ARM®-based 32-bit Cortex®-M4F MCU, 256 to 512 KB Flash, sLib, QSPI, SDRAM, 17 timers, 2 ADCs, 23 communication interfaces (3x CAN, OTGFS, EMAC)

- **Core: ARM® 32-bit Cortex®-M4F CPU with FPU**
  - 192 MHz maximum frequency, with a memory protection unit (MPU), single-cycle multiplication and hardware division
  - Floating point unit (FPU)
  - DSP instructions
- **Memories**
  - 256 to 512 Kbytes of Flash memory
  - 26 Kbytes of boot memory used as a Bootloader or as a general instruction/data memory (one-time-configurable)
  - 4 Kbytes of OTP memory
  - sLib: configurable part of main Flash as a library area with code executable but secured, non-readable
  - 108 to 144 Kbytes of SRAM (configurable as 96 to 128 KB SRAM with parity check)
  - External memory controller (XMC) with 16-bit data bus supporting NOR, PSRAM, SRAM and SDRAM memories
  - QSPI interfacing for external SPI Flash or SPI RAM extension, supporting address mapping
- **XMC as LCD parallel interface, 8080/6800 modes**
- **Power control (PWC)**
  - 2.4 to 3.6 V supply
  - Power-on reset (POR)/low voltage reset (LVR), and power voltage monitoring (PVM)
  - Low power modes: Sleep, Deepsleep and Standby modes (woke up via 6 WKUP pins)
  - V_{BAT} supply for LEXT, ERTC and 20x 32-bit battery powered registers (ERTC_BPR)
- **Clock and reset management (CRM)**
  - 4 to 25 MHz crystal oscillator (HEXT)
  - 48 MHz internal factory-trimmed high speed clock (HICK) with ±1% accuracy at T_A= 25 °C and ±2.5% at T_A= -40 °C to +105 °C, with automatic clock calibration (ACC)
  - PLL with configurable frequency multiplication and division factors
  - 32 kHz crystal oscillator (LEXT)
  - Low speed internal clock (LICK)
- **Analog**
  - 2x 12-bit 5.33MSPS A/D converters, up to 16 external input channels; 12/10/8/6-bit resolution, hardware oversampling up to equivalent 16-bit resolution
  - Temperature sensor (V_{TS}), internal reference voltage (V_{INTR}), V_{BAT} monitor (V_{BAT}/4)
  - 2x 12-bit D/A converters
- **DMA**
  - 2x 7-channel DMA controllers (14 channels in total)
- **Up to 117 fast GPIOs**
  - All mappable on 16 external interrupts
  - Almost all 5 V-tolerant
- **Up to 17 timers (TMR)**
  - 2x 16-bit 8-channel advanced timers, including PWM outputs with dead-time generator and emergency brake
  - Up to 8x 16-bit and 2x 32-bit general-purpose timers, each with up to 4 IC/OC/PWM or pulse counter and incremental encoder input, including 6 timers supporting complementary output with dead-time generator and emergency brake
  - 2x 16-bit basic timers
  - 2x watchdog timers (WDT and WWDT)
  - SysTick timer: 24-bit downcounter
- **ERTC: enhanced RTC, with auto wakeup, alarm, subsecond accuracy, hardware calendar, and calibration feature**
- **Up to 23 communication interfaces**
  - Up to 3x I^2C interfaces (SMBus/PMBus)
  - Up to 8x USART/UART interfaces support ISO7816, LIN, IrDA, modem control and RS485 driver enable, supporting TX/RX swap
  - Up to 4x SPI interfaces (40 Mbit/s), all with multiplexed I^2S, and I^2S2/I^2S3 full-duplex mode
  - 1x separated full-duplex I^2S interface (I^2SF)
  - Up to 3x CAN interfaces, each with dedicated 1408 bytes of buffer (AT32F455: 2.0B Active, AT32F456/AT32F457: FD Active)
  - SDIO interface
  - USB2.0 FS/host/OTG device interface, supporting crystal-less in device mode
  - 10/100M Ethernet MAC (EMAC) with dedicated DMA and 4 Kbytes of SRAM, IEEE 1588 support, MII/RMII available (for AT32F457 only)
  - Infrared transmitter (IRTMR)
- **CRC calculation unit**
- **96-bit unique ID (UID)**
- **AES hardware accelerator supporting 256/192/128-bit key**
- **True random number generator (TRNG)**
- **Debug mode**
  - Serial wire debug (SWD) and serial wire output (SWO)
- **Operating temperature: -40 to +105 °C**

- **Packages**
  - LQFP144 20 x 20 mm
  - LQFP100 14 x 14 mm
  - LQFP64 10 x 10 mm
  - LQFP48 7 x 7 mm
  - QFN48 6 x 6 mm
- **List of models**

Internal Flash	Model
256 Kbytes	AT32F455ZCT7 AT32F455VCT7 AT32F455RCT7 AT32F455CCT7 AT32F455CCU7 AT32F456ZCT7 AT32F456VCT7 AT32F456RCT7 AT32F456CCT7 AT32F456CCU7 AT32F457ZCT7 AT32F457VCT7 AT32F457RCT7
512 Kbytes	AT32F455ZET7 AT32F455VET7 AT32F455RET7 AT32F455CET7 AT32F455CEU7 AT32F456ZET7 AT32F456VET7 AT32F456RET7 AT32F456CET7 AT32F456CEU7 AT32F457ZET7 AT32F457VET7 AT32F457RET7

Contents

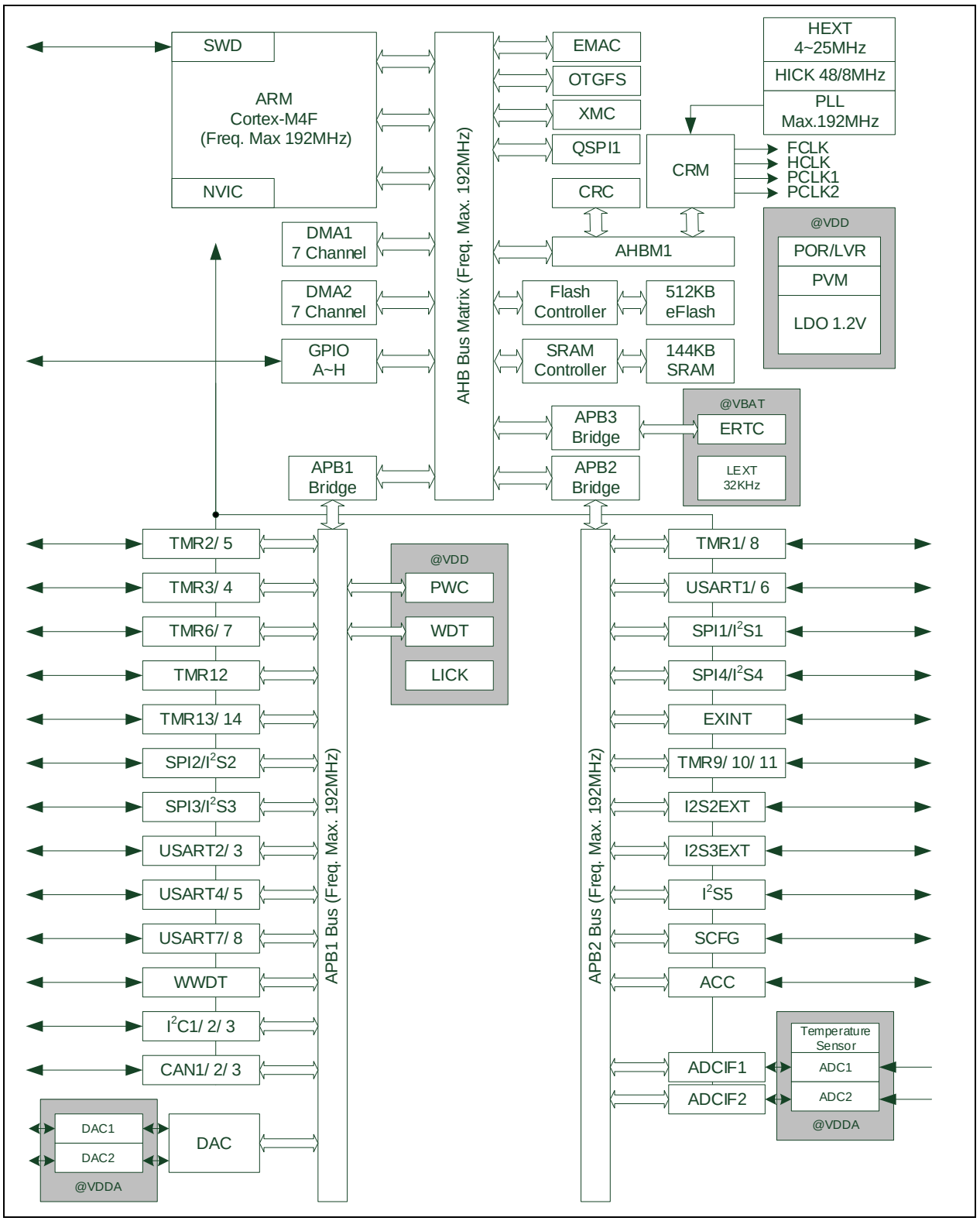
List of figures

List of tables

1 System architecture

AT32F455/456/457 series microcontroller consist of 32-bit ARM Cortex®-M4F processor, multiple 16-bit and 32-bit timers, infrared transmitter (IRTMR), DMA controller, ERTC, communication interfaces such as SPI, I^2C, USART/UART, CAN bus controller, external memory controller (XMC), USB2.0 OTG FS interfae, HICK with auto clock calibration (ACC), 12-bit ADC, programmable voltage monitor (PVM) and other peripherals. Cortex®-M4F proessor supports enhanced high-performance DSP instruction set, including extended single-cycle 16-bit/32-bit multiply accumulator (MAC), dual 16-bit MAC instructions, optimized 8-bit/16-bit SIMD operation and saturation operation instructions, and single-precision (IEEE-754) floating point unit (FPU), as shown in Figure 1-1.

Figure 1-1 AT32F455/456/457 series microcontroller system architecture


SWD
ARM
Cortex-M4F
(Freq. Max 192MHz)
NVIC
AHB Bus Matrix (Freq. Max. 192MHz)
EMAC
OTGFS
XMC
QSPI1
CRC
HEXT
4~25MHz
HICK 48/8MHz
PLL
Max.192MHz
CRM
FCLK
HCLK
PCLK1
PCLK2
@VDD
POR/LVR
PVM
LDO 1.2V
DMA1
7 Channel
DMA2
7 Channel
GPIO
A~H
AHBM1
Flash
Controller
512KB
eFlash
SRAM
Controller
144KB
SRAM
@VBAT
ERTC
LEXT
32KHz
APB3
Bridge
APB1
Bridge
APB2
Bridge
TMR2/ 5
TMR3/ 4
TMR6/ 7
TMR12
TMR13/ 14
SPI2/I2S2
SPI3/I2S3
USART2/ 3
USART4/ 5
USART7/ 8
WWDT
I2C1/ 2/ 3
CAN1/ 2/ 3
DAC1
DAC2
@VDDA
DAC
APB1 Bus (Freq. Max. 192MHz)
@VDD
PWC
WDT
LICK
TMR1/ 8
USART1/ 6
SPI1/I2S1
SPI4/I2S4
EXINT
TMR9/ 10/ 11
I2S2EXT
I2S3EXT
I2S5
SCFG
ACC
ADCIF1
ADCIF2
Temperature
Sensor
ADC1
ADC2
@VDDA
APB2 Bus (Freq. Max. 192MHz)

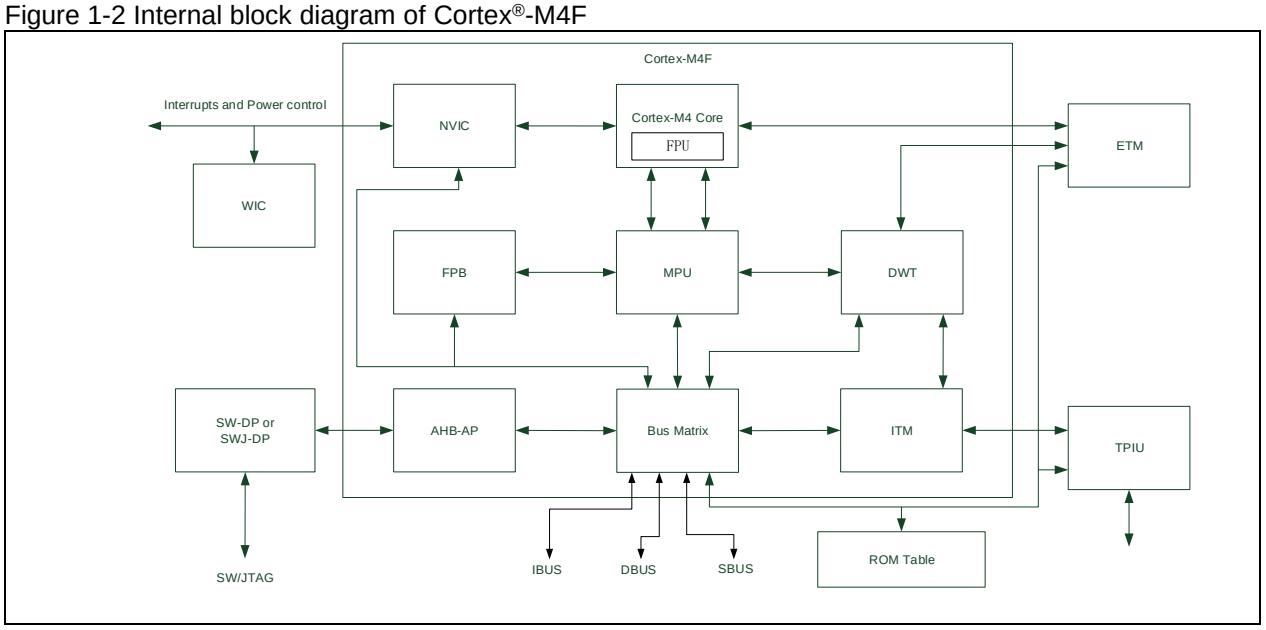
1.1 System overview

1.1.1 ARM Cortex®-M4F processor

Cortex®-M4F processor is a low-power consumption processor featuring with low gate count, low interrupt latency and low-cost debug. It supports DSP instruction set and FPU, and it is applicable to deeply embedded applications that require quicker response to interruption. Cortex®-M4F processor is based on ARMv7-M architecture, supporting both Thumb instruction set and DSP instruction set.

Figure 1-2 shows the internal block diagram of Cortex®-M4F processor. Please refer to *ARM® Cortex-M4 Technical Reference Manual* for more information.

Figure 1-2 Internal block diagram of Cortex®-M4F



1.1.2 Bit band

With the help of bit-band, read and write access to a single bit can be performed using common load/store operations. The Cortex®-M4F memory includes two bit-band regions: the least significant 1 Mbyte of SRAM and the least significant 1 Mbyte of peripherals. In addition to access to bit-band addresses, their respective bit-band alias region can be used to access to any bit in these two bit-band regions. The bit-band alias region transforms each bit into 32-bit word. Thus, accessing to a bit in an alias region has the same effect as read-modify-write operation on the corresponding bit in a bit-band region.

Figure 1-3 Comparison between bit-band region and its alias region: image A

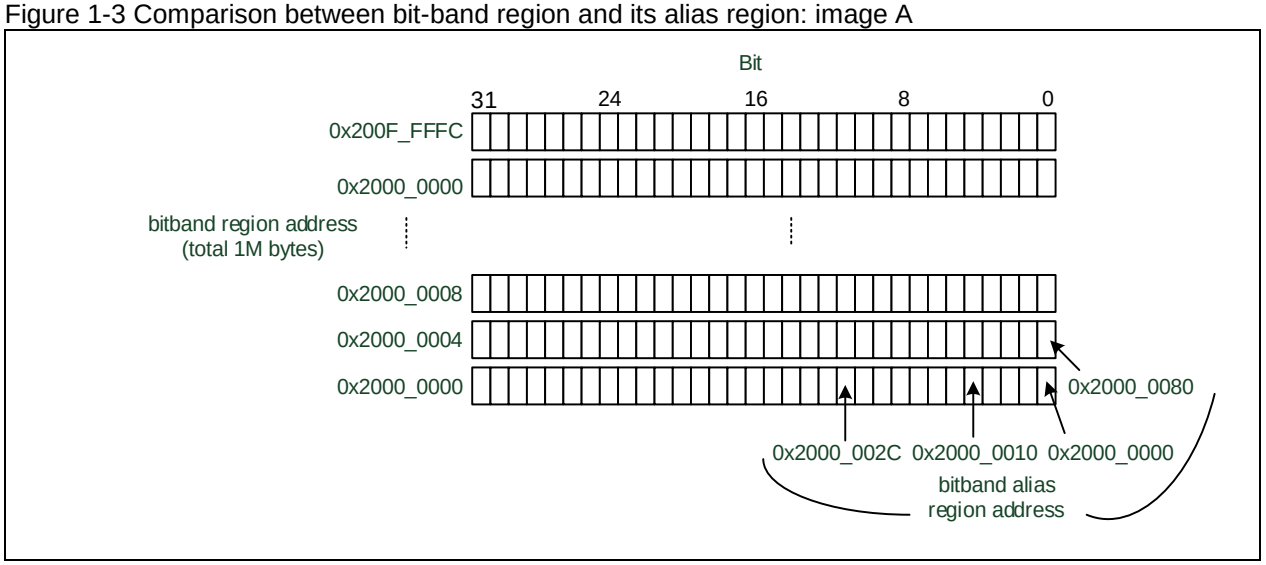
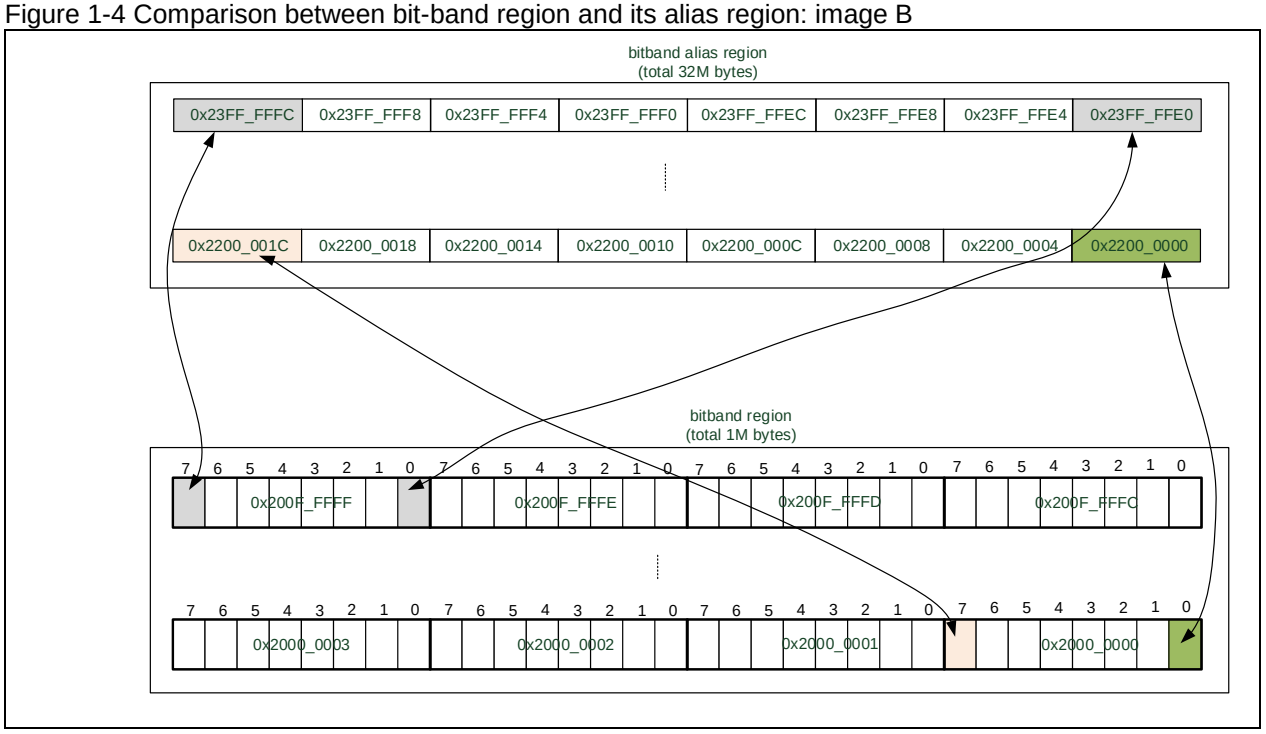


Figure 1-4 Comparison between bit-band region and its alias region: image B



Bit-band region: address region for bit-band operations

Bit-band alias region: access to the alias region has the same effect as read-modify-write operation on the bit-band region

Each bit in a bit-band region is mapped into a word (LSB) in an alias region. When accessing to the address in a bit-band alias region, such address is transformed into a bit-band address first. For a read operation, read one word in the bit-band region, and then move the targeted bit to the right to LSB before returning LSB. For a write operation, first move the targeted bit to the left to the corresponding bit number, then perform a read-modify-write operation on bit level.

The address ranges of two memories supporting bit-band operations:

Least significant 1 Mbyte in SRAM: 0x2000_0000~0x200F_FFFF

Least significant 1 Mbyte in peripherals: 0x4000_0000~0x400F_FFFF

For a bit in the SRAM bit-band region, if the byte address is A, the bit number is n (0<=n<=7), then the alias address where the bit is:

AliasAddr = 0x2200_0000+ (A-0x2000_0000)*32+n*4

For a bit in the peripheral bit-band region, if the byte address is A, the bit number is n (0<=n<=7), then the alias address where the bit is:

AliasAddr = 0x4200_0000+ (A-0x4000_0000)*32+n*4

Table 1-1 shows the mapping between bit-band region and alias region in SRAM.

Table 1-1 Bit-band address mapping in SRAM

Bit-band region	Equivalent alias address
0x2000_0000.0	0x2200_0000.0
0x2000_0000.1	0x2200_0004.0
0x2000_0000.2	0x2200_0008.0
…	…
0x2000_0000.31	0x2200_007C.0
0x2000_0004.0	0x2200_0080.0
0x2000_0004.1	0x2200_0084.0

0x2000_0004.2	0x2200_0088.0
…	…
0x200F_FFFC.31	0x23FF_FFFC.0

Table 1-2 shows the mapping between bit-band region and alias region in the peripheral area.

Table 1-2 Bit-band address mapping in the peripheral area

Bit-band region	Equivalent alias address
0x4000_0000.0	0x4200_0000.0
0x4000_0000.1	0x4200_0004.0
0x4000_0000.2	0x4200_0008.0
…	…
0x4000_0000.31	0x4200_007C.0
0x4000_0004.0	0x4200_0080.0
0x4000_0004.1	0x4200_0084.0
0x4000_0004.2	0x4200_0088.0
…	…
0x400F_FFFC.31	0x43FF_FFFC.0

In terms of bit-band operation, one of the advantages is to control LED ON/OFF independently via GPIO pins. On the other hand, it brings great convenience for serial interface operations. In short, it is best suited to hardware I/O-intensive low-level applications.

In addition, bit-band operations can also simplify jump process. When jump operation is based on a bit level, the previous steps are:

- Read the whole register
- Mask the undesired bits
- Compare and jump

For now, you just need to:

- Read the bit status from the bit-band alias region
- Compare and jump

Apart from making code more concise, its important function is also reflected in multi-task environment. When it comes to multiple tasks, it turns the read-modify-write operations into a hardware-supported atomic operation to avoid the scenario where the read-modify-write operation is disrupted, resulting in disorder.

