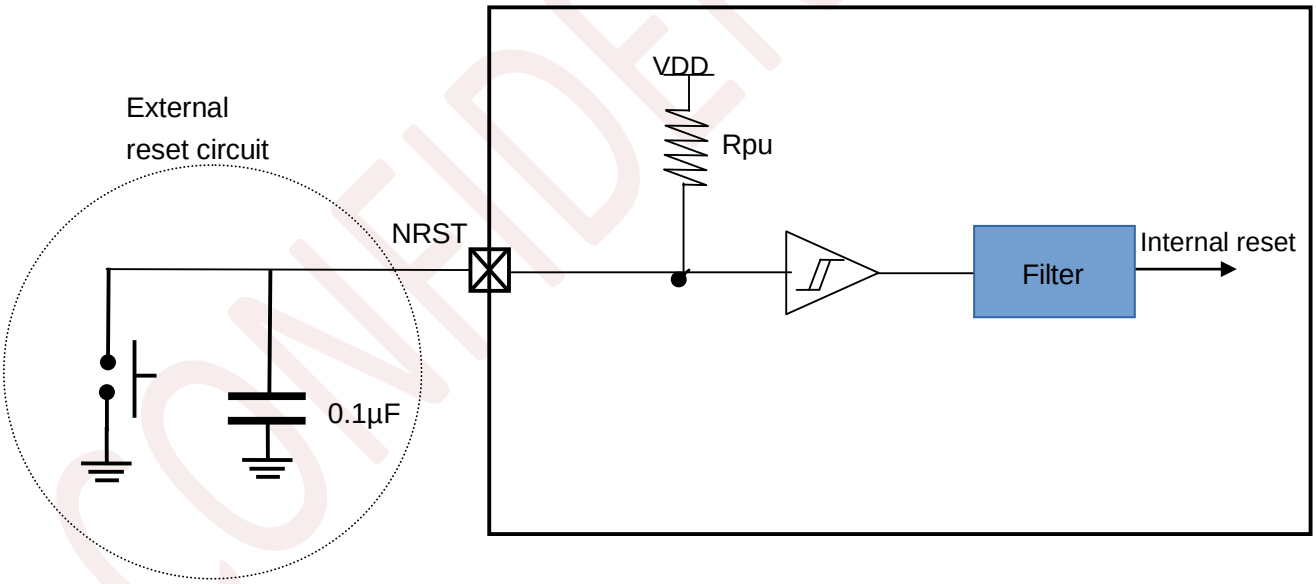
NRST pin characteristics

The NRST pin input driver uses CMOS technology. It is connected to a permanent pull-up resistor, RPU.

NRST pin characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
VIL(NRST)	NRST Input low level voltage	—	—	—	0.2VDD	V
VIH(NRST)	NRST Input high level voltage	—	0.5VDD	—	—	V
Vhys(NRST)	NRST Schmitt trigger voltage hysteresis	—	—	200	—	mV
RPU	Weak pull-up equivalent resistor	VIN = VSS	30	40	50	kΩ
VF(NRST)	NRST Input filtered pulse	—	—	—	100	ns
VNF(NRST)	NRST Input not filtered pulse	—	500	—	—	ns

Recommended NRST pin protection



Symbol	Parameter	Conditions	Min	Typ	Max	Unit
fPP	Clock frequency in data transfer mode	—	0	—	48	MHz
tW(CKL)	Clock low time	fpp = 48 MHz	10.5	11	—	ns
tW(CKH)	Clock high time	fpp = 48 MHz	9.5	10	—	ns
CMD, D inputs (referenced to CK) in MMC and SD HS mode						
tISU	Input setup time HS	fpp = 48 MHz	4	—	—	ns
tIH	Input hold time HS	fpp = 48 MHz	3	—	—	ns
CMD, D outputs (referenced to CK) in MMC and SD HS mode						
tOV	Output valid time HS	fpp = 48 MHz	—	—	13.8	ns
tOH	Output hold time HS	fpp = 48 MHz	12	—	—	ns
CMD, D inputs (referenced to CK) in SD default mode						
tISUD	Input setup time SD	fpp = 24 MHz	3	—	—	ns
tIHD	Input hold time SD	fpp = 24 MHz	3	—	—	ns
CMD, D outputs (referenced to CK) in SD default mode						
tOVD	Output valid default time SD	fpp = 24 MHz	—	2.4	2.8	ns
tOHD	Output hold default time SD	fpp = 24 MHz	0.8	—	—	ns