1.1.3 Interrupt and exception vectors

Table 1-3 AT32F455/456/457 series vector table

Pos.	Priority	Prority type	Name	Description	Address
	-	-	-	Reserved	0x0000_0000
	-3	Fixed	Reset	Reset	0x0000_0004
	-2	Fixed	NMI	Non-maskable interrupt CRM clock fail detector (CFD) and SRAM verification are linked to NMI vector	0x0000_0008
	-1	Fixed	HardFault	All class of faults	0x0000_000C
	0	Configurable	MemoryManage	Memory management	0x0000_0010
	1	Configurable	BusFault	Pre-fetch default, memory access fault	0x0000_0014
	2	Configurable	UsageFault	Undefined instruction or illegal state	0x0000_0018
	-	-	-	Reserved	0x0000_001C ~0x0000_002B
	3	Configurable	SVCall	System service call via SWI instruction	0x0000_002C
	4	Configurable	Debug Monitor	Debug monitor	0x0000_0030
	-	-	-	Reserved	0x0000_0034
	5	Configurable	PendSV	Pendable request for system service	0x0000_0038
	6	Configurable	SysTick	System tick timer	0x0000_003C
0	7	Configurable	WWDT	Window timer interrupt	0x0000_0040
1	8	Configurable	PVM	PVM interrupt linked to EXINT	0x0000_0044
2	9	Configurable	TAMPER	Tamper detection interrupt	0x0000_0048
3	10	Configurable	ERTC_WKUP	ERTC wakeup interrupt	0x0000_004C
4	11	Configurable	FLASH	Flash global interrupt	0x0000_0050
5	12	Configurable	CRM	Clock and Reset manage (CRM) interrupt	0x0000_0054
6	13	Configurable	EXINT0	EXINT line 0 interrupt	0x0000_0058
7	14	Configurable	EXINT1	EXINT line 1 interrupt	0x0000_005C
8	15	Configurable	EXINT2	EXINT line 2 interrupt	0x0000_0060
9	16	Configurable	EXINT3	EXINT line 3 interrupt	0x0000_0064
10	17	Configurable	EXINT4	EXINT line 4 interrupt	0x0000_0068
11	18	Configurable	DMA1 channel 1	DMA1 channel 1 global interrupt	0x0000_006C
12	19	Configurable	DMA1 channel 2	DMA1 channel 2 global interrupt	0x0000_0070
13	20	Configurable	DMA1 channel 3	DMA1 channel 3 global interrupt	0x0000_0074
14	21	Configurable	DMA1 channel 4	DMA1 channel 4 global interrupt	0x0000_0078
15	22	Configurable	DMA1 channel 5	DMA1 channel 5 global interrupt	0x0000_007C
16	23	Configurable	DMA1 channel 6	DMA1 channel 6 global interrupt	0x0000_0080
17	24	Configurable	DMA1 channel 7	DMA1 channel 7 global interrupt	0x0000_0084
18	25	Configurable	ADC1_2	ADC global interrupt	0x0000_0088
19	26	-	-	-	0x0000_008C
20	27	-	-	-	0x0000_0090
21	28	-	-	-	0x0000_0094
22	29	-	-	-	0x0000_0098
23	30	Configurable	EXINT9_5	EXINT line [9:5] interrupt	0x0000_009C
24	31	Configurable	TMR1_BRK_TMR9	TMR1 break interrupt and TMR9 global interrupt	0x0000_00A0
25	32	Configurable	TMR1_OVF _TMR10	TMR1 overflow interrupt and TMR10 global interrupt	0x0000_00A4
26	33	Configurable	TMR1_TRG_HALL _TMR11	TMR1 trigger, HALL interrupt and TMR11 global interrupt	0x0000_00A8

27	34	Configurable	TMR1_CH	TMR1 channel interrupt	0x0000_00AC
28	35	Configurable	TMR2	TMR2 global interrupt	0x0000_00B0
29	36	Configurable	TMR3	TMR3 global interrupt	0x0000_00B4
30	37	Configurable	TMR4	TMR4 global interrupt	0x0000_00B8
31	38	Configurable	I2C1_EVT	I^2C1 event interrupt	0x0000_00BC
32	39	Configurable	I2C1_ERR	I^2C1 error interrupt	0x0000_00C0
33	40	Configurable	I2C2_EVT	I^2C2 event interrupt	0x0000_00C4
34	41	Configurable	I2C2_ERR	I^2C2 error interrupt	0x0000_00C8
35	42	Configurable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	Configurable	SPI2_I2S2EXT	SPI2 and IS2EXT global interrupt	0x0000_00D0
37	44	Configurable	USART1	USART1 global interrupt	0x0000_00D4
38	45	Configurable	USART2	USART2 global interrupt	0x0000_00D8
39	46	Configurable	USART3	USART3 global interrupt	0x0000_00DC
40	47	Configurable	EXINT15_10	EXINT line [15:10] interrupt	0x0000_00E0
41	48	Configurable	ERTCAlarm	ERTC alarm interrupt linked to EXINT	0x0000_00E4
42	49	Configurable	OTGFS1_WKUP	OTGFS1 Standby wakeup interrupt linked to EXINT	0x0000_00E8
43	50	Configurable	TMR8_BRK _TMR12	TMR8 break interrupt and TMR12 global interrupt	0x0000_00EC
44	51	Configurable	TMR8_OVF _TMR13	TMR8 overflow interrupt and TMR13 global interrupt	0x0000_00F0
45	52	Configurable	TMR8_TRG_HALL _TMR14	TMR8 trigger, HALL interrupt and TMR14 global interrupt	0x0000_00F4
46	53	Configurable	TMR8_CH	TMR8 channel interrupt	0x0000_00F8
47	54	-	-	-	0x0000_00FC
48	55	Configurable	XMC	XMC global interrupt	0x0000_0100
49	56	Configurable	SDIO1	SDIO1 global interrupt	0x0000_0104
50	57	Configurable	TMR5	TMR5 global interrupt	0x0000_0108
51	58	Configurable	SPI3_I2S3EXT	SPI3 and IS3EXT global interrupt	0x0000_010C
52	59	Configurable	USART4	USART4 global interrupt	0x0000_0110
53	60	Configurable	USART5	USART5 global interrupt	0x0000_0114
54	61	Configurable	TMR6_DAC	TMR6 global interrupt DAC1 and DAC2 underflow error interrupt	0x0000_0118
55	62	Configurable	TMR7	TMR7 global interrupt	0x0000_011C
56	63	Configurable	DMA2 channel 1	DMA2 channel 1 global interrupt	0x0000_0120
57	64	Configurable	DMA2 channel 2	DMA2 channel 2 global interrupt	0x0000_0124
58	65	Configurable	DMA2 channel 3	DMA2 channel 3 global interrupt	0x0000_0128
59	66	Configurable	DMA2 channel 4	DMA2 channel 4 global interrupt	0x0000_012C
60	67	Configurable	DMA2 channel 5	DMA2 channel 5 global interrupt	0x0000_0130
61	68	Configurable	EMAC[1]	Ethernet global interrupt	0x0000_0134
62	69	Configurable	EMAC_WKUP[1]	Ethernet wakeup interrupt linked to EXINT	0x0000_0138
63	70	-	-	-	0x0000_013C
64	71	-	-	-	0x0000_0140
65	72	-	-	-	0x0000_0144
66	73	-	-	-	0x0000_0148
67	74	Configurable	OTGFS1	OTGFS1 global interrupt	0x0000_014C
68	75	Configurable	DMA2 channel 6	DMA2 channel 6 global interrupt	0x0000_0150
69	76	Configurable	DMA2 channel 7	DMA2 channel 7 global interrupt	0x0000_0154
70	77	-	-	-	0x0000_0158

71	78	Configurable	USART6	USART6 global interrupt	0x0000_015C
72	79	Configurable	I2C3_EVT	I²C2 event interrupt	0x0000_0160
73	80	Configurable	I2C3_ERR	I²C2 error interrupt	0x0000_0164
74	81	-	-	-	0x0000_0168
75	82	-	-	-	0x0000_016C
76	83	-	-	-	0x0000_0170
77	84	-	-	-	0x0000_0174
e	85	-	-	-	0x0000_0178
79	86	-	-	-	0x0-000_017C
80	87	-	-	-	0x0000_0180
81	88	Configurable	FPU	FPU exception interrupt	0x0000_0184
82	89	Configurable	USART7	USART7 global interrupt	0x0000_0188
83	90	Configurable	USART8	USART8 global interrupt	0x0000_018C
84	91	Configurable	SPI4	SPI4 global interrupt	0x0000_0190
85	92	Configurable	I2SF5	I²SF5 global interrupt	0x0000_0194
86	93	-	-	-	0x0000_0198
87	94	-	-	-	0x0000_019C
88	95	-	-	-	0x0000_01A0
89	96	-	-	-	0x0000_01A4
90	97	-	-	-	0x0000_01A8
91	98	-	-	-	0x0000_01AC
92	99	Configurable	QSPI1	QSPI1 global interrupt	0x0000_01B0
93	100	-	-	-	0x0000_01B4
94	101	Configurable	DMAMUX	DMAMUX overflow interrupt	0x0000_01B8
95	102	-	-	-	0x0000_01BC
96	103	-	-	-	0x0000_01C0
97	104	-	-	-	0x0000_01C4
98	105	-	-	-	0x0000_01C8
99	106	-	-	-	0x0000_01CC
100	107	-	-	-	0x0000_01D0
101	108	-	-	-	0x0000_01D4
102	109	-	-	-	0x0000_01D8
103	110	Configurable	ACC	ACC global interrupt	0x0000_01DC

Note: AT32F457 supports EMAC and EMAC_WKUP interrupts, while AT32F455/AT32F456 do not.

1.1.4 System Tick (SysTick)

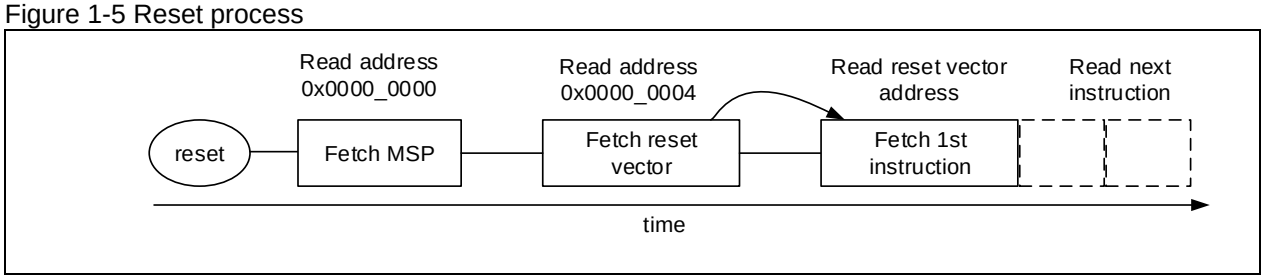
The System Tick is a 24-bit downcounter. It will be reloaded with the initial value automatically when it is decremented to zero. It can generate periodic interrupts, so it is often used as multi-task scheduling counter for embedded operating system, and also to call the periodic tasks for non-embedded system. The System Tick calibration value is fixed to 9000, which gives a reference time base of 1 ms when the System Tick clock is set to 9 MHz.

1.1.5 Reset

The processor reads the first two words from the CODE memory after a system reset and before program execution.

- Get the initial value of the main stack pointer (MSP) from address 0x0000_0000
- Get the initial value of the program counter (PC) from address 0x0000_0004. This value is a reset vector and LSB must be 1. Then, take the instructions from the address corresponding to this value.

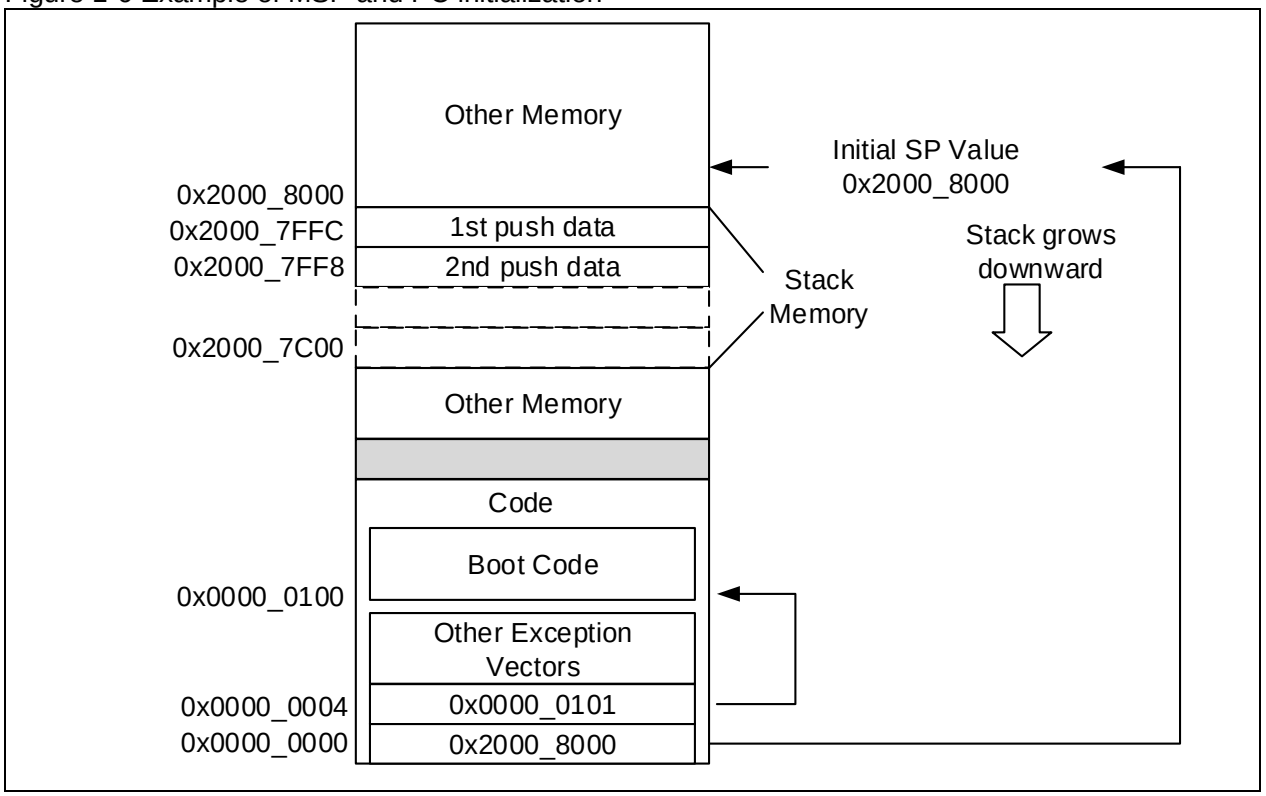
Figure 1-5 Reset process



Cortex®-M4F uses a full stack that increases downward, so the initial value of the main stack pointer (MSP) must be the end address of the stack memory plus 1. For example, if the stack area is set between 0x2000_7C00 and 0x2000_7FFF, then the initial value of MSP must be defined as 0x2000_8000.

The vector table follows the initial value of MSP. Cortex®-M4F operates in Thumb state, and thus each value in the vector table must set the LSB to 1. In Figure 1-6, 0x0000_0101 is used to represent 0x0000_0100. After the instruction at 0x0000_0100 is executed, the program starts running formally. Before that, it is a must for initializing MSP, because the first instruction may be interrupted by NMI or other faults before being executed. After the completion of MSP initialization, it is ready to prepare stack room for its service routines.

Figure 1-6 Example of MSP and PC initialization



In the AT32F455/456/457 series, the main Flash memory, boot memory or SRAM can be remapped to the CODE area between 0x0000_0000 and 0x07FF_FFFF. The BOOT1 and BOOT0 are used to set the specific memory from which CODE starts.

{BOOT1，BOOT0}=00/10: CODE starts from the main Flash memory

{BOOT1，BOOT0}=01: CODE starts from the boot memory

{BOOT1，BOOT0}=11: CODE starts from SRAM

After a system reset or when leaving from Standby mode, the pin values of both BOOT1 and BOOT0 will be relatched.

Boot memory contains an embedded Bootloader program that provides not only Flash programming function through USART, I²C, SPI, CAN or USB interface, but also provides extra firmware including communication protocol stacks that can be called for use by software developers through API.

1.2 List of abbreviations for registers

Table 1-4 List of abbreviations for registers

Register t	Description
rw	Software can read and write to this bit.
ro	Software can only read this bit
wo	Software can only write to this bit. Reading it returns to its reset value
rrc	Software can read this bit. Reading this bit automatically clears it.
rw0c	Software can read this bit and clear it by writing 0. Writing 1 has n
rw1c	Software can read this bit and clear it by writing 1. Writing 0 has n
rw1s	Software can read this bit and set it by writing 1. Writing 0 has no effe
tog	Software can read this bit and toggle it by writing 1. Writing 0 has n
rwt	Software can read this bit. Writing any value will trigger an event.
resd	Reserved

1.3 Device characteristics information

Table 1-5 Base address and reset value of registers

Register abbr.	Base address	Reset value
F_SIZE	0x1FFF F7E0	0xXXXX
UID[31:0]	0x1FFF F7E8	0xXXXX XXXX
UID[63:32]	0x1FFF F7EC	0xXXXX XXXX
UID[95:64]	0x1FFF F7F0	0xXXXX XXXX

1.3.1 Flash memory size register

This register contains the information about Flash memory size.

Bit	Name	Reset value	Type	Description
Bit 15:0	F_SIZE	0xXXXX	ro	Flash size, in terms of Kbyte For example, 0x0040 = 64 Kbytes

1.3.2 Device electronic signature

The device electronic signature contains the memory size and the unique device ID (96 bits). It is stored in the information block of the Flash memory. The 96-bit ID is unique for any device, and cannot be altered by users. It can be used for the following:

- Serial number, such as USB string serial number
- Part of security keys

Bit	Name	Reset value	Type	Description
Bit 31:0	UID[31:0]	0xXXXX XXXX	ro	UID for bit31 to bit0

Bit	Name	Reset value	Type	Description
Bit 31:0	UID[63:32]	0xXXXX XXXX	ro	UID for bit63 to bit32

Bit	Name	Reset value	Type	Description
Bit 31:0	UID[95:64]	0xXXXX XXXX	ro	UID for bit95 to bit64

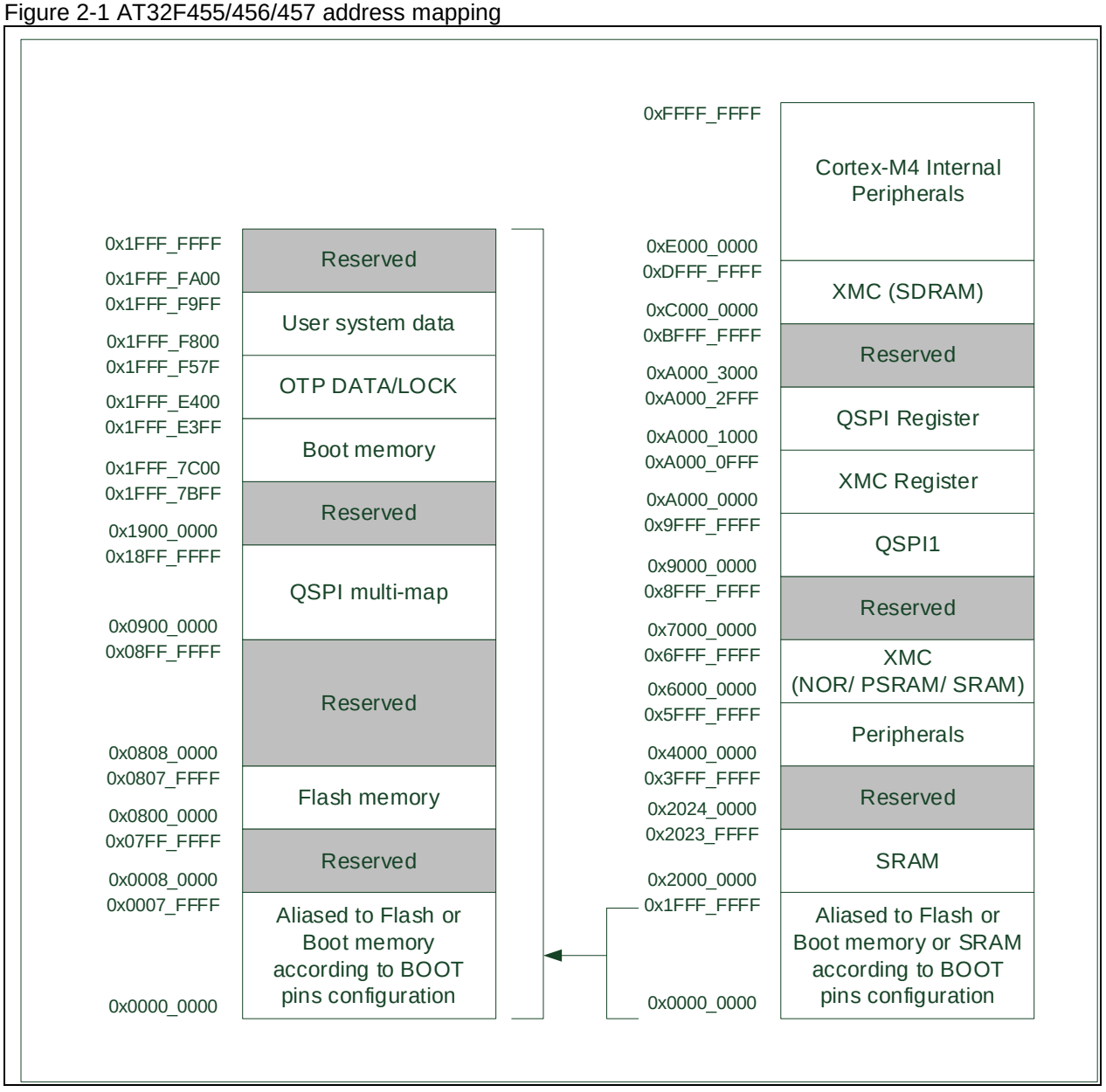
Note: UID[95:88] is Series ID, which is 0x15 for AT32F455, 0x16 for AT32F456 and 0x17 for AT32F457.

2 Memory resources

2.1 Internal memory address map

Internal memory contains program memory (Flash), data memory (SRAM), peripheral registers and core registers. Their respective address mapping are shown in Figure 2-1.

Figure 2-1 AT32F455/456/457 address mapping



2.2 Flash memory

AT32F455/456/457 series provide up to 512 KB of on-chip Flash memory, supporting aa single-cycle 32-bit read operation.

Refer to Chapter 5 for more details about Flash memory controller and register configuration.

Flash memory organization (512 KB)

The main memory contains bank 1 (512 Kbytes), including 256 sectors, 2 Kbytes per sector.

Table 2-1 Flash memory organization (512 KB)

Bank		Name	Address range
Main memory	Bank 1 512 KB	Sector 0	0x0800 0000 – 0x0800 07FF
		Sector 1	0x0800 0800 – 0x0800 0FFF
		Sector 2	0x0800 1000 – 0x0800 17FF
		…	…
		Sector 255	0x0807 F800 – 0x0807 FFFF
Information block		26 KB bootloader	0x1FFF 7C00 – 0x1FFF E3FF
		OTP 4KB	0x1FFF E400 – 0x1FFF F3FF
		LOCK 128B	0x1FFF F500 – 0x1FFF F57F
		512 B user system data	0x1FFF F800 – 0x1FFF F9FF

Flash memory organization (256 KB)

The main memory contains bank 1 (256 Kbytes), including 128 sectors, 2 Kbytes per sector.

Table 2-2 Flash memory organization (256 KB)

Bank		Name	Address range
Main memory	Bank 1 256 KB	Sector 0	0x0800 0000 – 0x0800 07FF
		Sector 1	0x0800 0800 – 0x0800 0FFF
		Sector 2	0x0800 1000 – 0x0800 17FF
		…	…
		Sector 127	0x0803 F800 – 0x0803 FFFF
Information block		26 KB bootloader	0x1FFF 7C00 – 0x1FFF E3FF
		OTP 4KB	0x1FFF E400 – 0x1FFF F3FF
		LOCK 128B	0x1FFF F500 – 0x1FFF F57F
		512 B user system data	0x1FFF F800 – 0x1FFF F9FF

2.3 SRAM memory

The AT32F455/456/457 series contain a 144-KB on-chip SRAM that starts at the address of 0x2000_0000. It can be accessed by bytes, half-words (16-bit) or words (32-bit).

The odd parity checking of SRAM can be enabled or disabled by setting the nRAM_PRT_CHK bit in the Flash user data system area. Once enabled, the maximum SRAM size available is only 128KB, with the remaining 16KB used to store odd parity checking result.

In addition, when data are written to SRAM, odd parity check is automatically calculated on a 1-KB unit level by hardware which will then store the generated 1-bit odd parity check bit into SRAM. When reading, the hardware will automatically check the integrity of the result, set NMI bit if an error occurs, and report the error state to the bit 8 in the SCFG_CFG2 register. Enabling the SRAM_OPERR_LK bit can allow the SRAM odd parity check error state to be connected to the brake input of TMR1/TMR9/10/11/13/14.

Note: SRAM odd parity check applies to the first 64KB of the SRAM only. To enable SRAM odd parity check, it is necessary to initialize the whole SRAM to prevent the occurrence of odd parity check error.

2.4 Peripheral address map

Table 2-3 Peripheral boundary address

Bus	Boundary address	Peripherals
AHB	0xC000 0000 - 0xDFFF FFFF	XMC_MEM (SDRAM)
	0xE000 0000 - 0xFFFF FFFF	Reserved
	0xA000 1400 - 0xBFFF FFFF	Reserved
	0xA000 1000 - 0xA000 13FF	QSPI1 Register
	0xA000 0000 - 0xA000 0FFF	XMC Register
	0x9000 0000 - 0x9FFF FFFF	QSPI1_MEM
	0x6000 0000 - 0x8FFF FFFF	XMC_MEM
	0x5004 0000 - 0x5FFF FFFF	Reserved
	0x5000 0000 - 0x5003 FFFF	OTG_FS1
	0x4008 0000 - 0x4FFF FFFF	Reserved
	0x4002 A000 - 0x4007 FFFF	Reserved
	0x4002 8000 - 0x4002 9FFF	EMAC
	0x4002 6800 - 0x4002 7FFF	Reserved
	0x4002 6400 - 0x4002 67FF	DMA2
	0x4002 6000 - 0x4002 63FF	DMA1
	0x4002 5000 - 0x4002 5FFF	Reserved
	0x4002 4C00 - 0x4002 4FFF	SDIO1
	0x4002 4000 - 0x4002 4BFF	Reserved
	0x4002 3C00 - 0x4002 3FFF	Flash memory interface (FLASH)
	0x4002 3800 - 0x4002 3BFF	Clock and reset manage (CRM)
	0x4002 3400 - 0x4002 37FF	Reserved
	0x4002 3000 - 0x4002 33FF	CRC
	0x4002 2000 - 0x4002 2FFF	Reserved
	0x4002 1C00 - 0x4002 1FFF	GPIO port H
	0x4002 1800 - 0x4002 1BFF	GPIO port G
	0x4002 1400 - 0x4002 17FF	GPIO port F
	0x4002 1000 - 0x4002 13FF	GPIO port E

	0x4002 0C00 - 0x4002 0FFF	GPIO port D
	0x4002 0800 - 0x4002 0BFF	GPIO port C
	0x4002 0400 - 0x4002 07FF	GPIO port B
	0x4002 0000 - 0x4002 03FF	GPIO port A
APB2	0x4001 8000 - 0x4001 FFFF	Reserved
	0x4001 7C00 - 0x4001 7FFF	I²S3EXT
	0x4001 7800 - 0x4001 7BFF	I²S2EXT
	0x4001 7400 - 0x4001 77FF	ACC
	0x4001 7000 - 0x4001 73FF	Reserved
	0x4001 6C00 - 0x4001 6FFF	Reserved
	0x4001 6800 - 0x4001 6BFF	Reserved
	0x4001 6400 - 0x4001 67FF	Reserved
	0x4001 6000 - 0x4001 63FF	Reserved
	0x4001 5C00 - 0x4001 5FFF	Reserved
	0x4001 5800 - 0x4001 5BFF	Reserved
	0x4001 5400 - 0x4001 57FF	Reserved
	0x4001 5000 - 0x4001 53FF	I²SF5
	0x4001 4C00 - 0x4001 4FFF	Reserved
	0x4001 4800 - 0x4001 4BFF	TMR11 timer
	0x4001 4400 - 0x4001 47FF	TMR10 timer
	0x4001 4000 - 0x4001 43FF	TMR9 timer
	0x4001 3C00 - 0x4001 3FFF	EXINT
	0x4001 3800 - 0x4001 3BFF	SCFG
	0x4001 3400 - 0x4001 37FF	SPI4/I2S4
	0x4001 3000 - 0x4001 33FF	SPI1/I2S1
	0x4001 2C00 - 0x4001 2FFF	Reserved
	0x4001 2800 - 0x4001 2BFF	Reserved
	0x4001 2400 - 0x4001 27FF	Reserved
	0x4001 2000 - 0x4001 23FF	ADC1_2
	0x4001 1C00 - 0x4001 1FFF	Reserved
	0x4001 1800 - 0x4001 1BFF	Reserved
	0x4001 1400 - 0x4001 17FF	USART6
	0x4001 1000 - 0x4001 13FF	USART1
	0x4001 0C00 - 0x4001 0FFF	Reserved
	0x4001 0800 - 0x4001 0BFF	Reserved
	0x4001 0400 - 0x4001 07FF	TMR8 timer
	0x4001 0000 - 0x4001 03FF	TMR1 timer
APB1	0x4000 8000 - 0x4000 FFFF	Reserved
	0x4000 7C00 - 0x4000 7FFF	USART8
	0x4000 7800 - 0x4000 7BFF	USART7
	0x4000 7400 - 0x4000 77FF	DAC

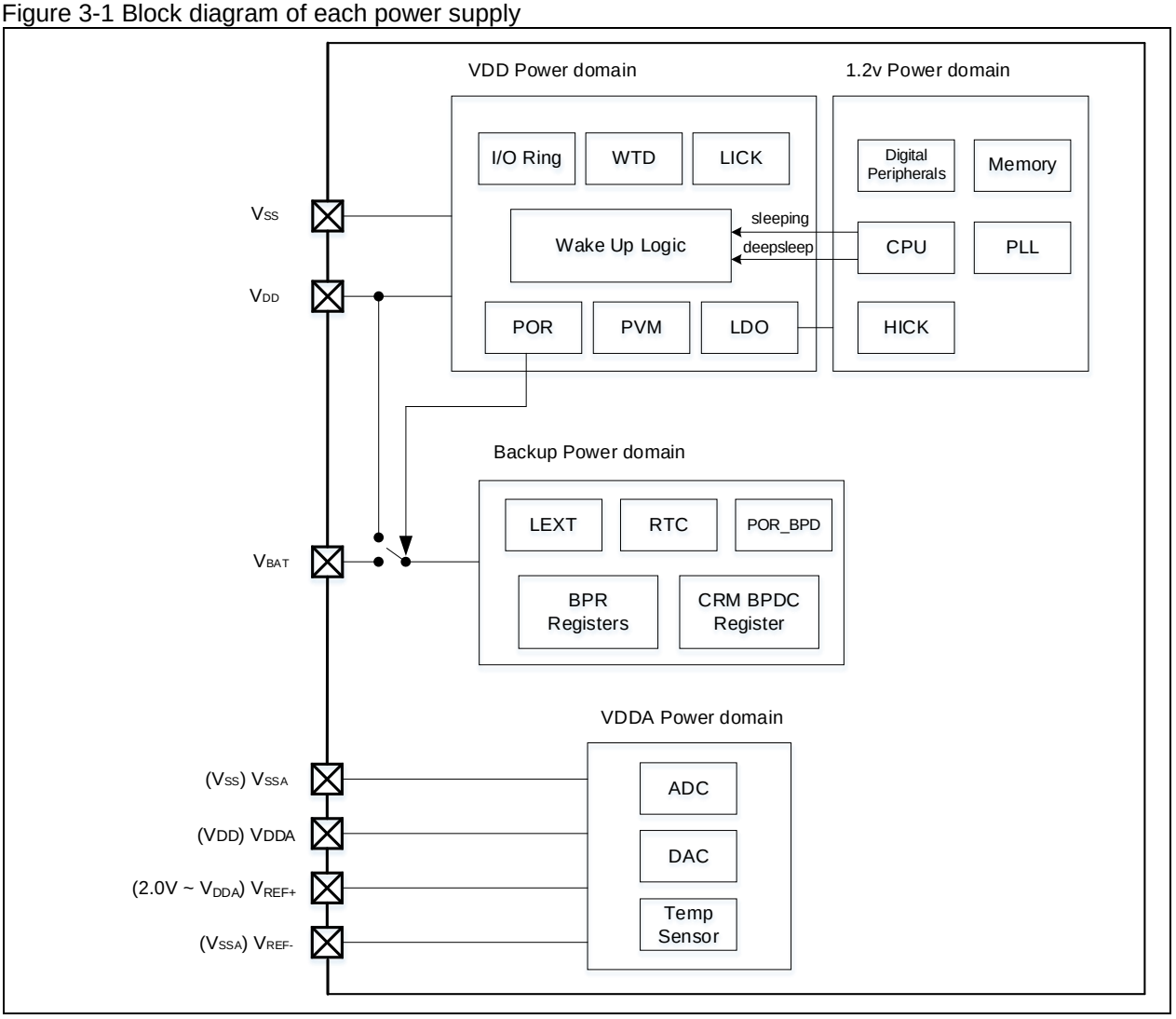
	0x4000 7000 - 0x4000 73FF	Power control (PWC)
	0x4000 6C00 - 0x4000 6FFF	CAN3
	0x4000 6800 - 0x4000 6BFF	CAN2
	0x4000 6400 - 0x4000 67FF	CAN1
	0x4000 6000 - 0x4000 63FF	Reserved
	0x4000 5C00 - 0x4000 5FFF	I^2C3
	0x4000 5800 - 0x4000 5BFF	I^2C2
	0x4000 5400 - 0x4000 57FF	I^2C1
	0x4000 5000 - 0x4000 53FF	USART5
	0x4000 4C00 - 0x4000 4FFF	USART4
	0x4000 4800 - 0x4000 4BFF	USART3
	0x4000 4400 - 0x4000 47FF	USART2
	0x4000 4000 - 0x4000 43FF	Reserved
	0x4000 3C00 - 0x4000 3FFF	SPI3/I2S3
	0x4000 3800 - 0x4000 3BFF	SPI2/I2S2
	0x4000 3400 - 0x4000 37FF	Reserved
	0x4000 3000 - 0x4000 33FF	Watchdog timer (WDT)
	0x4000 2C00 - 0x4000 2FFF	Window watchdog timer (WWDT)
	0x4000 2800 - 0x4000 2BFF	ERTC
	0x4000 2400 - 0x4000 27FF	Reserved
	0x4000 2000 - 0x4000 23FF	TMR14 timer
	0x4000 1C00 - 0x4000 1FFF	TMR13 timer
	0x4000 1800 - 0x4000 1BFF	TMR12 timer
	0x4000 1400 - 0x4000 17FF	TMR7 timer
	0x4000 1000 - 0x4000 13FF	TMR6 timer
	0x4000 0C00 - 0x4000 0FFF	TMR5 timer
	0x4000 0800 - 0x4000 0BFF	TMR4 timer
	0x4000 0400 - 0x4000 07FF	TMR3 timer
	0x4000 0000 - 0x4000 03FF	TMR2 timer

3 Power control (PWC)

3.1 Introduction

For AT32F455/456/457 series, the operating voltage supply is 2.4 V ~ 3.6 V, with an operating temperature range of -40~+105 ºC. To reduce power consumption, this series provides three types of power saving modes, including Sleep, Deepsleep and Standby modes so as to achieve the best tradeoff among the conflicting demands of CPU operating time, speed and power consumption. The AT32F455/456/457 series have three power domains, including VDD/VDDA domain, 1.2 V domain and battery powered domain. The VDD/VDDA domain is supplied directly by external power, the 1.2 V domain is powered by an embedded LDO in the VDD/VDDA domain, and the battery powered domain is supplied via V_{BAT}.

Figure 3-1 Block diagram of each power supply



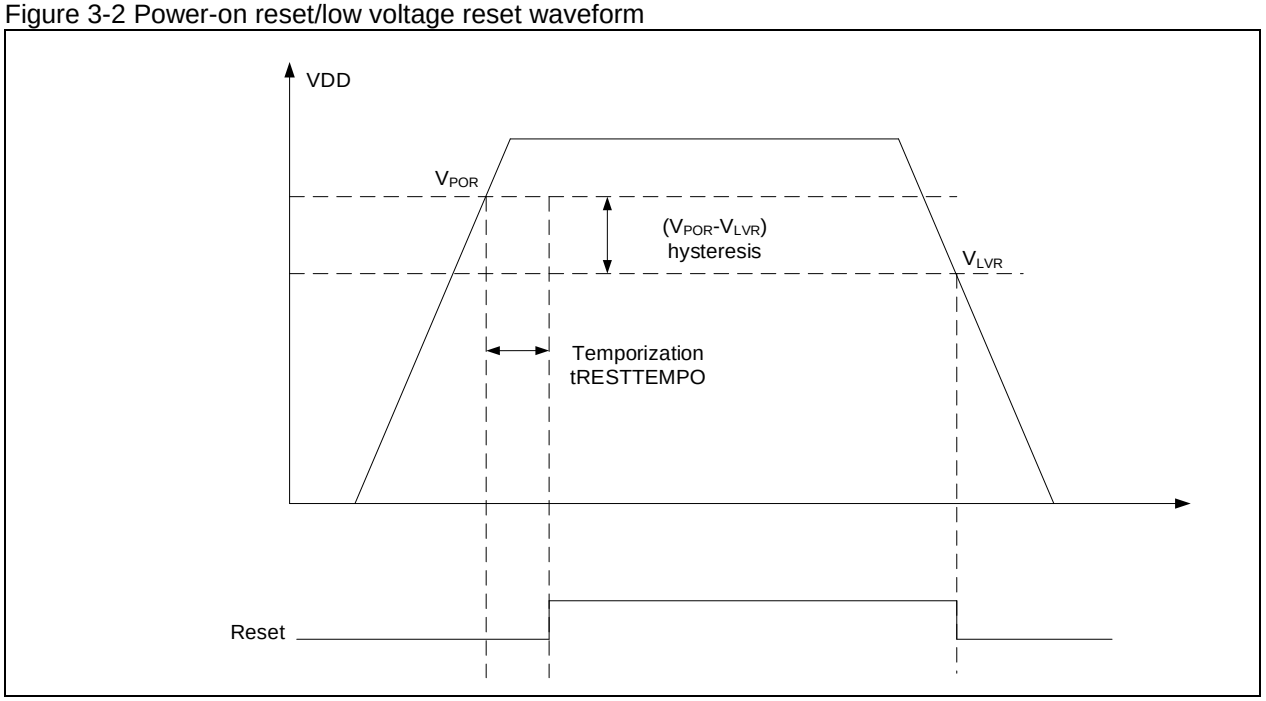
3.2 Main features

- Three power domains: VDD/VDDA domain, 1.2 V domain and battery powered domain
- Three types of power saving modes: Sleep mode, Deepsleep mode and Standby mode
- Internal voltage regulator supplies 1.2 V voltage source for the core domain
- Power voltage detector is provided to issue an interrupt or event when the supply voltage is lower or higher than the programmed threshold
- Battery supplied domain is supplied via V_{BAT} when VDD supply is OFF
- VDD/VDDA applies separated digital and analog module to reduce noise on external power

3.3 POR/LVR

A POR analog module embedded in the VDD/VDDA domain is used to generate a power reset. The power reset signal is released at V_{POR} when the VDD is increased from 0 V to the operating voltage, or it is triggered at V_{LVR} when the VDD frops from the operating voltage to 0 V. During the power-on reset period, hysteresis occurs in power-on reset (POR) and low voltage reset (LVR).

Figure 3-2 Power-on reset/low voltage reset waveform

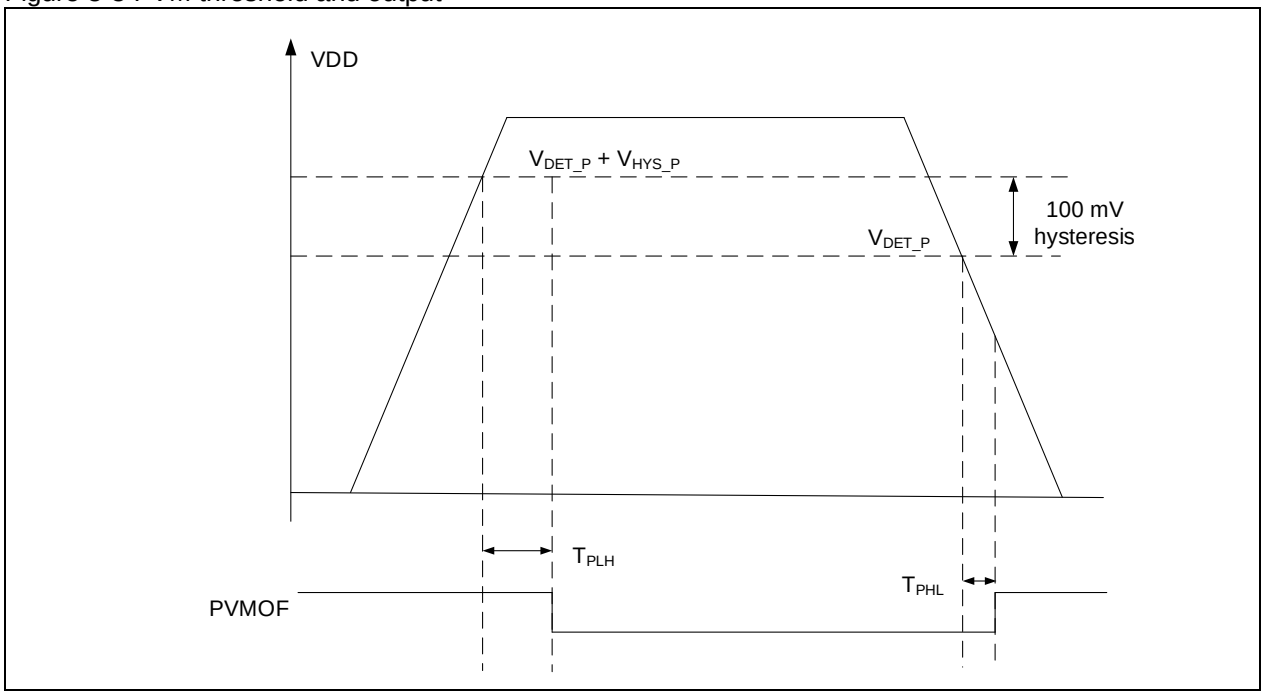


3.4 Power voltage monitor (PVM)

The PVM is used to monitor the power supply variations. It is enabled by setting the PVMEN bit in the power control register (PWC_CTRL), and the threshold value for voltage monitor is selected with the PVMSEL[2:0].

After PVM is enabled, the comparison result between VDD and the programmed threshold is indicated by the PVMOF bit in the PWC_CTRLSTS register, with the hysteresis voltage VHYS_P being 100 mV. The PVM interrupt will be generated through EXINT line 16 when VDD rises above the PVM threshold.

Figure 3-3 PVM threshold and output



3.5 Power domain

1.2 V domain

The 1.2 V core domain includes a CPU core, SRAM, embedded digital peripherals and Phase Locked Loop (PLL). Such power domain is supplied by LDO (voltage regulator).

VDD/VDDA domain

VDD/VDDA domain includes VDD domain and VDDA domain. The VDD domain contains I/O circuit, power-saving mode wakeup circuit, watchdog timer (WDT), power-on reset/low voltage reset (POR/LVR), LDO and all PAD circuits except PC13, PC14 and PC15. The VDDA domain contains an ADC/DAC (AD/DA converter), temperature sensor and so on.

Typically, to ensure a better accuracy of ADC/DAC at a low voltage, the digital circuit is supplied by VDD while the analog circuit is powered by VDDA. On 64-pin packages and packages with less pins, the external reference voltage VREF+ and VREF- are connected to VDDA and VSSA, respectively.

In Run Mode, the LDO supplies full power to the 1.2 V core domain and outputs 1.2 V voltage, by default. The LDO output voltage is selected through the PWC_LDOOV register. the maximum operating frequency for the system depends on the selected output voltage. Refer to *AT32F455/456/457 Series Datasheet* for details. The LDO output voltage is changeable only when the HEXT or HICK is used as system clock.

LDO output voltage regulation

1) Select HICK or HEXT as system clock;
2) Change LDO voltage (LDOOVSEL[1:0]);
3) Set the FLASH_PSR register;
4) Set the target frequency for PLL-related registers, enable PLL, and wait for PLL_STBL;
5) Set pre-division factors for AHB and APB;
6) Enable auto step-by-step frequency switch function when PLL frequency is greater than 108 MHz;
7) Switch the system clock to PLL.

Battery powered domain

The battey powered domain contains ERTC circuit, LEXT oscillator, PC13, PC14 and PC15, which is powered by either VDD or VBAT pin. When the VDD is cut off, the battery powered domain is automatically switched to VBAT pin to ensure that ERTC can work normally.

1) When the battery powered domain is powered by VDD, the PC13 can be used as a general-purpose I/O, tamper pin, ERTC calibration clock, ERTC alarm or second output, while the PC14 and PC15 can be used as a GPIO or LEXT pin (as an I/O power, PC13 to PC15 must be limited below 2 MHz, and to the maximum load of 30pF, and these I/O ports must not be used as current sourecs).
2) When the battery powered domain is powered by V_{BAT}, the PC13 can be used as a tamper pin, ERTC alarm or second output, while the PC14 and PC15 can only be used as a LEXT pin.

The switch of the battery powered domain will not be disconnected from V_{BAT} because of the VDD being at its rising phrase or due to VDD low voltage reset. If the power switch has not been switched to the VDD when the VDD is powered on quickly, it is recommended to add a low voltage drop diode between VDD and V_{BAT} in order to prevent the currents of VDD from being injected to VBAT. If there is no external battery in the application, it is better to connect the V_{BAT} to a 100 nF ceramic filter capacitor that is externally connected to VDD.

3.6 Power saving modes

When the CPU does not need to be kept running, there are three types of low-power modes available (Sleep mode, Deepsleep mode and Standby mode) to save power. Users can select the mode that gives the best compromise according to the low-power consumption, short startup time, and available wakeup sources. In addition, the power consumption in Run mode can be reduced by slowing down the system clocks or gating the clocks to the APB and AHB peripherals when they are not used.

Sleep mode

The Sleep mode is entered by executing WFI or WFE instruction. There are two options to select the Sleep mode entry mechanism through the SLEEPONEXIT bit in the Cortex®-M4F system control register.

SLEEP-NOW mode:

When SLEEPDEEP=0 and SLEEPONEXIT=0, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.

SLEEP-ON-EXIT mode:

When SLEEPDEEP=0 and SLEEPONEXIT=1, the MCU enters Sleep mode as soon as the system exits the lowest-priority interrupt service routine by executing the WFI instruction.

In Sleep mode, all clocks and LDO work normally except CPU clocks (stepped), and all I/O pins keep the same state as in Run mode. The LDO provides a 1.2 V power (for CPU core, memory and embedded peripherals) as it is in normal power consumption mode.

1) If the WFI is executed to enter Sleep mode, any peripheral interrupt can wake up the device from Sleep mode.
2) If the WFE is executed to enter Sleep mode, the MCU exits Sleep mode as soon as an event occurs. The wakeup event can be generated by the following:

- Enabling a peripheral interrupt (it is not enabled in the NVIC) and enabling the SEVONPEND bit. When the MCU resumes, the peripheral interrupt pending bit and NVIC channel pending bit must be cleared.
- Configuring an internal EXINT line as an event mode to generate a wakeup event.

The wakeup time required by a WFE instruction is the hortest, since no time is wasted on interrupt entry/exit.

Deepsleep mode

Deepsleep mode is entered by setting the SLEEPDEEP bit in the Cortex®-M4F system control register and clearing the LPSEL bit in the PWC_CTRL register before executing WFI or WFE instructions.

The LDO status is selected by setting the VRSEL bit in the power control register (PWC_CTR). When VRSEL=0, the LDO works in normal mode. When VRSEL=1, the LDO is set in low-power consumption mode.

In addition, in Deepsleep mode, with VRSEL=1 and LDO in low-power mode, the system power consumption can be further reduced by setting the VREXLPEN bit.

In Deepsleep mode, all clocks in 1.2 V domain are stopped, and both HICK and HEXT oscillators are disabled. The LDO supplies power to the 1.2 V domain in normal mode or low-power mode. All I/O pins keep the same state as in Run mode. SRAM and register contents are preserved.

1) If the WFI is executed to enter Deepsleep mode, the interrupt generated on any external interrupt line in Interrupt mode can wake up the system from Deepsleep mode.
2) If the WFE is executed to enter Deepsleep mode, the event generated on any external interrupt line in Event mode can wake up the system from Deepsleep mode.

When the MCU exits the Deepsleep mode, the HICK RC oscillator is enabled and selected as system clock after stabilization. When the LDO operates in low-power mode, an additional wakeup delay is incurred for the reason that the LDO must be stabilized before the system is waken from the Deepsleep mode.

Low-power Deepsleep LDO voltage regulation process (note that Sleep and Standby modes have no such limits)

1) Select HICK as system clock
2) Change LDO voltage to 1.0 V by setting the LDOOVSEL[1:0] bit
3) Set the LDO in low-power mode by setting the VRSEL bit
4) System enters Deepsleep state
5) System exits Deepsleep state (if wakeup conditions are met)
6) Change LDO voltage by setting the LDOOVSEL[1:0] bit
7) Set the FLASH_PSR register
8) If the HEXT is used as PLL clock source, enable HEXT and wait for HEXTSTBL
9) Set the target frequency for PLL-related registers
10) Enable PLL and wait for PLL_STBL
11) Set pre-division factors for AHB and APB
12) Enable auto step-by-step frequency switch function when PLL frequency is greater than 108 MHz

13) Switch system clock to PLL

Note: If the clock, after low-power mode is waken up, needs to keep the same state as in low-power mode, the above-mentioned steps 3/9/11 can be ignored.

Standby mode

Standby mode can achieve the lowest power consumption for the device. In this mode, the LDO is disabled. The whole 1.2 V domain, PLL, HICK and HEXT oscillators are also powered off. SRAM and register contents are lost. Only battery powered registers and standby circuitry remain supplied.

The Standby mode is entered by the following procedures:
- Set the SLEEPDEEP bit in the Cortex®-M4F system control register
- Set the LPSEL bit in the PWC_CTRL register
- Clear all wakeup event flags in the PWC_CTRLSTS register
- Execute a WFI or WFE instruction

In Standby mode, all I/O pins remain in high-impedance state except reset pins, tamper pins that are set as anti-tamper or calibration output, and the wakeup pins enabled.

The MCU leaves the Standby mode when an external reset (NRST pin), a WDT reset, ERTC periodic wakeup, ERTC timestamp, ERTC tamper event and an effective edge on the WKUP pin or the rising edge of an ERTC alarm event occurs.

Debug mode

By default, the debug connection is lost if the MCU enters Deepsleep mode or Standby mode while debugging. The reason is that the Cortex®-M4F core is no longer clocked. However, the software can be debugged even in the low-power mode by setting some configuration bits in the DEBUG control register (DEBUG_CTRL).

3.7 PWC registers

These peripheral registers can be accessed by half-words (16 bits) or words (32 bits).

Table 3-1 PWC register map and reset values

Register abbr.	Offset	Reset value
PWC_CTRL	0x00	0x0000 0000
PWC_CTRLSTS	0x04	0x0000 0000
PWC_CLR	0x08	0x0000 0000
PWC_LDOOV	0x10	0x0000 0012

3.7.1 Power control register (PWC_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30	SWP7POL	0x0	rw	Standby wake-up pin7 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 29	SWP6POL	0x0	rw	Standby wake-up pin6 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 28	SWP5POL	0x0	rw	Standby wake-up pin5 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 27	SWP4POL	0x0	rw	Standby wake-up pin4 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 23	Reserved	0x0	resd	Kept at its default value.