- **UART characteristics**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
fSCK	SCK clock frequency	fPCLK = 120 MHz	—	—	60	MHz
tSCK(H)	SCK clock high time	fPCLK = 120 MHz	7.5	—	—	ns
tSCK(L)	SCK clock low time	fPCLK = 120 MHz	7.5	—	—	ns

● USB characteristics

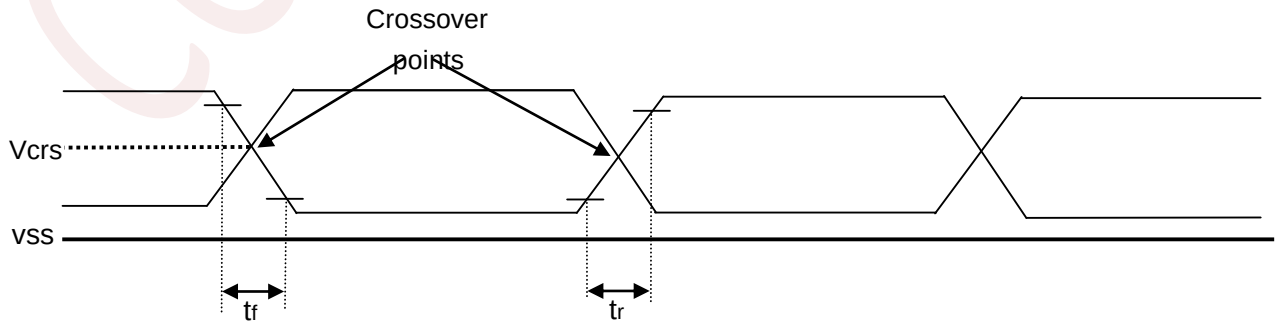
USB DC electrical characteristics

Symbol		Parameter	Conditions	Min	Typ	Max	Unit
Input levels	VDD	USB operating voltage	—	3	—	3.3	V
	VDI	Differential input sensitivity	I(USBDP, USBDM)	0.2	—	—	V
	VCM	Differential common mode range	Includes VDI range	0.8	—	2.5	V
	VSE	Single ended receiver threshold	—	1.3	—	2.0	V
Output Levels	VOL	Static output level low	RL of 1.5 kΩ to 3.6 V	—	—	0.3	V
	VOH	Static output level high	RL of 15 kΩ to VSS	2.8	3.3	3.6	V
tSTARTUP		USBFS startup time	—	—	—	1	μs

USB full speed-electrical characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
tR	Rise time	CL = 50 pF	4	—	20	ns
tF	Fall time	CL = 50 pF	4	—	20	ns
tRFM	Rise/ fall time matching	tR/tF	90	—	110	%
vCRS	Output signal crossover voltage	—	1.3	—	2.0	V

USB timings: definition of data signal rise and fall time



TIMER characteristics

TIMER characteristics

Symbol	Parameter	Conditions	Min	Max	Unit
tres	Timer resolution time	—	1	—	$t_{TIMERxCLK}$
		$f_{TIMERxCLK}$ = 120MHz	8.4	—	ns
fEXT	Timer external clock frequency	—	0	$f_{TIMERxCLK}/2$	MHz
		$f_{TIMERxCLK}$ = 120MHz	0	60	MHz
RES	Timer resolution	—	—	16	bit
$t_{COUNTER}$	16-bit counter clock period when internal clock is selected	—	1	65536	$t_{TIMERxCLK}$
		$f_{TIMERxCLK}$ = 120MHz	0.0084	546	μs
t_{MAX_COUNT}	Maximum possible count	—	—	65536x65536	$t_{TIMERxCLK}$
		$f_{TIMERxCLK}$ = 120MHz	—	35.7	s

24 Package and operation temperature

LQFP100 (AG32VF303Vx,AG32VF407Vx),LQFP64 (AG32VF407Rx) and LQFP48 (AG32VF303Cx)

Operation temperature range: -40 °C to +85 °C ν

25 Order Information

Ordering code	Flash (KB)	Package	Package type	Temperature operating range
AG32VF303CCT6	256	LQFP48	Green	Industrial -40°C to +85°C
AG32VF303VCT6	256	LQFP100	Green	Industrial -40°C to +85°C
AG32VF407RGT6	1024	LQFP64	Green	Industrial -40°C to +85°C
AG32VF407VGT6	1024	LQFP100	Green	Industrial -40°C to +85°C

26 Revision history

Revision No.	Description	Date
1.0	Initial Release	Apr.20,2022
1.1	Second Edition	May.10,2023