Bit 25	SWP2POL	0x0	rw	Standby wake-up pin2 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 24	SWP1POL	0x0	rw	Standby wake-up pin1 polarity 0: Rising edge trigger 1: Falling edge trigger
Bit 23	Reserved	0x0	resd	Kept at its default value.
Bit 22	SWPEN7	0x0	rw	Standby wake-up pin6 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 21	SWPEN6	0x0	rw	Standby wake-up pin6 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 20	SWPEN5	0x0	rw	Standby wake-up pin5 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 19	SWPEN4	0x0	rw	Standby wake-up pin4 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 18	Reserved	0x0	resd	Kept at its default value.
Bit 17	SWPEN2	0x0	rw	Standby wake-up pin2 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 16	SWPEN1	0x0	rw	Standby wake-up pin1 enable 0: Disabled (this pin can be used as a general-purpose I/O) 1: Enabled (this pin is forced in input pull-down

				mode, and no longer used as a general-purpose I/O) Note: This bit is cleared by hardware at system reset.
Bit 15:9	Reserved	0x0	resd	Kept at its default value.
Bit 8	BPWEN	0x0	rw	Battery powered domain write enable 0: Disabled 1: Enabled Note: After reset, the battery powered domain write access is disabled. To write, this bit must be set.
Bit 7:5	PVMSEL	0x0	rw	Power voltage monitoring boundary select 000: Unused, not configurable 001: 2.3 V 010: 2.4 V 011: 2.5 V 100: 2.6 V 101: 2.7 V 110: 2.8 V 111: 2.9 V
Bit 4	PVMEN	0x0	rw	Power voltage monitoring enable 0: Disabled 1: Enabled
Bit 3:2	Reserved	0x0	resd	Kept at its default value.
Bit 1	LPSEL	0x0	rw	Low power mode select when Cortex®-M4F is in Deepsleep mode 0: Enter Deepsleep mode 1: Enter Standby mode
Bit 0	VRSEL	0x0	rw	Voltage regulator state select in Deepsleep mode 0: Normal mode 1: Low-power consumption mode

3.7.2 Power control/status register (PWC_CTRLSTS)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x000000	resd	Kept at its default value.
Bit 15	SWEF	0x0	ro	Standby wake-up event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (on a wakeup event), and it is cleared by POR/LVR or a wakeup event. A wakeup event is generated by one of the following: At an ERTC alarm/tamper/timestamp/periodic auto wakeup event that enables an interrupt
Bit 14:10	Reserved	0x00	resd	Kept at its default value.
Bit 9	PVMOF	0x0	ro	Power voltage monitoring output flag 0: Power voltage is higher than the threshold

				1: Power voltage is lower than the threshold Note: The power voltage monitoring is stopped in Standby mode.
Bit 8	SEF	0x0	ro	Standby mode entry flag 0: Device is not in Standby mode 1: Device is in Standby mode Note: This bit is set by hardware (enter Standby mode) and cleared by POR/LVR or by setting the CLSEF bit.
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6	SW7EF	0x0	ro	Standby wake-up7 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW7EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 7 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.
Bit 5	SW6EF	0x0	ro	Standby wake-up6 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW6EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 6 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.
Bit 4	SW5EF	0x0	ro	Standby wake-up5 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW5EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 5 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.

Bit 3	SW4EF	0x0	ro	Standby wake-up4 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW4EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 4 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	SW2EF	0x0	ro	Standby wake-up2 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW2EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 2 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.
Bit 0	SW1EF	0x0	ro	Standby wake-up1 event flag 0: No wakeup event occurs 1: Wakeup event occurs Note: This bit is set by hardware (at a wakeup event), and it is cleared by POR/LVR or by setting the CLSW1EF bit. A wakeup event is generated by one of the following: When the rising/falling edge on the Standby wakeup pin 1 occurs; This Standby wakeup pin is enabled when ihe Standby wakeup pin level is high.

3.7.3 Power contrl flag clear register (PWC_CLR)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15	CLSWEF	0x0	wo	Clear SWEF flag 0: No effect 1: Clear SWEF flag Note: This bit is cleared by hardware after the SWEF is cleared. Reading this bit returns zero.
Bit 14:9	Reserved	0x00	resd	Kept at its default value.
Bit 8	CLSEF	0x0	wo	Clear SEF flag 0: No effect

				1: Clear SEF flag Note: This bit is cleared by hardware after the SEF is cleared. Reading this bit returns zero.
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6	CLSW7EF	0x0	wo	Clear SW7EF flag 0: No effect 1: Clear SW7EF flag Note: Clearing the SW7EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW7EF is cleared. Reading this bit returns zero.
Bit 5	CLSW6EF	0x0	wo	Clear SW6EF flag 0: No effect 1: Clear SW6EF flag Note: Clearing the SW6EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW6EF is cleared. Reading this bit returns zero.
Bit 4	CLSW5EF	0x0	wo	Clear SW5EF flag 0: No effect 1: Clear SW5EF flag Note: Clearing the SW5EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW5EF is cleared. Reading this bit returns zero.
Bit 3	CLSW4EF	0x0	wo	Clear SW4EF flag 0: No effect 1: Clear SW4EF flag Note: Clearing the SW4EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW4EF is cleared. Reading this bit returns zero.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	CLSW2EF	0x0	wo	Clear SW2EF flag 0: No effect 1: Clear SW2EF flag Note: Clearing the SW2EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW2EF is cleared. Reading this bit returns zero.
Bit 0	CLSW1EF	0x0	wo	Clear SW1EF flag 0: No effect 1: Clear SW1EF flag Note: Clearing the SW1EF flag requires about 2 system clocks. This bit is cleared by hardware after the SW1EF is cleared. Reading this bit returns zero.

3.7.4 LDO output voltage select register (PWC_LDOOV)

Bit	Name	Reset value	Type	Description
Bit 31:5	Reserved	0x0000000	resd	Kept at its default value.
Bit 4	VREXLPEN	0x1	rw	Voltage regulator extra low power mode enable This bit works together with the LPSEL and VRSEL bits in the PWC_CTRL register, and it is valid when VRSEL = 1 and the chip enters Deepsleep mode. 0: Disabled 1: Enabled Note: To enable extra low power mode, set the VREXLPEN bit before setting LPSEL and VRSEL bits. When this bit is set and the chip enters Deepsleep mode, the LDO drive strength is limied. In this case, it is not allowed to enable DEEPSLEEP_DEBUG for debugging.
Bit 3:2	Reserved	0x0	resd	Kept at its default value.
Bit 1:0	LDOOVSEL	0x02	rw	LDO output voltage select 01: 1.1 V 10: 1.2 V 11: 1.3 V

4 Clock and reset manage (CRM)

4.1 Clock

AT32F455/456/457 series provide different clock sources, including HEXT oscillator clock, HICK oscillator clock, PLL clock, LEXT oscillator clock and LICK oscillator clock.

Figure 4-1 AT32F455/456 clock tree

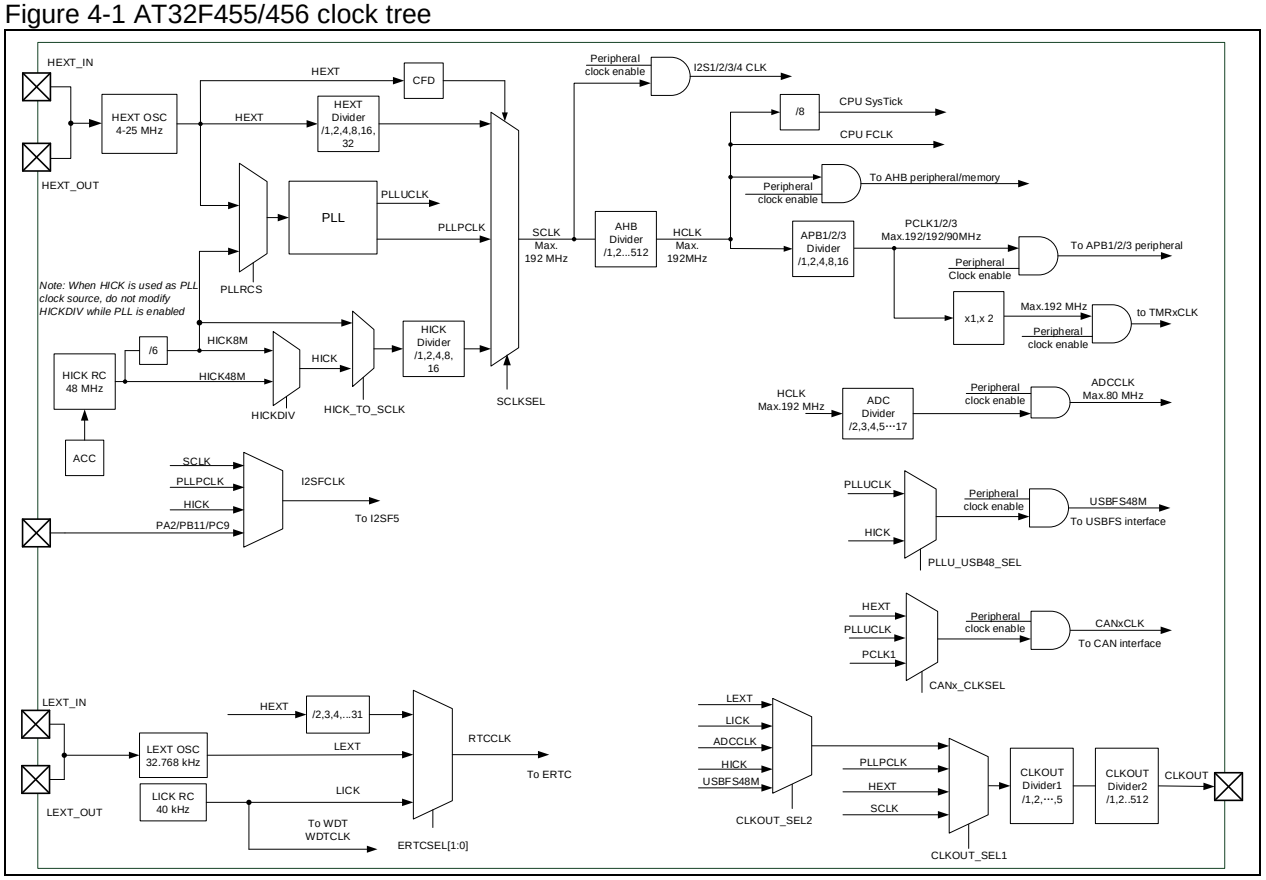
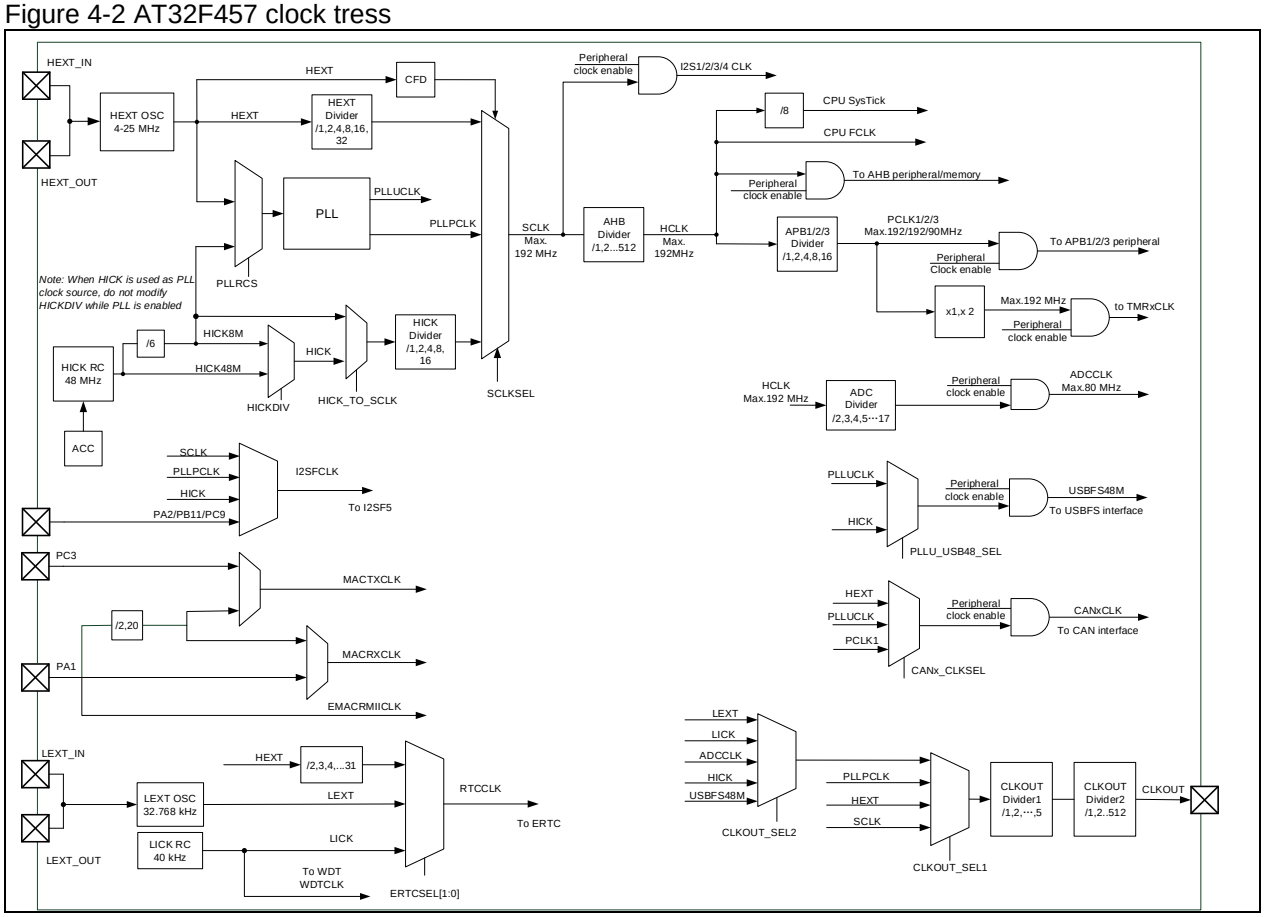


Figure 4-2 AT32F457 clock tress



AHB, APB1, APB2 and APB3 all support multiple frequency divisions. The AHB, APB1 and APB2 domains have a maximum frequency of 192 MHz, and APB3 domain is up to 90 MHz.

4.1.1 Clock sources

- High-speed external oscillator (HEXT)

The HEXT includes two clock sources: crystal/ceramic resonator and bypass clock.

The HEXT crystal/ceramic resonator is connected externally to a 4~25 MHz HEXT crystal that produces a highly accurate clock for the system. The HEXT clock signal is not released until it becomes stable.

An external clock source can be provided by HEXT bypass. Its frequency can be up to 25 MHz. The external clock signal should be connected to the HEXT_IN pin, and the HEXT_OUT pin can be released for GPIO control.

- High-speed internal clock (HICK)

The HICK oscillator is clocked by a high-speed RC in the microcontroller. The internal frequency of the HICK clock is 48 MHz. Although it is less accurate, its startup time is shorter than the HEXT crystal oscillator. The HICK clock frequency of each device is calibrated by ARTERY to ±1% accuracy (T_A=25°C) in factory. The factory-trimmed value is loaded into the HICKCAL[7:0] bit of the clock control register. The RC oscillator speed may be affected by voltage or temperature variations. Thus the HICK frequency can be trimmed by setting the HICKTRIM[5:0] bit in the clock control register.

The HICK clock signal is not released until it becomes stable.

- PLL clock

The HICK or HEXT clock can be used as an input clock source of the PLL. The PLL input clock, after being divided by a pre-divider internally, is sent to the VCO for frequency multiplication, and the VCO output frequency is output after being divided by a post-divider. At the same time, the clock after predivider must remain between 2 MHz and 16 MHz, and the VCO operating frequency must be kept between 500 MHz and 1200 MHz. The PLL must be configured before being enabled. The reason is that the configuration parameters cannot be changed once the PLL is enabled. The PLL clock signal is not released until it becomes stable.

PLL formula:

PLL output clock = PLL input clock x PLL frequency multiplication factor / (PLL pre-divider factor x PLL post-divider factor)

500MHz <= PLL input clock x PLL frequency multiplication factor / PLL pre-divider factor <= 1200MHz

2MHz <= PLL input clock / PLL pre-divider factor <= 16MHz

For example, when the PLL input clock is 25 MHz, the PLL output frequency = 25 x 192 / (5 x 5) = 192 MHz.

- Low-speed external oscillator (LEXT)

The LEXT oscillator provides two clock sources: LEXT crystal/ceramic resonator and LEXT bypass.

LEXT crystal/ceramic resonator:

The LEXT crystal/ceramic resonator provides a low-power and accurate 32.768 KHz low-speed clock source. The LEXT clock signal is not released until it becomes stable.

- LEXT bypass clock

In LEXT bypass mode, an external clock source with a frequency of 32.768 KHz can be provided. The external clock signal should be connected to the LEXT_IN pin, and the LEXT_OUT pin should be released for GPIO control.

- Low-speed internal RC oscillator (LICK)

The LICK oscillator is clocked by an internal low-speed RC oscillator. The clock frequency is between 30 KHz and 60 KHz. It acts as a clock source that can be kept running in Deepsleep mode and Standby mode for watchdog and auto wakeup unit.

The LICK clock signal is not released until it becomes stable.

4.1.2 System clock

After a system reset, the HICK oscillator is selected as system clock. The system clock can make flexible switch among HICK oscillator, HEXT oscillator and PLL clock. However, a switch from one clock source to another occurs only if the target clock source becomes stable. When the HICK oscillator is used directly or indirectly through the PLL as the system clock, it cannot be stopped.

4.1.3 Peripheral clock

Most peripherals use HCLK, PCLK1 or PCLK2 clock. The individual peripherals have their dedicated clocks.

System Tick timer (SysTick) is clocked by CPU FCLK (HCLK) or CPU Systick (HCLK/8).

ADC is clocked by HCLK divided by 2, 3, 4, 5…17.

The timers are clocked by APB1/2. In particular, if the APB prescaler is 1, the timer clock frequency is equal to that of APB1/2; otherwise, the timer clock frequency doubles that of the APB1/2 frequency.

The USB clock source can be switched between HICK and PLLU. If the HICK is selected as a clock source, it should be set as 48 MHz. If the PLLU is selected as a clock source, it should be configured as 48 MHz.

ERTC clock source includes divided HEXT oscillator, LEXT oscillator and LICK oscillator. Once the clock source is selected, it cannot be altered unless a reset of battery powered domain is performed. If the LEXT is used as an ERTC clock, the ERTC is not affected when the VDD is powered off. If the HEXT or LICK is selected as an ERTC clock, the ERTC state is not guaranteed when both HEXT and LICK are powered off.

Watchdog is clocked by LICK oscillator. If the watchdog is enabled by either hardware option or software access, the LICK oscillator is forced ON. The clock is provided to the watchdog only after the LICK oscillator temporization.

4.1.4 Clock fail detector

The clock fail detector (CFD) is designed to respond to HEXT clock failure when the HEXT is used as the system clock, directly or indirectly. If a failure is detected on the HEXT clock, a clock failure event is sent to the brake input of TMR1/9/10/11/12/13/14 and a CFD interrupt is generated. This interrupt is directly linked to CPU NMI so that the software can perform rescue operations. The NMI interrupt keeps executing until the CFD interrupt pending bit is cleared. This is way the CFD interrupt has to be cleared in the NMI service routine. The HEXT clock failure will result in a switch of the system clock to the HICK clock, the CFD to be disabled, HEXT clock to be stopped, and even PLL to be disabled if the HEXT is selected as system clock through PLL.

4.1.5 Auto step-by-step system clock switch

The automatic frequency switch is designed to ensure a smooth and stable switch of system frequency when the system clock source is switched from others to the PLL or when the AHB prescaler factor is changed from large to small. When the operational target is larger than 108 MHz, it is recommended to enable the automatic frequency switch. Once it is enabled, the AHB bus is halted by hardware till the completion of the switch. During this switch period, the DMA remain working, and the interrupt events are recorded and then handled by NVIC when the AHB bus resumes.

4.1.6 Internal clock output

The microcontroller allows the internal clock signal to be output to external CLKOUT pin. That is, ADCCLK, USB48M, SCLK, LICK, LEXT, HICK, HEXT and PLLCLK can be used as CLKOUT clock. When being used as the CLKOUT clock output pin, the corresponding GPIO port registers must be configured accordingly.

4.1.7 Interrupts

The microcontroller specifies a stable flag for each clock source. As a result, when a clock source is enabled, it is possible to determine if the clock is stable by checking the flag pertaining to the clock source. An interrupt request is generated when the interrupt corresponding to the clock source is enabled.

If a failure is detected on the HEXT clock, the CFD interrupt is generated. Such interrupt is directly linked to CPU NMI.

4.2 Reset

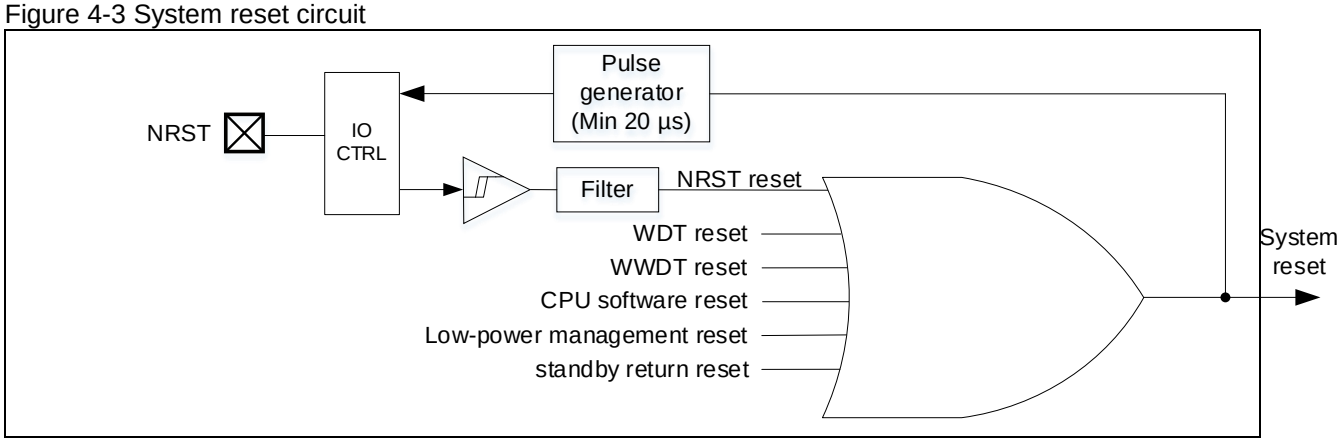
4.2.1 System reset

AT32F455/456/457 series provide the following system reset sources:

- NRST reset: on the external NRST pin
- WDT reset: watchdog overflow
- WWDT reset: window watchdog overflow
- CPU software reset: Cortex®-M4F software reset
- Low-power management reset: This type of reset is enabled when entering Standby mode (by clearing the nSTDBY_RST bit in the user system data area), or it is enabled when entering Deepsleep mode (by clearing the nDEPSLP_RST bit in the user system data area)
- POR reset: power-on reset
- LVR reset: low voltage reset
- When exiting Standby mode

NRST reset, WDT reset, WWDT reset, software reset and low-power management reset sets all registers to their reset values except the clock control/status register (CRM_CTRLSTS) and battery powered domain registers. The power-on reset, low voltage reset or reset generated when exiting Standby mode sets all registers to their reset values except the battery powered domain registers.

Figure 4-3 System reset circuit



4.2.2 Battery powered domain reset

Battery powered domain has two specific reset sources:

- Software reset: triggered by setting the BPDRST bit in the battery powered domain control register (CRM_BPDC)
- VDD or VBAT power on, if VDD or VBAT has been powered off

Software reset affects only the battery powered domain.

4.3 CRM registers

These peripheral registers can be accessed by bytes (8 bits), half-words (16 bits) and words (32 bits).

Table 4-1 CRM register map and reset values

Register	Offset	Reset value
CRM_CTRL	0x000	0x0000 XX83
CRM_PLLCFG	0x004	0x0000 07C1
CRM_CFG	0x008	0x4000 0000
CRM_CLKINT	0x00C	0x0000 0000
CRM_AHBRST1	0x010	0x0000 0000
CRM_AHBRST2	0x014	0x0000 0000
CRM_AHBRST3	0x018	0x0000 0000
CRM_APB1RST	0x020	0x0000 0000
CRM_APB2RST	0x024	0x0000 0000
CRM_AHBEN1	0x030	0x0000 0000
CRM_AHBEN2	0x034	0x0000 0000
CRM_AHBEN3	0x038	0x0000 0000
CRM_APB1EN	0x040	0x0000 0000
CRM_APB2EN	0x044	0x0000 0000
CRM_AHBLPEN1	0x050	0x1F41 90FF
CRM_AHBLPEN2	0x054	0x0000 80D0
CRM_AHBLPEN3	0x058	0x0000 0003
CRM_APB1LPEN	0x060	0xFEFE C9FF
CRM_APB2LPEN	0x064	0xE017 7333
CRM_PICLKS	0x068	0x0000 0000
CRM_BPDC	0x070	0x0000 0018

	CRM_CTRLSTS	0x074	0x0C00 0000
	CRM_MISC1	0x0A0	0x000F 0000
	CRM_MISC2	0x0A4	0x4380 500D

4.3.1 Clock control register (CRM_CTRL)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:27	Reserved	0x00	resd	Kept at its default value.
Bit 26	PLLUSTBL	0x0	ro	PLLU clock stable This bit is set by hardware after PLLU is ready. 0: PLLU clock is not ready 1: PLLU clock is ready
Bit 25	PLLSTBL	0x0	ro	PLL clock stable This bit is set by hardware after PLL is ready. 0: PLL is not ready 1: PLL is ready
Bit 24	PLLEN	0x0	rw	PLL enable This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. When the PLL clock is used as system clock, this bit cannot be cleared. 0: PLL is disabled 1: PLL is enabeld
Bit 23:20	Reserved	0x0	resd	Kept at its default value.
Bit 19	CFDEN	0x0	rw	Clock fail detector enable 0: Disabled 1: Enabled
Bit 18	HEXTBYPS	0x0	rw	High-speed external crystal bypass This bit can be set only if the HEXT is disabled. 0: Disabled 1: Enabled
Bit 17	HEXTSTBL	0x0	ro	High-speed external crystal stable This bit is set by hardware after HEXT becomes stable. 0: HEXT is not ready 1: HEXT is ready
Bit 16	HEXTEN	0x0	rw	High-speed external crystal enable This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. When the HEXT clock is used as system clock, this bit cannot be cleared 0: Disabled 1: Enabled
Bit 15:8	HICKCAL	0xXX	rw	High-speed internal clock calibration The default value of this field is the initial factory calibration value. When the HICK output frequency is 48 MHz, it needs adjust 240 KHz (design value) based on this frequency for each HICKCAL value change. When HICK output frequency is 8 MHz, it needs adjust 40 KHz (design value) based on this frequency for each HICKCAL value change. Note: This bit can be written only if the HICKCAL_KEY[7:0] is set as 0x5A.
Bit 7:2	HICKTRIM	0x20	rw	High-speed internal clock trimming These bits work with the HICKCAL[7:0] to determine the HICK oscillator frequency. The default value is 32, which can trim the HICK accuracy to be ±1%.
Bit 1	HICKSTBL	0x1	ro	High-speed internal clock stable

				This bit is set by hardware after HICK is ready. 0: HICK is not ready 1: HICK is ready
Bit 0	HICKEN	0x1	rw	High-speed internal clock enable This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. When a HEXT clock failure occurs, this bit can also be set. When the HICK clock is used as system clock, this bit cannot be cleared. 0: Disabled 1: Enabled

4.3.2 PLL clock configuration register (CRM_PLLCFG)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its reset value.
Bit 30	PLLRCS	0x0	rw	PLL reference clock select The PLL reference clock source is selected by writing “1” or “0” too this bit by software. This bit can be written only when the PLL is disabled. 0: HICK is used as PLL reference clock 1: HEXT is used as PLL reference clock
Bit 29	PLLU_EN	0x0	rw	PLLU enable This bit is set and cleared by software. It can also be cleared by hardware when entering Standby or Deepsleep mode. 0: Disabled 1: Enabled
Bit 28:23	Reserved	0x0	resd	Kept at its reset value.
Bit 22:20	PLL_FU	0x0	rw	PLLU post-division PLL_FU range (0~7) This bit cannot be set when PLL is enabled. 000: PLLU post-division = 11 001: PLLU post-division = 13 010: PLLU post-division = 12 011: PLLU post-division = 14 100: PLLU post-division = 16 101: PLLU post-division = 18 110: PLLU post-division = 20 111: PLLU post-division = 11 Pay attention to the correlation between PLL_FR value and the post-division factor.
Bit 19:16	PLL_FP	0x0	rw	PLLP post-division PLL_FP range (0~15) This bit cannot be set when PLL is enabled. 0000: PLLP post-division = 1 0001: PLLP post-division = 2 0010: PLLP post-division = 4 0011: PLLP post-division = 6 0100: PLLP post-division = 8

				0101: PLLP post-division = 10 0110: PLLP post-division = 12 0111: PLLP post-division = 14 1000: PLLP post-division = 16 1001: PLLP post-division = 18 1010: PLLP post-division = 20 1011: PLLP post-division = 22 1100: PLLP post-division = 24 1101: PLLP post-division = 26 1110: PLLP post-division = 28 1111: PLLP post-division = 30 Pay attention to the correlation between PLL_FR value and the post-division factor.
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14:6	PLL_NS	0x01F	rw	PLL multiplication factor PLL_NS range (31~500) This bit cannot be set when PLL is enabled. 000000000 ~ 000011110: Forbidden 000011111: 31 000100000: 32 000100001: 33 …… 111110011: 499 111110100: 500 111110101~111111111: Forbidden
Bit 5:4	Reserved	0x0	resd	Kept at its reset value.
Bit 3:0	PLL_MS	0x1	rw	PLL pre-division PLL_MS range (1~15) This bit cannot be set when PLL is enabled. 0000: Forbidden 0001: 1 0010: 2 0011: 3 …… 1110: 14 1111: 15

Note: PLL clock formula:

PLLP output clock = PLL input clock x PLL frequency multiplication factor / (PLL pre-divider factor x PLL_FP post-divider factor)

PLLU output clock = PLL input clock x PLL frequency multiplication factor / (PLL pre-divider factor x PLL_FU post-divider factor)

500MHz <= PLL input clock x PLL frequency multiplication factor / PLL pre-divider factor <= 1000MHz

2MHz <= PLL input clock / PLL pre-divider factor <= 16MHz

4.3.3 Clock configuration register (CRM_CFG)

Access: 0 to 2 wait states, accessible by words, half-words and bytes; 1 or 2 wait states are inserted only when the access occurs during a clock source switch

Bit	Name	Reset value	Type	Description
Bit 31:30	CLKOUT_SEL1	0x1	rw	Clock output selecction 1 This bit is set or cleared by software. 00: System clock (SCLK) output 01: Secondary clock output, selected by the CLKOUT_SEL2 bit in the CRM_MISC1 register 10: External oscillator clock (HEXT) output 11: PLL clock output Note: This clock may be cut off during the startup and switch of CLKOUT. While being used as an output to CLKOUT pin, the system clock output must be no more than 50 MHz (the maximum frequency of an I/O port).
Bit 29:27	CLKOUTDIV1	0x0	rw	Clock output division1 0xx: CLKOUT 100: CLKOUT/2 101: CLKOUT/3 110: CLKOUT/4 111: CLKOUT/5
Bit 26:24	Reserved	0x0	resd	Kept at its reset value.
Bit 23:22	I2SF5CLKSEL	0x0	resd	I^2SF5 clock source selection 00: System clock 01: PLL 10: HICK 11: External input clock
Bit 21	Reserved	0x0	resd	Kept at its reset value.
Bit 20:16	ERTCDIV	0x00	rw	HEXT division for ERTC clock This field is set and cleared by software to divide the HEXT for ERTC clock. This field must be configured before selecting the ERTC clock source. 00000: Forbidden 00001: Forbidden 00010: HEXT/2 00011: HEXT/3 00100: HEXT/4 … 11110: HEXT/30 11111: HEXT/31
Bit 15:13	APB2DIV	0x0	rw	APB2 division The divided HCLK is used as APB2 clock. 0xx: Not divided 100: HCLK/2 101: HCLK/4

				110: HCLK/8 111: HCLK/16
Bit 12:10	APB1DIV	0x0	rw	APB1 division The divided HCLK is used as APB1 clock. 0xx: Not divided 100: HCLK/2 101: HCLK/4 110: HCLK/8 111: HCLK/16
Bit 9:8	Reserved	0x0	resd	Kept at its reset value.
Bit 7:4	AHBDIV	0x0	rw	AHB division 0xxx: SCLK not divided 1000: SCLK/2 1100: SCLK/64 1001: SCLK/4 1101: SCLK/128 1010: SCLK/8 1110: SCLK/256 1011: SCLK/16 1111: SCLK/512
Bit 3:2	SCLKSTS	0x0	ro	System clock select status 00: HICK 01: HEXT 10: PLL/2 11: Reserved. Kept at its default value.
Bit 1:0	SCLKSEL	0x0	rw	System clock select 00: HICK 01: HEXT 10: PLL/2 11: Reserved. Kept at its default value.

4.3.4 Clock interrupt register (CRM_CLKINT)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	resd	Kept at its reset value.
Bit 23	CFDFC	0x0	wo	Clock failure detection interrupt clear Writing “1” by software to clear CFDF. 0: No effect 1: Clear
Bit 22:21	Reserved	0x0	resd	Kept at its reset value.
Bit 20	PLLSTBLFC	0x0	wo	PLL stable flag clear Writing “1” by software to clear PLLSTBLF. 0: No effect 1: Clear
Bit 19	HEXTSTBLFC	0x0	wo	HEXT stable flag clear Writing “1” by software to clear HEXTSTBLF. 0: No effect 1: Clear
Bit 18	HICKSTBLFC	0x0	wo	HICK stable flag clear Writing “1” by software to clear HICKSTBLF.

				0: No effect 1: Clear
Bit 17	LEXTSTBLFC	0x0	wo	LEXT stable flag clear Writing “1” by software to clear LEXTSTBLF. 0: No effect 1: Clear
Bit 16	LICKSTBLFC	0x0	wo	LICK stable flag clear Writing “1” by software to clear LICKSTBLF. 0: No effect 1: Clear
Bit 15:13	Reserved	0x0	resd	Kept at its reset value.
Bit 12	PLLSTBLIEN	0x0	rw	PLL stable interrupt enable 0: Disabled 1: Enabled
Bit 11	HEXTSTBLIEN	0x0	rw	HEXT stable interrupt enable 0: Disabled 1: Enabled
Bit 10	HICKSTBLIEN	0x0	rw	HICK stable interrupt enable 0: Disabled 1: Enabled
Bit 9	LEXTSTBLIEN	0x0	rw	LEXT stable interrupt enable 0: Disabled 1: Enabled
Bit 8	LICKSTBLIEN	0x0	rw	LICK stable interrupt enable 0: Disabled 1: Enabled
Bit 7	CFDF	0x0	ro	Clock failure detection flag This bit is set by hardware when the HEXT clock failure occurs. 0: No clock failure 1: Clock failure
Bit 6:5	Reserved	0x0	resd	Kept at its reset value.
Bit 4	PLLSTBLF	0x0	ro	PLL stable flag This bit is set by hardware. 0: PLL is not ready 1: PLL is ready
Bit 3	HEXTSTBLF	0x0	ro	HEXT stable flag This bit is set by hardware. 0: HEXT is not ready 1: HEXT is ready
Bit 2	HICKSTBLF	0x0	ro	HICK stable flag This bit is set by hardware. 0: HICK is not ready 1: HICK is ready
Bit 1	LEXTSTBLF	0x0	ro	LEXT stable flag This bit is set by hardware.

				0: LEXT is not ready 1: LEXT is ready
Bit 0	LICKSTBLF	0x0	ro	LICK stable flag This bit is set by hardware. 0: LICK is not ready 1: LICK is ready

4.3.5 AHB peripheral reset register 1 (CRM_AHBRST1)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:26	Reserved	0x0	resd	Kept at its reset value.
Bit 25	EMACRST	0x0	rw	EMCA reset 0: Does not reset EMCA 1: Reset EMCA
Bit 24	DMA2RST	0x0	rw	DMA2 reset 0: Does not reset DMA2 1: Reset DMA2
Bit 23	Reserved	0x0	resd	Kept at its reset value.
Bit 22	DMA1RST	0x0	rw	DMA1 reset 0: Does not reset DMA1 1: Reset DMA1
Bit 21:13	Reserved	0x000	resd	Kept at its reset value.
Bit 12	CRCRST	0x0	rw	CRC reset 0: Does not reset CRC 1: Reset CRC
Bit 11:8	Reserved	0x00	resd	Kept at its reset value.
Bit 7	GPIOHRST	0x0	rw	IO port H reset 0: Does not reset IO port H 1: Reset IO port H
Bit 6	GPIOGRST	0x0	rw	IO port G reset 0: Does not reset IO port G 1: Reset IO port G
Bit 5	GPIOFRST	0x0	rw	IO port F reset 0: Does not reset IO port F 1: Reset IO port F
Bit 4	GPIOERST	0x0	rw	IO port E reset 0: Does not reset IO port E 1: Reset IO port E
Bit 3	GPIODRST	0x0	rw	IO port D reset 0: Does not reset IO port D 1: Reset IO port D
Bit 2	GPIOCRST	0x0	rw	IO port C reset 0: Does not reset IO port C 1: Reset IO port C
Bit 1	GPIOBRST	0x0	rw	IO port B reset 0: Does not reset IO port B

				1: Reset IO port B
Bit 0	GPIOARST	0x0	rw	IO port A reset 0: Does not reset IO port A 1: Reset IO port A

4.3.6 AHB peripheral reset register 2 (CRM_AHBRST2)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x000000	resd	Kept at its reset value.
Bit 15	SDIO1RST	0x0	rw	SDIO reset 0: Does not reset SDIO 1: Reset SDIO
Bit 14:8	Reserved	0x00	resd	Kept at its reset value.
Bit 7	OTGFS1RST	0x0	rw	OTGFS1 reset 0: Does not reset 1: Reset OTGFS1
Bit 6	TRNGRST	0x0	rw	TRNG reset 0: Does not reset 1: Reset TRNG
Bit 5	Reserved	0x00	resd	Kept at its reset value.
Bit 4	AESRST	0x0	rw	AES reset 0: Does not reset 1: Reset AES
Bit 3:0	Reserved	0x00	resd	Kept at its reset value.

4.3.7 AHB peripheral reset register 3 (CRM_AHBRST3)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:2	Reserved	0x00000000	resd	Kept at its reset value.
Bit 1	QSPI1RST	0x0	rw	QSPI reset 0: Does not reset QSPI 1: Reset QSPI
Bit 0	XMCRST	0x0	rw	XMC reset 0: Does not reset XMC 1: Reset XMC

4.3.8 APB1 peripheral reset register (CRM_APB1RST)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	USART8RST	0x0	rw	USART8 reset 0: Does not reset USART8 1: Reset USART8
Bit 30	USART7RST	0x0	rw	USART7 reset 0: Does not reset USART7 1: Reset USART7
Bit 29	DACRST	0x0	rw	DAC reset 0: Does not reset DAC 1: Reset DAC
Bit 28	PWCRST	0x0	rw	Power interface reset 0: Does not reset power interface 1: Reset power interface
Bit 27	CAN3RST	0x0	rw	CAN3 reset 0: Does not reset CAN3 1: Reset CAN3
Bit 26	CAN2RST	0x0	rw	CAN2 reset 0: Does not reset CAN2 1: Reset CAN2
Bit 25	CAN1RST	0x0	rw	CAN1 reset 0: Does not reset CAN1 1: Reset CAN1
Bit 24	Reserved	0x0	resd	Kept at its reset value.
Bit 23	I2C3RST	0x0	rw	I^2C3 reset 0: Does not reset I^2C3 1: Reset I^2C3
Bit 22	I2C2RST	0x0	rw	I^2C2 reset 0: Does not reset I^2C2 1: Reset I^2C2
Bit 21	I2C1RST	0x0	rw	I^2C1 reset 0: Does not reset I^2C1 1: Reset I^2C1
Bit 20	USART5RST	0x0	rw	USART5 reset 0: Does not reset USART5 1: Reset USART5
Bit 19	USART4RST	0x0	rw	USART4 reset 0: Does not reset USART4 1: Reset USART4
Bit 18	USART3RST	0x0	rw	USART3 reset This bit is set or cleared by software. 0: No effect 1: Reset USART3
Bit 17	USART2RST	0x0	rw	USART2 reset 0: Does not reset USART2

				1: Reset USART2
Bit 16	Reserved	0x0	resd	Kept at its reset value.
Bit 15	SPI3RST	0x0	rw	SPI3 reset 0: Does not reset SPI3 1: Reset SPI3
Bit 14	SPI2RST	0x0	rw	SPI2 reset 0: Does not reset SPI2 1: Reset SPI2
Bit 13:12	Reserved	0x0	resd	Kept at its reset value.
Bit 11	WWDTRST	0x0	rw	Window watchdog reset 0: Does not reset WWDT 1: Reset WWDT
Bit 10:9	Reserved	0x0	resd	Kept at its reset value.
Bit 8	TMR14RST	0x0	rw	Timer14 reset 0: Does not reset Timer14 1: Reset Timer14
Bit 7	TMR13RST	0x0	rw	Timer13 reset 0: Does not reset Timer13 1: Reset Timer13
Bit 6	TMR12RST	0x0	rw	Timer12 reset 0: Does not reset Timer12 1: Reset Timer12
Bit 5	TMR7RST	0x0	rw	Timer7 reset 0: Does not reset Timer7 1: Reset Timer7
Bit 4	TMR6RST	0x0	rw	Timer6 reset 0: Does not reset Timer6 1: Reset Timer6
Bit 3	TMR5RST	0x0	rw	Timer5 reset 0: Does not reset Timer5 1: Reset Timer5
Bit 2	TMR4RST	0x0	rw	Timer4 reset 0: Does not reset Timer4 1: Reset Timer4
Bit 1	TMR3RST	0x0	rw	Timer3 reset 0: Does not reset Timer3 1: Reset Timer3
Bit 0	TMR2RST	0x0	rw	Timer2 reset 0: Does not reset Timer2 1: Reset Timer2

4.3.9 APB2 peripheral reset register (CRM_APB2RST)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	I2S3EXTRST	0x0	rw	I2S3EXT APB reset 0: Does not reset I2S3EXT APB 1: Reset I2S3EXT APB
Bit 30	I2S2EXTRST	0x0	rw	I2S2EXT APB reset 0: Does not reset I2S2EXT APB 1: Reset I2S2EXT APB
Bit 29	ACCRST	0x0	rw	ACC reset 0: Does not reset ACC 1: Reset ACC
Bit 28:21	Reserved	0x00	resd	Kept at its reset value.
Bit 20	I2SF5RST	0x0	rw	I^2SF5 reset 0: Does not reset I^2SF5 1: Reset I^2SF5
Bit 19	Reserved	0x0	resd	Kept at its reset value.
Bit 18	TMR11RST	0x0	rw	Timer11 reset 0: Does not reset Timer11 1: Reset Timer11
Bit 17	TMR10RST	0x0	rw	Timer10 reset 0: Does not reset Timer10 1: Reset Timer10
Bit 16	TMR9RST	0x0	rw	Timer9 reset 0: Does not reset Timer9 1: Reset Timer9
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14	SCFGRST	0x0	rw	SCFG reset 0: Does not reset SCFG 1: Reset SCFG
Bit 13	SPI4RST	0x0	rw	SPI4 reset 0: Does not reset SPI4 1: Reset SPI4
Bit 12	SPI1RST	0x0	rw	SPI1 reset 0: Does not reset SPI1 1: Reset SPI1
Bit 11:9	Reserved	0x0	resd	Kept at its reset value.
Bit 8	ADCRST	0x0	rw	ADC interface reset 0: Does not reset ADC interface 1: Reset ADC interface
Bit 7:6	Reserved	0x0	resd	Kept at its reset value.
Bit 5	USART6RST	0x0	rw	USART6 reset 0: Does not reset USART6 1: Reset USART6
Bit 4	USART1RST	0x0	rw	USART1 reset 0: Does not reset USART1

				1: Reset USART1
Bit 3:2	Reserved	0x0	resd	Kept at its reset value.
Bit 1	TMR8RST	0x0	rw	TMR8 reset 0: Does not reset TMR8 1: Reset TMR8
Bit 0	TMR1RST	0x0	rw	TMR1 reset 0: Does not reset TMR1 1: Reset TMR1

4.3.10 AHB peripheral clock enable register 1 (CRM_AHBEN1)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:29	Reserved	0x0	resd	Kept at its reset value.
Bit 28	EMACPTPEN	0x0	rw	EMAC PTP clock enable 0: Disabled 1: Enabled
Bit 27	EMACRXEN	0x0	rw	EMAC RX clock enable 0: Disabled 1: Enabled Note: In RMII mode, if this clock is enabled, then the MAC RMII clock is also enabled.
Bit 26	EMACTXEN	0x0	rw	EMAC TX clock enable 0: Disabled 1: Enabled Note: In RMII mode, if this clock is enabled, then the MAC RMII clock is also enabled.
Bit 25	EMACEN	0x0	rw	EMAC TX clock enable 0: Disabled 1: Enabled
Bit 24	DMA2EN	0x0	rw	DMA2 clock enable 0: Disabled 1: Enabled
Bit 23	Reserved	0x0	resd	Kept at its reset value.
Bit 22	DMA1EN	0x0	rw	DMA1 clock enable 0: Disabled 1: Enabled
Bit 21:13	Reserved	0x000	resd	Kept at its reset value.
Bit 12	CRCEN	0x0	rw	CRC clock enable) 0: Disabled 1: Enabled
Bit 11:8	Reserved	0x00	resd	Kept at its reset value.
Bit 7	GPIOHEN	0x0	rw	IO port H clock enable 0: Disabled 1: Enabled
Bit 6	GPIOGEN	0x0	rw	IO port G clock enable 0: Disabled

				1: Enabled
Bit 5	GPIOFEN	0x0	rw	IO port F clock enable 0: Disabled 1: Enabled
Bit 4	GPIOEEN	0x0	rw	IO port E clock enable 0: Disabled 1: Enabled
Bit 3	GPIODEN	0x0	rw	IO port D clock enable 0: Disabled 1: Enabled
Bit 2	GPIOCEN	0x0	rw	IO port C clock enable 0: Disabled 1: Enabled
Bit 1	GPIOBEN	0x0	rw	IO port B clock enable 0: Disabled 1: Enabled
Bit 0	GPIOAEN	0x0	rw	IO port A clock enable 0: Disabled 1: Enabled

4.3.11 AHB peripheral clock enable register 2 (CRM_AHBEN2)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x000000	resd	Kept at its reset value.
Bit 15	SDIO1EN	0x0	rw	SDIO clock enable 0: Disabled 1: Enabled
Bit 14:8	Reserved	0x000000	resd	Kept at its reset value.
Bit 7	OTGFS1EN	0x0	rw	OTGFS1 clock enable 0: Disabled 1: Enabled
Bit 6	TRNGEN	0x0	rw	TRNG clock enable 0: Disabled 1: Enabled
Bit 5	Reserved	0x0	resd	Kept at its reset value.
Bit 4	AESEN	0x0	rw	AES clock enable 0: Disabled 1: Enabled
Bit 3:0	Reserved	0x00	resd	Kept at its reset value.

4.3.12 AHB peripheral clock enable register 3 (CRM_AHBEN3)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:2	Reserved	0x00000000	resd	Kept at its reset value.
Bit 1	QSPI1EN	0x0	rw	QSPI clock enable 0: Disabled 1: Enabled
Bit 0	XMCEN	0x0	rw	XMC clock enable 0: Disabled 1: Enabled

4.3.13 APB1 peripheral clock enable register (CRM_APB1EN)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	USART8EN	0x0	rw	USART8 clock enable 0: Disabled 1: Enabled
Bit 30	USART7EN	0x0	rw	USART7 clock enable 0: Disabled 1: Enabled
Bit 29	DACEN	0x0	rw	DAC clock enable 0: Disabled 1: Enabled
Bit 28	PWCEN	0x0	rw	Power interface clock enable 0: Disabled

				1: Enabled
Bit 27	CAN3EN	0x0	rw	CAN3 clock enable 0: Disabled 1: Enabled
Bit 26	CAN2EN	0x0	rw	CAN2 clock enable 0: Disabled 1: Enabled
Bit 25	CAN1EN	0x0	rw	CAN1 clock enable 0: Disabled 1: Enabled
Bit 24	Reserved	0x0	resd	Kept at its reset value.
Bit 23	I2C3EN	0x0	rw	I^2C3 clock enable 0: Disabled 1: Enabled
Bit 22	I2C2EN	0x0	rw	I^2C2 clock enable 0: Disabled 1: Enabled
Bit 21	I2C1EN	0x0	rw	I^2C1 clock enable 0: Disabled 1: Enabled
Bit 20	USART5EN	0x0	rw	USART5 clock enable 0: Disabled 1: Enabled
Bit 19	USART4EN	0x0	rw	USART4 clock enable 0: Disabled 1: Enabled
Bit 18	USART3EN	0x0	rw	USART3 clock enable 0: Disabled 1: Enabled
Bit 17	USART2EN	0x0	rw	USART2 clock enable 0: Disabled 1: Enabled
Bit 16	Reserved	0x0	resd	Kept at its reset value.
Bit 15	SPI3EN	0x0	rw	SPI3 clock enable 0: Disabled 1: Enabled
Bit 14	SPI2EN	0x0	rw	SPI2 clock enable 0: Disabled 1: Enabled
Bit 13:12	Reserved	0x0	resd	Kept at its reset value.
Bit 11	WWDTEN	0x0	rw	Window watchdog clock enable 0: Disabled 1: Enabled
Bit 10:9	Reserved	0x0	resd	Kept at its reset value.
Bit 8	TMR14EN	0x0	rw	Timer14 clock enable 0: Disabled

				1: Enabled
Bit 7	TMR13EN	0x0	rw	Timer13 clock enable 0: Disabled 1: Enabled
Bit 6	TMR12EN	0x0	rw	Timer12 clock enable 0: Disabled 1: Enabled
Bit 5	TMR7EN	0x0	rw	Timer 7 clock enable 0: Disabled 1: Enabled
Bit 4	TMR6EN	0x0	rw	Timer 6 clock enable 0: Disabled 1: Enabled
Bit 3	TMR5EN	0x0	rw	Timer 5 clock enable 0: Disabled 1: Enabled
Bit 2	TMR4EN	0x0	rw	Timer 4 clock enable 0: Disabled 1: Enabled
Bit 1	TMR3EN	0x0	rw	Timer 3 clock enable 0: Disabled 1: Enabled
Bit 0	TMR2EN	0x0	rw	Timer 2 clock enable 0: Disabled 1: Enabled

4.3.14 APB2 peripheral clock enable register (CRM_APB2EN)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	I2S3EXTEN	0x0	rw	I2S3EXT APB clock enable 0: Disabled 1: Enabled
Bit 30	I2S2EXTEN	0x0	rw	I2S2EXT APB clock enable 0: Disabled 1: Enabled
Bit 29	ACCEN	0x0	rw	ACC clock enable 0: Disabled 1: Enabled
Bit 28:21	Reserved	0x00	resd	Kept at its reset value.
Bit 20	I2SF5EN	0x0	rw	I^2SF5 clock enable 0: Disabled 1: Enabled
Bit 19	Reserved	0x0	resd	Kept at its reset value.
Bit 18	TMR11EN	0x0	rw	Timer11 clock enable 0: Disabled 1: Enabled

Bit 17	TMR10EN	0x0	rw	Timer10 clock enable 0: Disabled 1: Enabled
Bit 16	TMR9EN	0x0	rw	Timer9 clock enable 0: Disabled 1: Enabled
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14	SCFGEN	0x0	rw	SCFG clock enable 0: Disabled 1: Enabled
Bit 13	SPI4EN	0x0	rw	SPI4 clock enable 0: Disabled 1: Enabled
Bit 12	SPI1EN	0x0	rw	SPI1 clock enable 0: Disabled 1: Enabled
Bit 11:10	Reserved	0x0	resd	Kept at its reset value.
Bit 9	ADC2EN	0x0	rw	ADC2 interface clock enable 0: Disabled 1: Enabled
Bit 8	ADC1EN	0x0	rw	ADC1 interface clock enable 0: Disabled 1: Enabled
Bit 7:6	Reserved	0x0	resd	Kept at its reset value.
Bit 5	USART6EN	0x0	rw	USART6 clock enable 0: Disabled 1: Enabled
Bit 4	USART1EN	0x0	rw	USART1 clock enable 0: Disabled 1: Enabled
Bit 3:2	Reserved	0x0	resd	Kept at its reset value.
Bit 1	TMR8EN	0x0	rw	TMR8 clock enable 0: Disabled 1: Enabled
Bit 0	TMR1EN	0x0	rw	TMR1 clock enable 0: Disabled 1: Enabled

4.3.15 AHB peripheral clock enable in low-power mode register 1 (CRM_AHBLPEN1)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:29	Reserved	0x0	resd	Kept at its reset value.
Bit 28	EMACPTPLPEN	0x1	rw	EMAC PTP clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 27	EMACRXLPEN	0x1	rw	EMAC RX clock enable in Sleep mode 0: Disabled 1: Enabled Note: In RMII mode, if this clock is enabled, then the MAC RMII clock is also enabled.
Bit 26	EMACTXLPEN	0x1	rw	EMAC TX clock enable in Sleep mode 0: Disabled 1: Enabled Note: In RMII mode, if this clock is enabled, then the MAC RMII clock is also enabled.
Bit 25	EMACLPEN	0x1	rw	EMAC clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 24	DMA2LPEN	0x1	rw	DMA2 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 23	Reserved	0x0	resd	Kept at its reset value.
Bit 22	DMA1LPEN	0x1	rw	DMA1 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 21:17	Reserved	0x00	resd	Kept at its reset value.
Bit 16	SRAMLPEN	0x1	rw	SRAM clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 15	FLASHLPEN	0x1	rw	FLASH clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 14:13	Reserved	0x0	resd	Kept at its reset value.
Bit 12	CRCLPEN	0x1	rw	CRC clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 11:8	Reserved	0x0	resd	Kept at its reset value.
Bit 7	GPIOHLPEN	0x1	rw	IO port H clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 6	GPIOGLPEN	0x1	rw	IO port G clock enable in Sleep mode 0: Disabled 1: Enabled

Bit 5	GPIOFLPEN	0x1	rw	IO port F clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 4	GPIOELPEN	0x1	rw	IO port E clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 3	GPIODLPEN	0x1	rw	IO port D clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 2	GPIOCLPEN	0x1	rw	IO port C clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 1	GPIOBLPEN	0x1	rw	IO port B clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 0	GPIOALPEN	0x1	rw	IO port A clock enable in Sleep mode 0: Disabled 1: Enabled

4.3.16 AHB peripheral clock enable in low-power mode register 2 (CRM_AHBLPEN2)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x000000	resd	Kept at its reset value.
Bit 15	SDIO1LPEN	0x1	rw	SDIO clock enable in Sleep mode) 0: Disabled 1: Enabled
Bit 14:8	Reserved	0x00	resd	Kept at its reset value.
Bit 7	OTGFS1LPEN	0x1	rw	OTGFS1 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 6	TRNGLPEN	0x1	rw	TRNG clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 5	Reserved	0x0	resd	Kept at its reset value.
Bit 4	AESLPEN	0x1	rw	AES clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 3:0	Reserved	0x00	resd	Kept at its reset value.

4.3.17 AHB peripheral clock enable in low-power mode register 3 (CRM_AHBLPEN3)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:2	Reserved	0x00000000	resd	Kept at its reset value.
Bit 1	QSPI1LPEN	0x1	rw	QSPI clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 0	XMCLPEN	0x1	rw	XMC clock enable in Sleep mode 0: Disabled 1: Enabled

4.3.18 APB1 peripheral clock enable in low-power mode register (CRM_APB1LPEN)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	USART8LPEN	0x1	rw	USART8 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 30	USART7LPEN	0x1	rw	USART7 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 29	DACLPEN	0x1	rw	DAC clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 28	PWCLPEN	0x1	rw	Power interface clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 27	CAN3LPEN	0x1	rw	CAN3 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 26	CAN2LPEN	0x1	rw	CAN2 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 25	CAN1LPEN	0x1	rw	CAN1 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 24	Reserved	0x0	resd	Kept at its reset value.
Bit 23	I2C3LPEN	0x1	rw	I²C3 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 22	I2C2LPEN	0x1	rw	I²C2 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 21	I2C1LPEN	0x1	rw	I²C1 clock enable in Sleep mode 0: Disabled

				1: Enabled
Bit 20	USART5LPEN	0x1	rw	USART5 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 19	USART4LPEN	0x1	rw	USART4 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 18	USART3LPEN	0x1	rw	USART3 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 17	USART2LPEN	0x1	rw	USART2 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 16	Reserved	0x0	resd	Kept at its reset value.
Bit 15	SPI3LPEN	0x1	rw	SPI3 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 14	SPI2LPEN	0x1	rw	SPI2 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 13:12	Reserved	0x0	resd	Kept at its reset value.
Bit 11	WWDTLPEN	0x1	rw	WWDT clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 10:9	Reserved	0x0	resd	Kept at its reset value.
Bit 8	TMR14LPEN	0x1	rw	TMR14 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 7	TMR13LPEN	0x1	rw	TMR13 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 6	TMR12LPEN	0x1	rw	TMR12 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 5	TMR7LPEN	0x1	rw	TMR7 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 4	TMR6LPEN	0x1	rw	TMR6 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 3	TMR5LPEN	0x1	rw	TMR5 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 2	TMR4LPEN	0x1	rw	TMR4 clock enable in Sleep mode 0: Disabled 1: Enabled

Bit 1	TMR3LPEN	0x1	rw	TMR3 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 0	TMR2LPEN	0x1	rw	TMR2 clock enable in Sleep mode 0: Disabled 1: Enabled

4.3.19 APB2 peripheral clock enable in low-power mode register (CRM_APB2LPEN)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31	I2S3EXTLPEN1	0x1	rw	I2S3EXT APB clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 30	I2S2EXTLPEN1	0x1	rw	I2S2EXT APB clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 29	ACCLPEN	0x1	rw	ACC clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 28:21	Reserved	0x00	resd	Kept at its reset value.
Bit 20	I2SF5LPEN	0x1	rw	I^2SF5 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 19	Reserved	0x0	resd	Kept at its reset value.
Bit 18	TMR11LPEN	0x1	rw	TMR11 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 17	TMR10LPEN	0x1	rw	TMR10 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 16	TMR9LPEN	0x1	rw	TMR9 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14	SCFGLPEN	0x1	rw	SCFG clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 13	SPI4LPEN	0x1	rw	SPI4 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 12	SPI1LPEN	0x1	rw	SPI1 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 11:10	Reserved	0x0	resd	Kept at its reset value.
Bit 9	ADC2LPEN	0x1	rw	ADC2 interface clock enable in Sleep mode

				0: Disabled 1: Enabled
Bit 8	ADC1LPEN	0x1	rw	ADC1 interface clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 7:6	Reserved	0x0	resd	Kept at its reset value.
Bit 5	USART6LPEN	0x1	rw	USART6 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 4	USART1LPEN	0x1	rw	USART1 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 3:2	Reserved	0x0	resd	Kept at its reset value.
Bit 1	TMR8LPEN	0x1	rw	TMR8 clock enable in Sleep mode 0: Disabled 1: Enabled
Bit 0	TMR1LPEN	0x1	rw	TMR1 clock enable in Sleep mode 0: Disabled 1: Enabled

4.3.20 Peripheral independent clock select register (CRM_PICLKS)

Access: 0 wait state, accessible by words, half-words and bytes

Bit	Name	Reset value	Type	Description
Bit 31:30	Reserved	0x0	resd	Kept at its reset value.
Bit 29:28	CAN3_CLKSEL	0x0	rw	CAN3 host clock select 00: HEXT 01: PLLU 10: PCLK1 11: Reserved
Bit 27:26	CAN2_CLKSEL	0x0	rw	CAN2 host clock select 00: HEXT 01: PLLU 10: PCLK1 11: Reserved
Bit 25:24	CAN1_CLKSEL	0x0	rw	CAN1 host clock select 00: HEXT 01: PLLU 10: PCLK1 11: Reserved
Bit 23:0	Reserved	0x00 0000	resd	Kept at its reset value.

4.3.21 Batteyr powered domain control register (CRM_BPDC)

This register is reset only by the battery powered domain reset.

Access: 0 to 3 wait states, accessible by words, half-words and bytes. Wait states are inserted in the case of consecutive accesses to this register.

Note: LEXTEN, LEXTBYPS, ERTCSEL and ERTCEN bits of the CRM_BPDC register are in the battery powered domain. As a result, these bits are write-protected after reset, and can only be modified by setting the BPWEN bit in the PWC_CTRL register. These bits could be reset only by battery powered domain reset. Any internal or external reset does not affect these bits.

Bit	Name	Reset value	Type	Description
Bit 31:17	Reserved	0x0000	resd	Kept at its reset value.
Bit 16	BPDRST	0x0	rw	Battery powered domain software reset 0: No reset 1: Reset
Bit 15	ERTCEN	0x0	rw	ERTC clock enable This bit is set or cleared by software. 0: Disabled 1: Enabled
Bit 14:10	Reserved	0x00	resd	Kept at its reset value.
Bit 9:8	ERTCSEL	0x0	rw	ERTC clock source selection Once the ERTC clock source is selected, it cannot be changed until the BPDRST bit is reset. 00: None 01: LEXT 10: LICK 11: Divided HEXT (with the ERTC_DIV bit in the CRM_CFG register)
Bit 7:5	Reserved	0x00	resd	Kept at its reset value.
Bit 4:3	LEXTDRV	0x3	rw	LEXT driving strength 00: LOW 01: MEDIUM LOW 10: MEDIUM HIGH 11: HIGH
Bit 2	LEXTBYPS	0x0	rw	LEXT bypass enable 0: Disabled 1: Enabled
Bit 1	LEXTSTBL	0x0	ro	LEXT stable This bit is set by hardware after the LEXT is ready. 0: LEXT is not ready 1: LEXT is ready
Bit 0	LEXTEN	0x0	rw	LEXT enable 0: Disabled 1: Enabled

4.3.22 Control/status register (CRM_CTRLSTS)

Reset flag can be cleared by power reset or by writing the RSTFC bit, while others are cleared by system reset.

Access: 0 to 3 wait states, accessible by words, half-words and bytes. Wait states are inserted in the case of consecutive accesses to this register.

Bit	Name	Reset value	Type	Description
Bit 31	LPRSTF	0x0	ro	Low-power reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No low-power reset occurs 1: Low-power reset occurs
Bit 30	WWDTRSTF	0x0	ro	WWDT reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No WWDT reset occurs 1: WWDT reset occurs
Bit 29	WDTRSTF	0x0	ro	WDT reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No WDT reset occurs 1: WDT reset occurs
Bit 28	SWRSTF	0x0	ro	Software reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No software reset occurs 1: Software reset occurs
Bit 27	PORRSTF	0x1	ro	POR/LVR reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No POR/LVR reset occurs 1: POR/LVR reset occurs
Bit 26	NRSTF	0x1	ro	NRST reset flag This bit is set by hardware and cleared by writing to the RSTFX bit with software. 0: No NRST reset occurs 1: NRST reset occurs
Bit 25	Reserved	0x0	resd	Kept at its reset value.
Bit 24	RSTFC	0x0	rw	Reset flag clear This bit is cleared by writing “1” with software. 0: No effect 1: Clear reset flag
Bit 23:2	Reserved	0x000000	resd	Kept at its reset value.
Bit 1	LICKSTBL	0x0	ro	LICK stable 0: LICK is not ready 1: LICK is ready
Bit 0	LICKEN	0x0	rw	LICK enable 0: Disabled 1: Enabled

4.3.23 Additional register 1 (CRM_MISC1)

Access: 0 wait state, accessible by words, half-words and bytes。

Bit	Name	Reset value	Type	Description
Bit 31:28	CLKOUTDIV2	0x0	rw	Clock output division2 0xxx: Clock output not divided 1000: Clock output divided by 2 1001: Clock output divided by 4 1010: Clock output divided by 8 1011: Clock output divided by 16 1100: Clock output divided by 64 1101: Clock output divided by 128 1110: Clock output divided by 256 1111: Clock output divided by 512
Bit 27:20	Reserved	0x00	resd	Kept at its reset value.
Bit 19:16	CLKOUT_SEL2	0xF	rw	Clock output select2 0000: USBFS 48M clock output 0001: ADC clock output 0010: Internal RC oscillator clock (HICK) output frequency divider 0011: LICK clock output 0100: LEXT clock output 0101: Reserved 0110: Reserved 0111: Reserved 1xxx: Reserved
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14	HICK_TO_SCLK	0x0	rw	HICK as system clock frequency select When the HICK is selected as system clock by setting the SCLKSEL bit, the frequency of SCLK is: 0: Fixed 8 MHz, that is, HICK/6 1: 48 MHz
Bit 13	Reserved	0x0	resd	Kept at its reset value.
Bit 12	Reserved	0x1	resd	Kept at its reset value.
Bit 11:8	Reserved	0x0	resd	Kept at its reset value.
Bit 7:0	HICKCAL_KEY	0x00	rw	HICK calibration key The HICKCAL [7:0] can be written only when this field is set to 0x5A.

4.3.24 Additional register 2 (CRM_MISC2)

Access: 0 wait state, accessible by words, half-words and bytes。

Bit	Name	Reset value	Type	Description
Bit 31:30	Reserved	0x1	resd	Fixed to 0x1. Do not modify.
Bit 29:26	Reserved	0x0	resd	Kept at its reset value.
Bit 25:24	VBATHDIV	0x3	rw	VBATAHB division 0x: SCLK not divided 10: SCLK/2 11: SCLK/4
Bit 23:22	HEXTDRV	0x2	rw	HEXT driving strength 00: LOW 01: MEDIUM LOW 10: MEDIUM HIGH 11: HIGH
Bit 21:19	HEXT_TO_SCLK_DIV	0x00	rw	HEXT as system clock frequency division 000: HEXT 001: HEXT/2 010: HEXT/4 011: HEXT/8 100: HEXT/16 101: HEXT/32 Others: Reserved
Bit 18:16	HICK_TO_SCLK_DIV	0x00	rw	HICK as system clock frequency division 000: HICK 001: HICK/2 010: HICK/4 011: HICK/8 100: HICK/16 Others: Reserved
Bit 15	Reserved	0x0	resd	Kept at its reset value.
Bit 14:12	APB3DIV	0x5	rw	APB3 division The divided HCLK is used as APB3 clock. 0xx: Not divided 100: HCLK/2 101: HCLK/4 110: HCLK/8 111: HCLK/16
Bit 11	Reserved	0x0	resd	Kept at its reset value.
Bit 10	PLLU_USB48_SEL	0x0	rw	USBFS 48M clock source select 0: PLLU 1: HICK
Bit 9	EMAC_PPS_SEL	0x0	rw	EMAC PPS select Refer to the description of POFC bit in section 27.3.55.
Bit 8:6	Reserved	0x0	resd	Kept at its reset value.
Bit 5:4	AUTO_STEP_EN	0x0	rw	Auto step-by-step system clock switch enable

				When the system clock source is switched from others to the PLL or when the AHB prescaler is changed from large to small (system frequency is from small to large), it is recommended to enable the auto step-by-step system clock switch if the operational target is larger than 108 MHz. Once it is enabled, the AHB bus is halted by hardware till the completion of the switch. During this switch period, the DMA remain working, and the interrupt events are recorded and then handled by NVIC when the AHB bus resumes. 00: Disabled 01: Reserved 10: Reserved 11: Enabled. When AHBDIV or SCLKSEL bit is modified, the auto step-by-step system clock switch is activated automatically.
Bit 3:0	Reserved	0xD	resd	Fixed to 0xD. Do not modify.

5 Flash memory controller (FLASH)

5.1 FLASH introduction

Flash memory is divided into three parts: main Flash memory, information block and Flash memory registers.

- Main Flash memory is up to 512 KB.
- Information block consists of 26 KB bootloader and the user system data area. The bootloader uses chip communication peripherals for ISP programming.

Main Flash memory contains bank 1 (512 KB), including 256 sectors, 2 KB per sector.

Table 5-1 Flash memory architecture (512 KB)

Bank		Name	Address range
Main memory	Bank 1 512 KB	Sector 0	0x0800 0000 – 0x0800 07FF
		Sector 1	0x0800 0800 – 0x0800 0FFF
		Sector 2	0x0800 1000 – 0x0800 17FF
		…	…
		Sector 255	0x0807 F800 – 0x0807 FFFF
Information block		26 KB bootloader	0x1FFF 7C00 – 0x1FFF E3FF
		OTP DATA 4KB	0x1FFF E400 – 0x1FFF F3FF
		OTP LOCK 128B	0x1FFF F500 – 0x1FFF F57F
		512 B user system data	0x1FFF F800 – 0x1FFF F9FF

Main Flash memory contains bank 1 (256 KB), including 128 sectors, 2 KB per sector.

Table 5-2 Flash memory architecture (256 KB)

Bank		Name	Address range
Main memory	Bank 1 256 KB	Sector 0	0x0800 0000 – 0x0800 07FF
		Sector 1	0x0800 0800 – 0x0800 0FFF
		Sector 2	0x0800 1000 – 0x0800 17FF
		…	…
		Sector 127	0x0803 F800 – 0x0803 FFFF
Information block		26 KB bootloader	0x1FFF 7C00 – 0x1FFF E3FF
		OTP DATA 4KB	0x1FFF E400 – 0x1FFF F3FF
		OTP LOCK 128B	0x1FFF F500 – 0x1FFF F57F
		512 B user system data	0x1FFF F800 – 0x1FFF F9FF

User system data area

The system data will be read from the information block of Flash memory whenever a system reset occurs, and it is saved in the user system data register (FLASH_USD) and erase/programming protection status register (FLASH_EPPS).

Each system data occupies two bytes, where the low bytes correspond to the contents in the system data area, and the high bytes represent the inverse code that is used to verify the correctness of the selected bit. When the high byte is not equal to the inverse code of the low byte (except when both high and low byte are all 0xFF), the system data loader will issue a system data error flag (USDERR) and the corresponding system data and their inverse codes are forced 0xFF.

Note: The update of the contents in the user system data area becomes effective only after a system reset.

Table 5-3 User system data area

Address	Bit	Description	
e0x1FFF_F800	[7:0]	**FAP[7:0]**: Flash memory access protection (Access protection enable/disable result is stored in the FLASH_USD [1] and [26]) 0xA5: Flash access protection disabled 0xCC: High-level Flash access protection enabled Others: Low-level Flash access protection enabled	
	[15:8]	**nFAP[7:0]**: Inverse code of FAP[7:0]	
	[23:16]	**SSB[7:0]**: System setting byte (stored in the FLASH_USD[9:2])	
		Bit 7 (nRAM_PRT_CHK)	0: Enable RAM parity check 1: Disable RAM parity check
		Bit 6 (nSTDBY_WDT)	0: WDT stops counting while entering Standby mode 1: WDT does not stop counting while entering Standby mode
		Bit 5 (nDEPSLP_WDT)	0: WDT stops counting while entering Deepsleep mode 1: WDT does not stop counting while entering Deepsleep mode
		Bit 4: 3	Reserved
		Bit 2 (nSTDBY_RST)	0: Reset occurs when entering Standby mode 1: No reset occurs when entering Standby mode
		Bit 1 (nDEPSLP_RST)	0: Reset occurs when entering Deepsleep mode 1: No reset occurs when entering Deepsleep mode
		Bit 0 (nWDT_ATO_EN)	0: Watchdog is enabled 1: Watchdog is disabled
	[31:24]	**nSSB[7:0]**: Inverse code of SSB[7:0]	
0x1FFF_F804	[7:0]	**Data0[7:0]**: User data 0 (stored in the FLASH_USD[17:10])	
	[15:8]	**nData0[7:0]**: Inverse code of Data0[7:0]	
	[23:16]	**Data1[7:0]**: User data 1 (stored in the FLASH_USD[25:18])	
	[31:24]	**nData1[7:0]**: Inverse code of Data1[7:0]	
0x1FFF_F808	[7:0]	**EPP0[7:0]**: Flash erase/write protection byte 0 (stored in the FLASH_EPPS[7:0]) This field is used to protect sector0~sector15 of the main Flash memory. Each bit takes care of 4 KB sectors. 0: Erase/write protection is enabled 1:擦写保护解除	
	[15:8]	**nEPP0[7:0]**: Inverse code of EPP0[7:0]	
	[23:16]	**EPP1[7:0]**: Flash erase/write protection byte1 (stored in the FLASH_EPPS[15:8]) This field is used to protect sector16~sector31 of the main Flash memory. Each bit takes care of 4 KB sectors. 0: Erase/write protection is enabled 1: Erase/write protection is disabled	
	[31:24]	**nEPP1[7:0]**: Inverse code of EPP1[7:0]	

0x1FFF_F80C	[7:0]	**EPP2[7:0]**: Flash erase/write protection byte2 (store in the FLASH_EPPS[23:16]) This field is used to protect sector32~sector47 of the main Flash memory. Each bit takes care of 4 KB sectors. 0: Erase/write protection is enabled 1: Erase/write protection is disabled
	[15:8]	**nEPP2[7:0]**: Inverse code of EPP2[7:0]
	[23:16]	**EPP3[7:0]**: Flash erase/write protection byte3 (store in the FLASH_EPPS[31:24]) Bit [6:0] is used to protect sector48~sector61 of the main Flash memory. Each bit takes care of 4 KB sectors. Bit [7] is used to protect sector62~sector255 of the main Flash memory (512 KB) and sector62~sector127 of the main Flash memory (256 KB). Bit [7] is also used for the main Flash memory (512 KB/256 KB) extension area. 0: Erase/write protection is enabled 1: Erase/write protection is disabled
	[31:24]	**nEPP3[7:0]**: Inverse code of EPP3[7:0]
0x1FFF_F810 ~ 0x1FFF_F830	[31:0]	Reserved
0x1FFF_F834	[7:0]	**QSPIKEY0[7:0]**: Quad SPI (QSPI) ciphertext access area encryption key byte 0 The situations for non-encryption includes: QSPIKEYx and nQSPIKEYx are 0xFF (default erase status) Write 0x00 to QSPIKEYx Write 0xFF to QSPIKEYx That is, {nQSPIKEYx, QSPIKEYx} are all 0xFFFF, 0xFF00, 0x00FF. Among them, QSPIKEY0-QSPIKEY3 are the encryption key of QSPI1.
	[15:8]	**nQSPIKEY0[7:0]**: Inverse code of QSPIKEY0[7:0]
	[23:16]	**QSPIKEY1[7:0]**: Quad SPI (QSPI) ciphertext access area encryption key byte 1
	[31:24]	**nQSPIKEY1[7:0]**: Inverse code of QSPIKEY1[7:0]
0x1FFF_F838	[7:0]	**QSPIKEY2[7:0]**: Quad SPI (QSPI) ciphertext access area encryption key byte 2
	[15:8]	**nQSPIKEY2[7:0]**: Inverse code of QSPIKEY2[7:0]
	[23:16]	**QSPIKEY3[7:0]**: Quad SPI (QSPI) ciphertext access area encryption key byte 3
	[31:24]	**nQSPIKEY3[7:0]**: Inverse code of QSPIKEY3[7:0]
0x1FFF_F83C ~ 0x1FFF_F848	[31:0]	Reserved
0x1FFF_F84C	**Check to confirm that both BOOT_PERIP_KEY1 and BOOT_PERIP_KEY2 are valid before setting the BOOT_PERIP1_EN and BOOT_PERIP2_EN. For other cases, all Bootloader peripherals are enabled, by default.**	
	[7:0]	**BOOT_PERIP_KEY1[7:0]**: Bootloader peripheral enable KEY1 0x4B: Valid Others: Invalid (all Bootloader peripherals are enabled, by default)
	[15:8]	**nBOOT_PERIP_KEY1[7:0]**: Inverse code of BOOT_PERIP_KEY1 [7:0]

	[23:16]	**BOOT_PERIP_KEY2[7:0]**:Bootloader peripheral enable KEY2 0x5C: Valid Others: Invalid (all Bootloader peripherals are enabled, by default)	
	[31:24]	**nBOOT_PERIP_KEY2 [7:0]**: Inverse code of BOOT_PERIP_KEY2 [7:0]	
0x1FFF_F850	[7:0]	**BOOT_PERIP_EN1[7:0]:** Bootloader peripheral enable 1	
		Bit 7 (BOOT_I2C3_EN)	Bootloader I²C3 enable 0: Disable I²C3 1: Enable I²C3
		Bit 6 (BOOT_I2C2_EN)	Bootloader I²C2 enable 0: Disable I²C2 1: Enable I²C2
		Bit 5 (BOOT_I2C1_EN)	Bootloader I²C1 enable 0: Disable I²C1 1: Enable I²C1
		Bit 4	Reserved
		Bit 3 (BOOT_USB1_EN)	Bootloader USB DFU enable 0: Disable USB DFU 1: Enable USB DFU
		Bit 2 (BOOT_USART3_EN)	Bootloader USART3 enable 0: Disable USART3 1: Enable USART3
		Bit 1 (BOOT_USART2_EN)	Bootloader USART2 enable 0: Disable USART2 1: Enable USART2
		Bit 0 (BOOT_USART1_EN)	Bootloader USART1 enable 0: Disable USART1 1: Enable USART1
	[15:8]	**nBOOT_PERIP1_EN[7:0]**: Inverse code of BOOT_PERIP_EN1 [7:0]	
	[23:16]	**BOOT_PERIP_EN2[7:0]**:Bootloader peripheral enable 2	
		Bit [7: 4]	Reserved
		Bit 3 (BOOT_SPI2_EN)	Bootloader SPI2 enable 0: Disable SPI2 1: Enable SPI2
		Bit 2 (BOOT_SPI1_EN)	Bootloader SPI1 enable 0: Disable SPI1 1: Enable SPI1
		Bit 1 (BOOT_CAN2_EN)	Bootloader CAN2 enable 0: Disable CAN2 1: Enable CAN2
		Bit 0 (BOOT_CAN1_EN)	Bootloader CAN1 enable 0: Disable CAN1 1: Enable CAN1
	[31:24]	**nBOOT_PERIP2_EN[7:0]**: Inverse code of nBOOT_PERIP2_EN [7:0]	
0x1FFF_F854	[7:0]	**Data2[7:0]**: User data 2	
	[15:8]	**nData2[7:0]**: Inverse code of Data2[7:0]	

	[23:16]	**Data3[7:0]**: User data 3
	[31:24]	**nData3[7:0]**: Inverse code of Data3[7:0]
..	..	..
0x1FFF_F9FC	[7:0]	**Data214[7:0]**: User data 214
	[15:8]	**nData214[7:0]**: Inverse code of Data214[7:0]
	[23:16]	**Data215[7:0]**: User data 215
	[31:24]	**nData215[7:0]**: Inverse code of Data215[7:0]

5.2 Flash memory operation

5.2.1 Unlock/lock

After reset, Flash memory is protected, by default. The FLASH_CTRL register cannot be written. Write and erase operation can be performed only when the Flash memory is unlocked.

Unlock procedure:

Flash memory block can be unlocked by writing KEY1 (0x45670123) and KEY2 (0xCDEF89AB) to the FLASH_UNLOCK register.

Note: Writing an incorrect key sequence leads to a bus error and the Flash memory is also locked until the next reset.

Lock procedure:

Flash memory block can be locked by setting the OPLK bit in the FLASH_CTRL register.

5.2.2 Erase operation

Erase operation must be done before programming. Flash memory erase includes sector erase and mass erase.

Sector erase

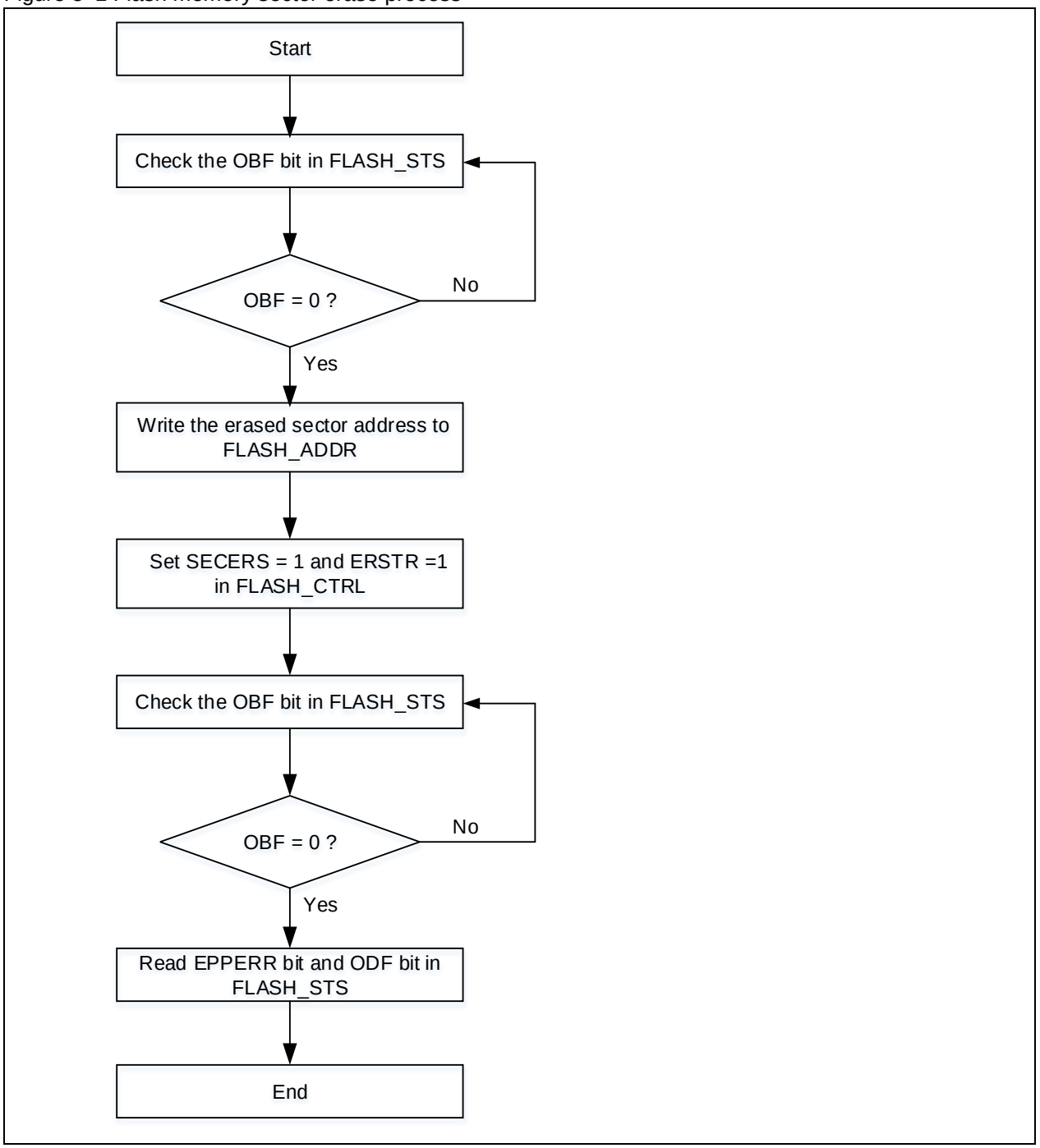
Any sector in the Flash memory and the Flash memory extension area can be erased with sector erase function independently.

The following process is recommended:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Write the sector to be erased in the FLASH_ADDR register;
- Set the SECERS and ERSTR bits in the FLASH_CTRL register to enable sector erase;
- Wait until the OBF bit in the FLASH_STS register becomes “0”. Read the EPPERR and ODF bits in the FLASH_STS register to verify the erased sectors.

Note: When the boot memory is configured as the Flash memory extension area, performing sectorerase operation erases the entire Flash memory extension area.

Figure 5-1 Flash memory sector erase process



Mass erase

Mass erase function can be used to erase all the Flash memory.

The following process is recommended:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Set the BANKERS and ERSTR bits in the FLASH_CTRL register to enable mass erase;
- Wait until the OBF bit in the FLASH_STS register becomes “0”. Read the EPPERR and ODF bits in the FLASH_STS register to verify the erase result.

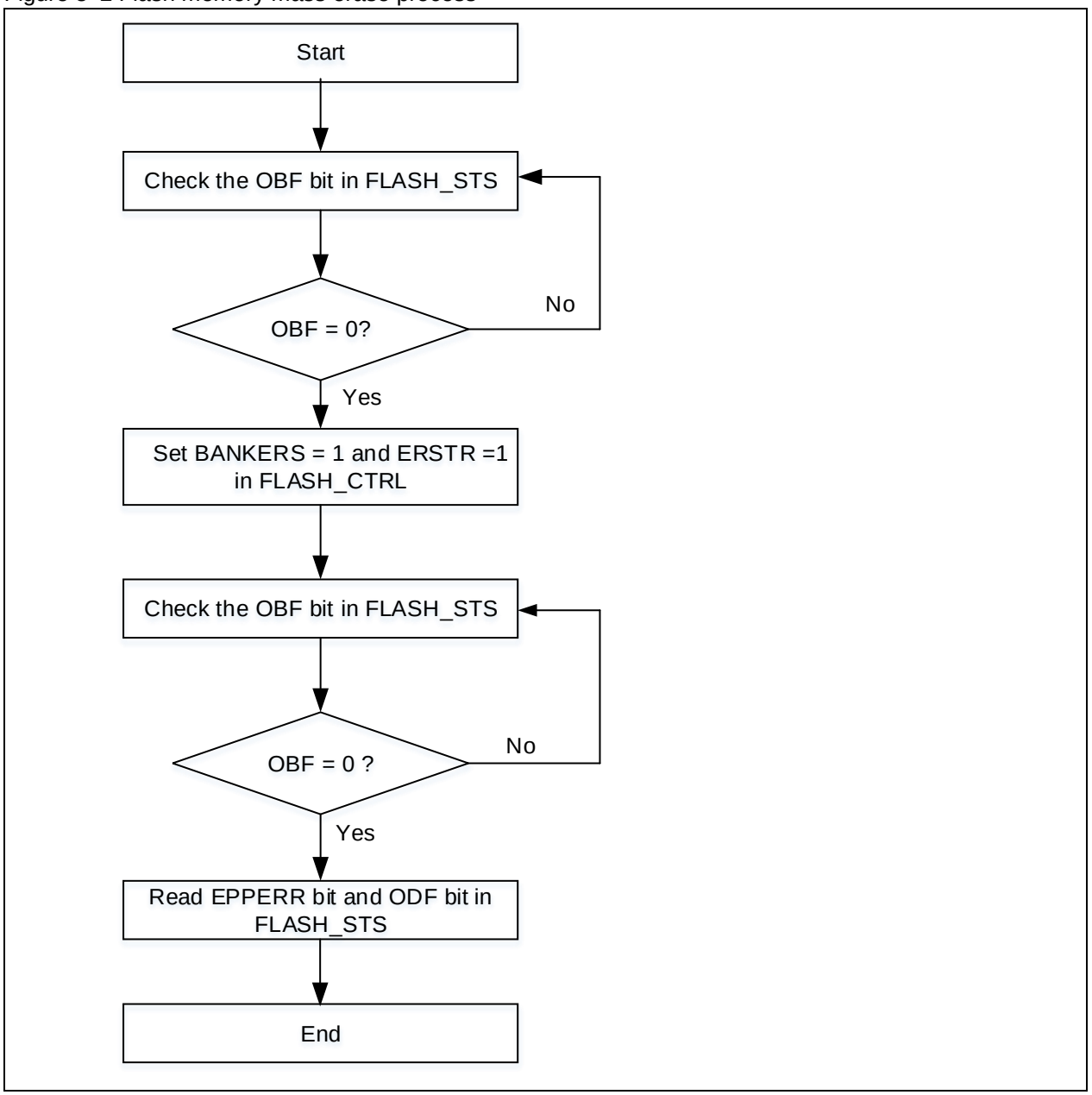
Note:

1) When the boot memory is configured as the Flash memory extension area, performing masserase operation erases automatically the entire Flash memory and its extension area.

2) Read access during erase operation halts the CPU and waits until the completion of erase.

3) Internal HICK must be enabled prior to erase operation.

Figure 5-2 Flash memory mass erase process



5.2.3 Programming operation

The Flash memory can be programmed with 32 bits, 16 bits or 8 bits at a time.

The following process is recommended:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Set FPRGM=1 int eh FLASH_CTRL register, so that the Flash memory programming instructions can be received;
- Write the data (word/half-word/byte) to be programmed to the designated address;
- Wait until the OBF bit in the FLASH_STS register becomes “0”. Read the EPPERR, PRGMERR and ODF bits in the FLASH_STS register to verify the programming result.

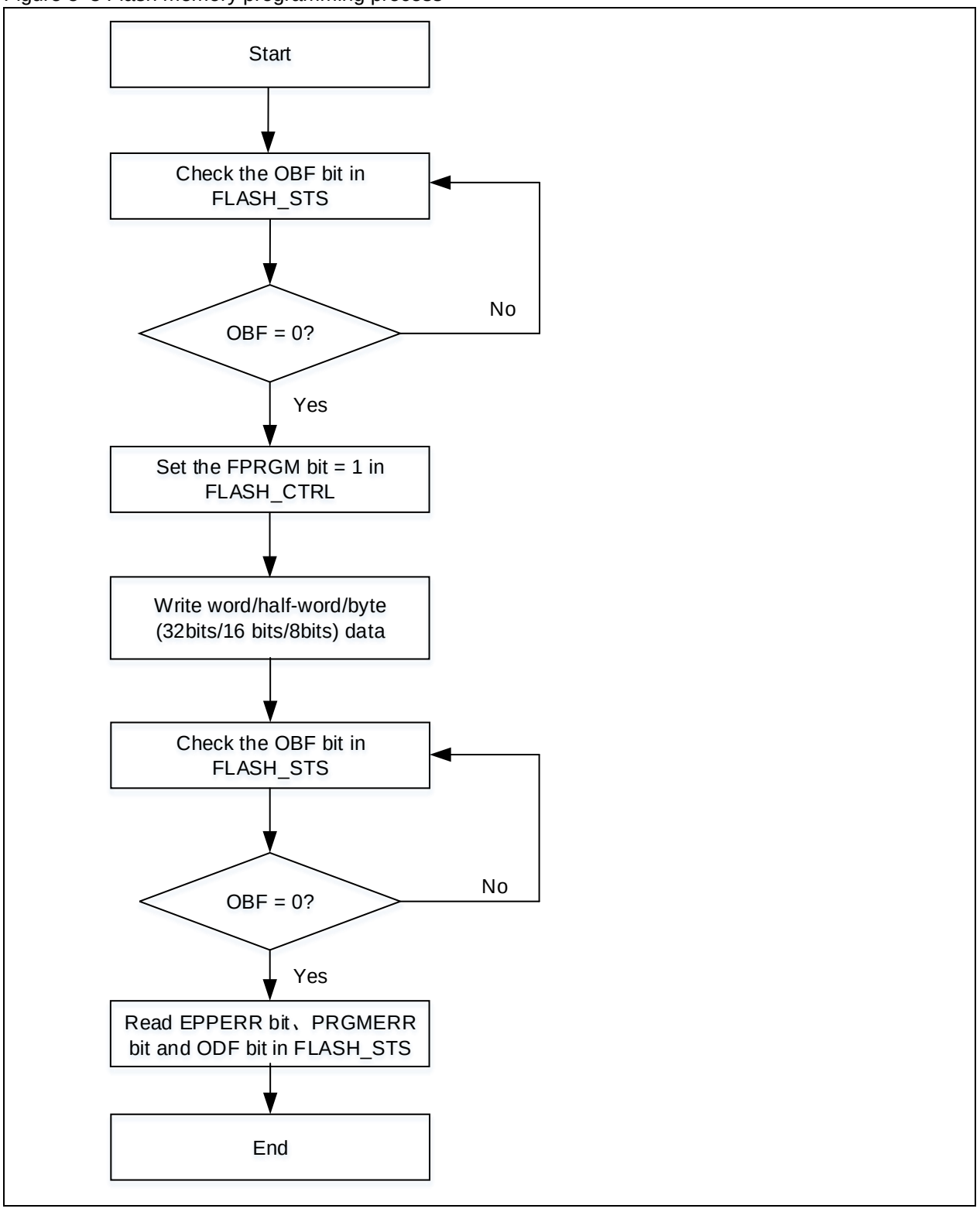
Note:

1) When the address to be written is not erased in advance, the programming operation is not executed unless the data to be written is all 0. In this case, a programming error is reported by the PRGMERR bit in the FLASH_STS register.

2) CPU halts during programming the and waits until the completion of programming.

3) Internal HICK must be enabled prior to programming.

Figure 5-3 Flash memory programming process



5.2.4 Read operation

Flash memory can be accessed through AHB bus of the CPU.

5.3 Main Flash memory extension area

Boot memory can also be programmed as the extension area of the main Flash memory to store user-application code. When used as main Flash memory extension area, it behaves like the main Flash memory, including read, unlock, erase and programming operations.

5.4 OTP operation

The OTP DATA and LOCK areas cannot be erased, and they behave like the main Flash memory, including read, unlock, erase and programming operations.

Each 32-byte DATA corresponds to 1-byte LOCK. When a LOCK byte is programmed as 0x00, its corresponding 32-byte DATA cannot be written.

DATA and LOCK can be configured as access-protected. Please refer to Table 5-5 Flash memory access limit for details.

Table 5-4 Correlation between OTP DATA and OTP LOCK

OTP LOCK address range		OTP DATA address range
LOCK0	0x1FFF F500	0x1FFF E400 – 0x1FFF E41F
LOCK1	0x1FFF F501	0x1FFF E420 – 0x1FFF E43F
LOCK2	0x1FFF F502	0x1FFF E440 – 0x1FFF E45F
…	…	
LOCK127	0x1FFF F57F	0x1FFF F3E0 – 0x1FFF F3FF

5.5 User system data area

5.5.1 Unlock/lock

After reset, user system data area is protected, by default. Write and erase operations can be performed only after the Flash memory is unlocked before the unlock operation for the user system data area.

Unlock procedure:

Flash memory can be unlocked by writing KEY1 (0x45670123) and KEY2 (0xCDEF89AB) to the FLASH_UNLOCK register.

When KEY1 (0x45670123) and KEY2 (0xCDEF89AB) are written to the FLASH_USD_UNLOCK register, the USDULKS bit in the FLASH_CTRL register will be automatically set by hardware, indicating that it support write/erase operation to the user system data area.

Note: Writing an incorrect key sequence leads to bus error and the Flash memory is also locked until the next reset.

Lock procedure:

User system data area is locked by clearing the USDULKS bit in the FLASH_CTRL register.

5.5.2 Erase operation

Erase operation must be done before programming. User system data area can be erased independently.

The following process is recommended:

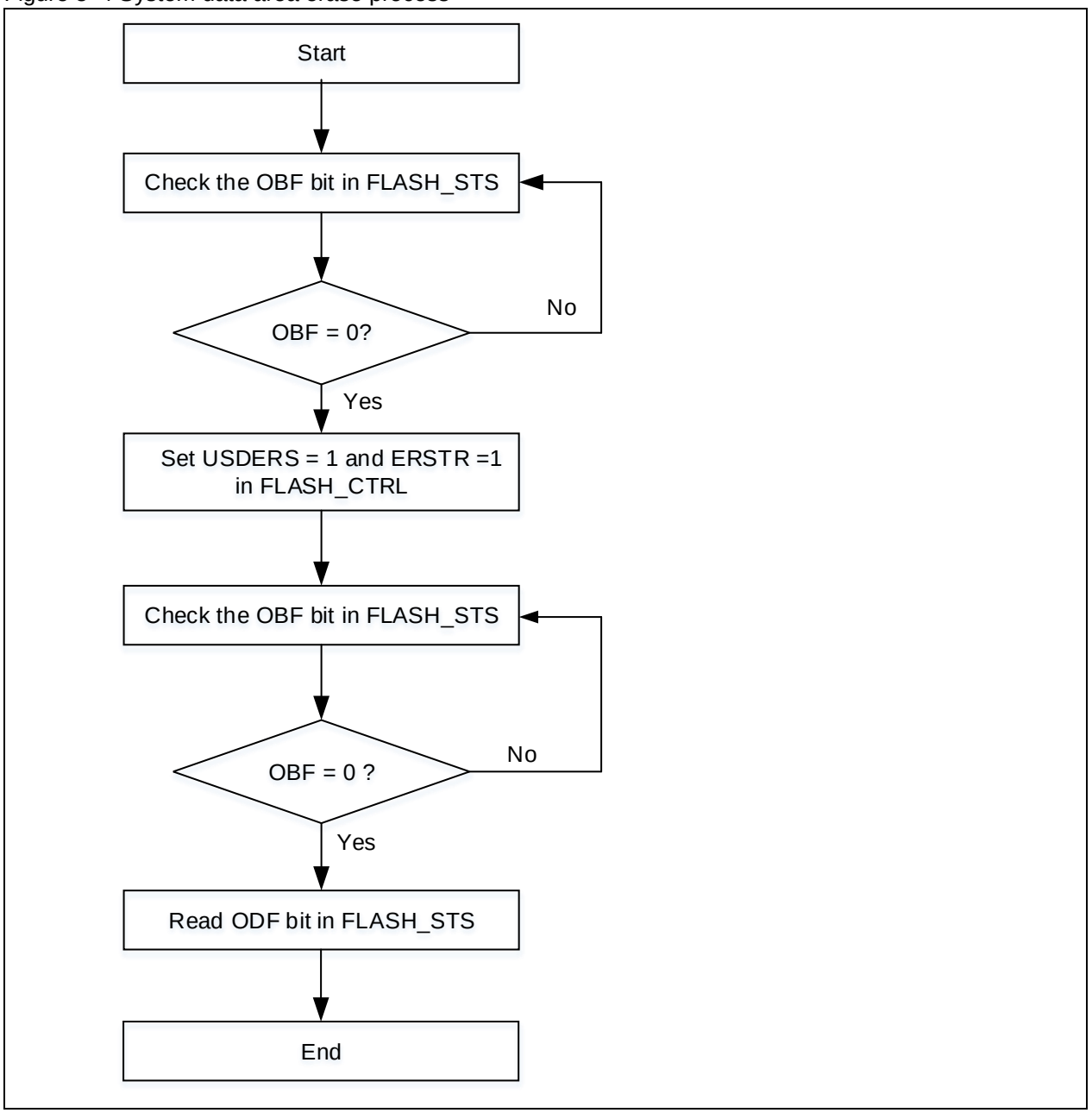
- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Set the USDERS and ERSTR bits in the FLASH_CTRL register to enable erase operation;
- Wait until the OBF bit in the FLASH_STS register becomes “0”. Read the ODF bit in theFLASH_STS register to verify the erase result.

Note:

Read operation to the Flash memory during programming halts CPU and waits until the completion of erase

The internal HICk must be enabled prior to erase operation.

Figure 5-4 System data area erase process



5.5.3 Programming operation

The user system data area can be programmed with 16 bits or 32 bits at a time.

The following process is recommended:

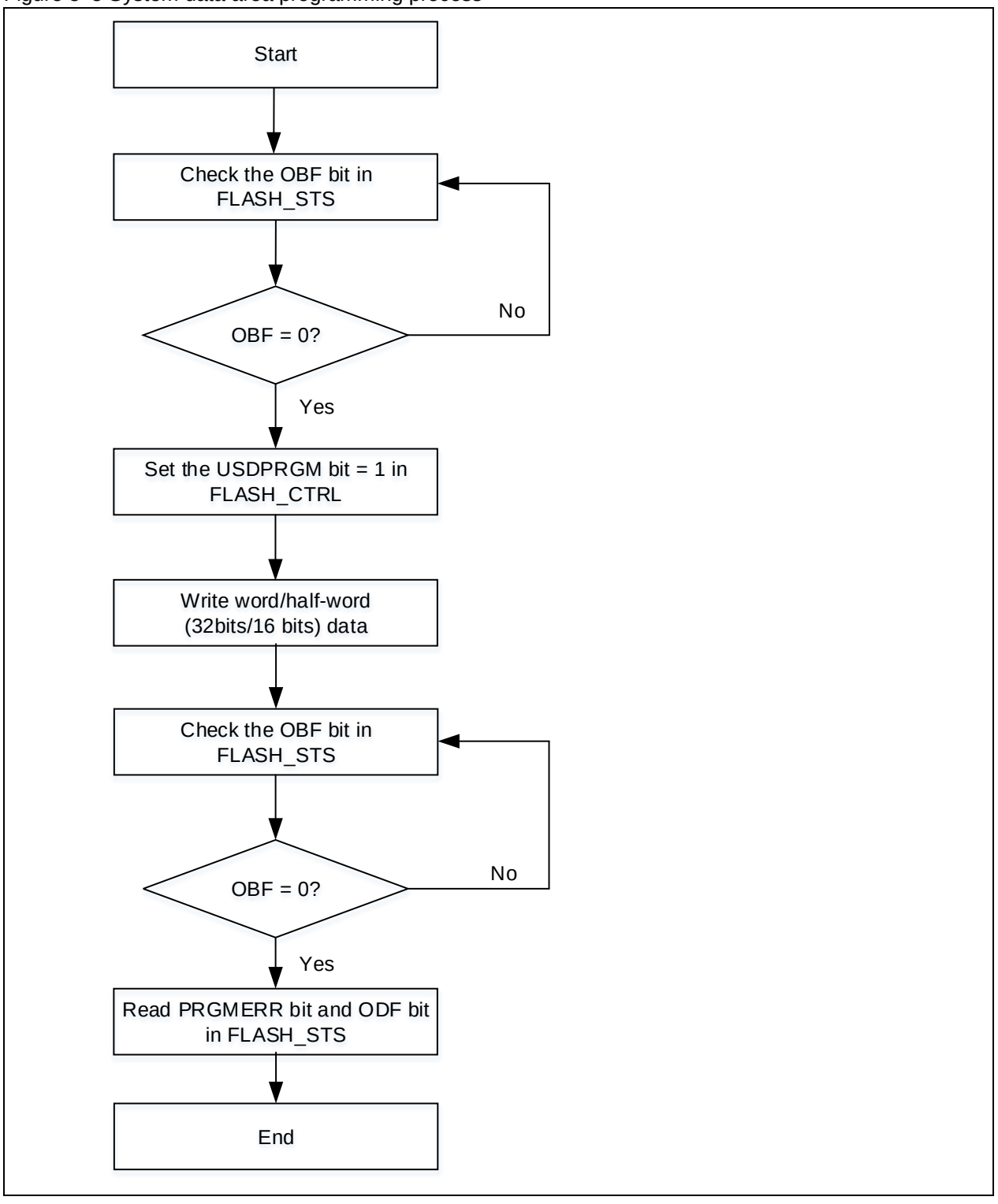
- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Set USDPRGM=1 in the FLASH_CTRL) register, so that the programming instructions for the user system data area can be received;
- Write the data (word/half-word) to be programmed to the designated address;
- Wait until the OBF bit in the FLASH_STS register becomes “0”. Read PRGMERR and ODF bits in the FLASH_STS register to verify the programming result.

Note:

CPU halts during programming and waits until the completion of programming.

The internal HICK must be enabled prior to programming operation.

Figure 5-5 System data area programming process



5.5.4 Read operation

User system data area can be accessed through AHB bus of the CPU.

5.6 Flash memory protection

Flash memory protection includes access and erase/program protection.

5.6.1 Access protection

Flash memory access protection is divided into two parts: high-level and low level.

Once enabled, only the Flash program is allowed to read Flash memory data. This read operation is not permitted in debug mode or by booting from non-Flash memory.

Low-level access protection

When the contents in the nFAP and FAP are different from 0x5A and 0xA5, and 0x33 and 0xCC, the low-level Flash memory access protection is enabled after a system reset.

When the Flash access is protected, the user can re-erase the system data area, and unlock Flash low-level access protection (switching from low-level protection to unprotected state will trigger mass erase on the Flash memory and its extension area automatically) by writing 0xA5 to FAP byte, and then perform a system reset. Subsequently, the system data loader will be reloaded with system data and updated with Flash memory access protection disable state (FAP byte).

High-level access protection

When the content in the nFAP is equal to 0x33, and the content in the FAP byte is equal to 0xCC, the high-level Flash memory access protection is enabled after a system reset.

Once enabled, it cannot be unlocked, and it is not permissible for users to re-erase and write the system data area.

Note:

1) The main Flash memory extension area can also be protected.

2) If the access protection bit is set in debug mode, then the debug mode has to be cleared by POR instead of system reset in order to resume access to Flash memory data.

Table 5-5 shows Flash memory access limits when Flash access protection is enabled.

Table 5-5 Flash memory access limit

Block	Protection level	Access limits					
		In debug mode or boot from SRAM or boot memory			Boot from main Flash memory		
		Read	Write	Erase	Read	Write	Erase
Main Flash memory	Low-level protection	Not allowed		Not allowed (1) (2)	Accessible		
	High-level protection	None (3)			Accessible		
User system data	Low-level protection	Not allowed	Accessible		Accessible		
	High-level protection	None (3)			Accessible	Not allowed	
OTP and LOCK area	Low-level protection	Not allowed			Accessible		Not allowed
	High-level protection	None (3)			Accessible		Not allowed

(1) Main Flash memory is cleared automamtically by hardware only when the acess protection is disabled;
(2) Only sector erase is forbidden, and mass erase is not affected;
(3) When the high-level access protection is enabled, the system automatically boots from the main Flash memory.

5.6.2 Erase/program protection

Erase/program protection is performed on the basis of 4 KB. This is used to protect the contents in the Flash memory against inadvertent operation when the program crash occurs.

Erase/program operation is not permitted under one of the following events, and the EPPERR bit is set accordingly when:

- Erasing/programming the pages (in Flash memory and its extension area) where erase/program protection is enabled;
- Performing mass erase on the sectors and main Flash memory extension area where erase/program protection enabled;
- When the Flash access protection is enabled, the sector0~sector1 in the main Flash memory will be protected against sector erase/programming automatically;
- When the Flash access protection is enabled, the main Flash memory is protected against sector erase and programming operation when the main Flash memory and its extension area are in debug mode or when it boots from non-main Flash memory.

5.7 Read access

To increase system clock frequency, program the number of wait states to access the Flash memory through the WTCYC bit in the FLASH_PSR register.

The Flash read times can be decreased through the PFT_EN, PFT_EN2 and PFT_LAT_DIS bits in the FLASH_PSR register.

5.8 Special functions

5.8.1 Security library settings

Security library is a defined area protected by a code in the main memory. This area is only executable but cannot be written or deleted unless a correct code is keyed in. Security library includes instruction security library (cannot be read) and data security library (can be read).

Advantages of security library:

Security library is protected by codes so that solution providers can program core algorithm into this area;

Security library cannot be read or deleted (including ISP/IAP/SWD) but only executed unless the code defined by the solution provider is keyed in;

The rest of the area can be used for secondary development by solution providers;

Solution providers can sell core algorithm with security library function and do not have to develop full solutions for every customer;

Security library helps prevent from deliberate damage or changing terminal application codes

Note:

Security library code must be programmed by sector, with its start address aligned with the main memory address;

Only CPU instruction is allowed to read instruction security library;

In an attempt of writing or erasing the security library code, a warning message will be issued by EPPERR=1 in the FLASH_STS register;

Executing mass erase in the main memory will not erase the security library.

By default, security library setting register is unreadable and write protected. To enable write access to this register, security library should be unlocked first, by writing 0xA35F6D24 to the SLIB_UNLOCK register, and checking the SLIB_ULKF bit in the SLIB_MISC_STS register to verify if it is unlocked successfully, and then writing the programmed value to the security library setting register.

The steps to enable security library are as follows:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;

- Writing 0xA35F6D24 to the SLIB_UNLOCK register to unlock the security library;
- Check SLIB_ULKF bit in the SLIB_MISC_STS register to confirm that it is unlocked successfully;
- If the security library is set in the main Flash memory, set the sectors to be protected in the SLIB_SET_RANGE register (including the addresses of instruction/data security library); if the security library is set in the main Flash memory extension area, set the EM_SLIB_SET register;
- Wait until the OBF bit becomes “0”;
- Set the security library password in the SLIB_SET_PWD register;
- Wait until the OBF bit becomes “0”;
- Program the code to be saved in security library;
- Perform a system reset, and then reload security library setting word;
- Read the SLIB_STS0/STS1 register to verify the security library settings.

Note:

The main Flash memory and its extension area cannot be set as security library at the same time.

Security library should be enabled when the Flash access protection is not activated.

The steps to unlock security library are as follows:

- Write the previously set security library password to the SLIB_PWD_CLR register;
- Wait until the OBF bit becomes “0”;
- Perform a system reset, and then reload security library setting word;
- Read the SLIB_STS0 register to verify that if the security library is unlocked successfully.

Note: Disabling the security library will automatically perform mass erase for the main Flash memory and its extension, as well as the security library setting block.

5.8.2 Boot memory used as Flash memory extension

There is only one chance for users to program the boot memory as the main Flash extension area, which will have the same features as those of Flash memory after successful configuration as follows:

- Read bit [0] in the SLIB_STS0 register to get the current mode of boot memory;
- Write 0xA35F6D24 to the SLIB_UNLOCK register to unlock the current mode of boot memory;
- Write non-0xFF to bit [7:0] in the BTM_MODE_SET register;
- Wait until the OBF bit becomes “0”;
- Perform a system reset, and then reload setting words;
- Read the SLIB_STS0 register to verify the settings.

Note: The above-mentioned process must be performed when the Flash memory access protection not activated.

Once the boot memory is used as Flash memory extension area, the “boot from boot memory” is forced to “boot from main Flash memory”.

5.8.3 CRC verify

The optional CRC check for security library code or user code is performed on a sector level.

- Generator polynomial: 0x4C11DB7,

  That is, $X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$
- CRC initial value: 0x00000000

CRC verify procedure is as follows:

- Check the OBF bit in the FLASH_STS register to confirm that there is no other programming operation in progress;
- Program the start address of the code for CRC check in the FLASH_CRC_ADDR register;
- Program the code count (in terms of sectors) for CRC check through the bit [15:0] in the FLASH_CRC_CTRL register;
- Enable CRC verify by setting the bit [16] in the FLASH_CRC_CTRL register;
- Wait until the OBF bit becomes “0”;
- Read the FLASH_CRC_CHKR register to verify the result.

Note: The values of FLASH_CRC_ADDR register must be aligned with the start address of the sector.

CRC verify must not cross the main Flash memory and its extension area.

5.9 FLASH registers

These peripheral registers must be accessed by words (32 bits).

Table 5-6 Flash memory register map and reset value

Register	Offset	Reset value
FLASH_PSR	0x00	0x0000 01F0
FLASH_UNLOCK	0x04	0xXXXX XXXX
FLASH_USD_UNLOCK	0x08	0xXXXX XXXX
FLASH_STS	0x0C	0x0000 0000
FLASH_CTRL	0x10	0x0002 0080
FLASH_ADDR	0x14	0x0000 0000
FLASH_USD	0x1C	0x03FF FFFC
FLASH_EPPS	0x20	0xFFFF FFFF
SLIB_STS0	0x74	0x00FF 0000
SLIB_STS1	0x78	0xFFFF FFFF
SLIB_PWD_CLR	0x7C	0xFFFF FFFF
SLIB_MISC_STS	0x80	0x0000 0000
FLASH_CRC_ADDR	0x84	0x0000 0000
FLASH_CRC_CTRL	0x88	0x0000 0000
FLASH_CRC_CHKR	0x8C	0x0000 0000
SLIB_SET_PWD	0x160	0x0000 0000
SLIB_SET_RANGE	0x164	0x0000 0000
EM_SLIB_SET	0x168	0x0000 0000
BTM_MODE_SET	0x16C	0x0000 0000
SLIB_UNLOCK	0x170	0x0000 0000

5.9.1 Flash performance select register (FLASH_PSR)

Bit	Name	Reset value	Type	Description
Bit 31:9	Reserved	0x000000	resd	Kept as its default value.
Bit 8	PFT_LAT_DIS	1	rw	Prefetch latency disable 0: Prefetch buffer latency enabled, one system clock cycle for buffer access 1: Prefetch buffer latency disabled, zero-wait state for buffer access It is recommended to set this bit to 1 and do not change.
Bit 7	PFT_ENF2	1	ro	Prefetch enabled flag2 When this bit is set, it indicates that the prefetch buffer 2 is enabled.
Bit 6	PFT_EN2	1	rw	Prefetch enable2 0: Prefetch buffer 2 is disabled 1: Prefetch buffer 2 is enabled It is recommended to set this bit to 1 and do not change.
Bit 5	PFT_ENF	1	ro	Prefetch enabled flag

				When this bit is set, it indicates that the prefetch buffer is enabled.
Bit 4	PFT_EN	1	rw	Prefetch enable 0: Prefetch buffer is disabled 1: Prefetch buffer is enabled
Bit 3	Reserved	0	resd	Kept as its default value.
Bit 2:0	WTCYC	0x0	rw	Wait cycle The wait states depend on the size of the system clock, and they are in terms of system clocks. 000: Zero wait state, 0 MHz< system clock≤32 MHz 001: One wait state, 32 MHz< system clock≤64 MHz 010: Two wait states, 64 MHz< system clock≤96 MHz 011: Three wait states, 96 MHz< system clock≤128 MHz 100: Four wait states,128 MHz< system clock≤160 MHz 101: Five wait states,160 MHz< system clock≤192 MHz 110: Six wait states,192 MHz< system clock≤224 MHz 111: Seven wait states, 224 MHz<system clock≤256MHz

5.9.2 Flash unlock register (FLASH_UNLOCK)

Bit	Name	Reset value	Type	Description
Bit 31:0	UKVAL	0xXXXX XXXX	wo	Unlock key value This is used to unlock the Flash memory and its extension area.

Note: All these bits are write-only, and return 0 when being read.

5.9.3 Flash user system data unlock register (FLASH_USD_UNLOCK)

Bit	Name	Reset value	Type	Description
Bit 31:0	USD_UKVAL	0xXXXX XXXX	wo	User system data unlock key value

Note: All these bits are write-only, and return 0 when being read.

5.9.4 Flash status register (FLASH_STS)

Bit	Name	Reset value	Type	Description
Bit 31:6	Reserved	0x0000000	resd	Kept as its default value.
Bit 5	ODF	0	rwc1	Operation done flag This bit is set by hardware when Flash memory operations (program/erase) are completed. It is cleared by writing “1”
Bit 4	EPPERR	0	rwc1	Erase/program protection error This bit is set by hardware when erasing or programming the erase/program-protected Flash memory address. It is cleared by writing “1”.
Bit 3	Reserved	0	resd	Kept as its default value.

Bit 2	PRGMERR	0	rwc1	Program error When the programming address is not in erase state, this bit is set by hardware. It is cleared by writing “1”.
Bit 1	Reserved	0	resd	Kept as its default value.
Bit 0	OBF	0	ro	Operation busy flag When this bit is set, it indicates that Flash memory operations are in progress. It is cleared when operations are completed.

5.9.5 Flash control register (FLASH_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31:13	Reserved	0x0000	resd	Kept as its default value.
Bit 12	ODFIE	0	rw	Operation done flag interrupt enable 0: Disabled 1: Enabled
Bit 11,8,3	Reserved	0	resd	Kept as its default value.
Bit 10	ERRIE	0	rw	Error interrupt enable This bit enables EPPERR or PRGMERR interrupt. 0: Disabled 1: Enabled
Bit 9	USDULKS	0	rw	User system data unlock success This bit is set by hardware when the user system data is unlocked properly, indicating that erase/program operation to the user system data is allowed. This bit is cleared by writing “0”, which will re-lock the user system data area.
Bit 7	OPLK	1	rw	Operation lock This bit is set by default, indicating that Flash memory is protected against operations. This bit is cleared by hardware after unlock, indicating that erase/program operation to Flash memory is allowed. Writing “1” can relock the Flash memory
Bit 6	ERSTR	0	rw	Erase start An erase operation is triggered when this bit is set by software. This bit is cleared by hardware after the completion of the erase operation.
Bit 5	USDERS	0	rw	User system data erase It indicates the user system data erase operation.
Bit 4	USDPRGM	0	rw	User system data program It indicates the user system data programming operation.
Bit 2	BANKERS	0	rw	Bank erase It indicates bank erase operation.
Bit 1	SECERS	0	rw	Sector erase It indicates sector erase operation.
Bit 0	FPRGM	0	rw	Flash program It indicates Flash programming operation.

5.9.6 Flash address register (FLASH_ADDR)

Bit	Name	Reset value	Type	Description
Bit 31:0	FA	0x0000 0000	wo	Flash address Select the address of the sectors to be erased.

5.9.7 User system data register (FLASH_USD)

Bit	Name	Reset value	Type	Description
Bit 31:27	Reserved	0x00	resd	Kept as its default value.
Bit 26	FAP_HL	0	ro	Flash access protection high level The statue of the Flash access protection is determined by bit [26] and bit [1]. 00: Flash access protection disabled, and FAP=0xA5 01: Low-level Flash access protection enabled, and FAP = non-0xCC and non-0xA5 10: Reserved 11: High-level Flash access protection enabled, and FAP=0xCC
Bit 25:18	USER_D1	0xFF	ro	User data 1
Bit 17:10	USER_D0	0xFF	ro	User data 0
Bit 9:2	SSB	0xFF	ro	System setting byte It includes the system setting bytes in the loaded user system data area. Bit 7: nRAM_PRT_CHK Bit 6: nSTDBY_WDT Bit 5: nDEPSLP_WDT Bit 4: Unused Bit 3: Unused Bit 2: nSTDBY_RST Bit 1: nDEPSLP_RST Bit 0: nWDT_ATO_EN
Bit 1	FAP	0	ro	Flash access protection
Bit 0	USDERR	0	ro	User system data error When this bit is set, it indicates that certain byte does not match its inverse code in the user system data area. At this point, this byte and its inverse code will be forced to 0xFF when being read.

5.9.8 Erase/program protection status register (FLASH_EPPS)

Bit	Name	Reset value	Type	Description
Bit 31:0	EPPS	0xFFFF FFFF	ro	Erase/program protection status This register reflects the erase/program protection byte status in the loaded user system data.

5.9.9 Flash security library status register 0 (SLIB_STS0)

For Flash security library only.

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	resd	Kept as its default value.
Bit 23:16	EM_SLIB_INST_SS	0xFF	ro	Extension memory sLib instruction start sector 00000000: Sector 0 00000001: Sector 1 00000010: Sector 2 … 00001100: Sector 12 11111111: None Others: Invalid
Bit 15:4	Reserved	0x000	resd	Kept as its default value.
Bit 3	SLIB_ENF	0	ro	sLib enabled flag When this bit is set, it indicates that the main Flash memory is partially or completely (depending on the setting of SLIB_STS1) used as security library
Bit 2	EM_SLIB_ENF	0	ro	Extension memory sLib enabled flag When this bit is set, it indicates that the boot memory is used as the Flash extension area (BTM_AP_ENF is set) and stores security library code.
Bit 1	Reserved	0	resd	Kept as its default value.
Bit 0	BTM_AP_ENF	0	ro	Boot memory store application code enabled flag When this bit is set, it indicates that the boot memory can be used as main Flash extension area to store user application code; otherwise, it is only used for system boot code.

5.9.10 Flash security library status register 1 (SLIB_STS1)

For Flash security library only.

Bit	Name	Reset value	Type	Description
Bit 31:22	SLIB_ES	0x3FF	ro	sLib end sector 0000000000: Sector 0 0000000001: Sector 1 0000000010: Sector 2 … 0001111111: Sector 127 (the last sector of 256 KB main Flash memory) … 0011111111: Sector 255 (the last sector of 512 KB main Flash memory)
Bit 21:11	SLIB_INST_SS	0x7FF	ro	sLib instruction start sector 00000000000: Sector 0 00000000001: Sector 1 00000000010: Sector 2 … 00001111111: Sector 127 (the last sector of 256 KB main Flash memory) … 00011111111: Sector 255 (the last sector of 512 KB

				main Flash memory) 11111111111: None
Bit 10:0	SLIB_SS	0x7FF	ro	sLib start sector 00000000000: Sector 0 00000000001: Sector 1 00000000010: Sector 2 … 00001111111: Sector 127 (the last sector of 256 KB main Flash memory) … 00011111111: Sector 255 (the last sector of 512 KB main Flash memory)

5.9.11 Security library password clear register (SLIB_PWD_CLR)

For Flash security library only.

Bit	Name	Reset value	Type	Description
Bit 31:0	SLIB_PCLR_VAL	0x0000 0000	wo	sLib password clear value This register is used to key in a correct sLib password in order to unlock sLib functions. The write status of this register is indicated by bit [0] and bit [1] of the SLIB_MISC_STS register.

5.9.12 Security library additional status register (SLIB_MISC_STS)

For Flash security library only.

Bit	Name	Reset value	Type	Description
Bit 31:3	Reserved	0x0000000	resd	Kept as its default value.
Bit 2	SLIB_ULKF	0	ro	sLib unlock flag When this bit is set, it indicates that sLib-related setting registers can be configured.
Bit 1	SLIB_PWD_OK	0	ro	sLib password ok This bit is set by hardware when the password is correct.
Bit 0	SLIB_PWD_ERR	0	ro	sLib password error This bit is set by hardware when the password is incorrect and the setting value of the password clear register is different from 0xFFFF FFFF. Note: When this bit is set, the hardware will no longer agree to re-program the password clear register until the next reset.

5.9.13 Flash CRC address register (FLASH_CRC_ADDR)

For the main Flash memory and its extension area.

Bit	Name	Reset value	Type	Description
Bit 31:0	CRC_ADDR	0x0000 0000	wo	CRC address The register is used to select the start address of a sector to be CRC checked. Note: The CRC address must align with the sector start address

Note: All these bits are write-only, and return no response when being read.

5.9.14 Flash CRC check control register (FLASH_CRC_CTRL)

For the main Flash memory and its extension area.

Bit	Name	Reset value	Type	Description
Bit 31:17	Reserved	0x0000	resd	Kept as its default value.
Bit 16	CRC_STRT	0	wo	CRC start This bit is used to enable CRC check for user code or sLib code. It is automatically cleared after enabling CRC by hardware. Note: CRC data ranges from CRC_ADDR to CRC_ADDR+CRC_SN*1.
Bit 15:0	CRC_SN	0x0000	wo	CRC sector number This bit is used to set the number of sectors for CRC check.

5.9.15 Flash CRC check result register (FLASH_CRC_CHKR)

For the main Flash memory and its extension area.

Bit	Name	Reset value	Type	Description
Bit 31:0	CRC_CHKR	0x0000 0000	ro	CRC check result

Note: All these bits are read-only, and return no response when being written.

5.9.16 Security library password setting register (SLIB_SET_PWD)

For Flash security library password setting only.

Bit	Name	Reset value	Type	Description
Bit 31:0	SLIB_PSET_VAL	0x0000 0000	wo	sLib password setting value Note: This register can be written only after sLib is unlocked. It is used to set a password of sLib. Writing 0xFFFF_FFFF or 0x0000_0000 has no effect.

Note: All these bits are write-only, and return 0 when being read.

5.9.17 Security library address setting register (SLIB_SET_RANGE)

For Flash security library address setting only.

Bit	Name	Reset value	Type	Description
Bit 31:22	SLIB_ES_SET	0x000	wo	sLib end sector setting These bits are used to set the security library end sector. 0000000000: Sector 0 0000000001: Sector 1 0000000010: Sector 2 … 0001111111: Sector 127 (the last sector of 256 KB main Flash memory) … 0011111111: Sector 255 (the last sector of 512 KB main Flash memory)
Bit 21:11	SLIB_ISS_SET	0x000	wo	sLib instruction start sector setting These bits are used to set the security library instruction start sector 00000000000: Sector 0 00000000001: Sector 1 00000000010: Sector 2 … 00001111111: Sector 127 (the last sector of 256 KB main Flash memory) … 00011111111: Sector 255 (the last sector of 512 KB main Flash memory) 11111111111: None
Bit 10:0	SLIB_SS_SET	0x000	wo	sLib start sector setting These bits are used to set the security library start sector. 00000000000: Sector 0 00000000001: Sector 1 00000000010: Sector 2 … 00001111111: Sector 127 (the last sector of 256

				KB main Flash memory) … 00011111111: Sector 255 (the last sector of 512 KB main Flash memory)

Note:

All these bits are write-only, and return 0 when being read.

This register can be written only when security library is unlocked.

Being out of the Flash address range is an invalid setting.

5.9.18 Flash extension memory security library setting register (EM_SLIB_SET)

For Flash extension only.

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	resd	Kept as its default value.
Bit 23:16	EM_SLIB_ISS_SET	0x000	wo	Extension memory sLib instruction start sector setting These bits are used to set the security library instruction area start sector 00000000: Sector 0 00000001: Sector 1 00000010: Sector 2 … 00001100: Sector 12 11111111: None Others: Invalid Note: When it is set to 0xFF, it indicates that the extension area from sector0 to sector3 is the security library, read-only.
Bit 15:0	EM_SLIB_SET	0x000	wo	Extension memory sLib setting Writing 0x5AA5 can configure the Flash extension area to store the sLib code.

Note: All these bits are write-only, and return no response when being read.

5.9.19 Boot memory mode setting register (BTM_MODE_SET)

For boot memory only.

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	resd	Kept as its default value.
Bit 7:0	BTM_MODE_SET	0x00	wo	Boot memory mode setting 0xFF: Boot memory serves as a system area that stores system boot code Others: Boot memory serves a Flash extension area that stores application code Note: This register is set when the Flash access protection is disabled.

Note: All these bits are write-only, and return no response when being read.

5.9.20 Security library unlock register (SLIB_UNLOCK)

For security library register unlock only.

Bit	Name	Reset value	Type	Description
Bit 31:0	SLIB_UKVAL	0x0000 0000	wo	sLib unlock key value The fixed key value is 0xA35F_6D24, used for security library setting register unlock.

Note: All these bits are write-only, and return 0 when being read.

6 GPIOs and IOMUX

6.1 Introduction

AT32F455 series supports up to 117 bidirectional I/O pins, namely PA0-PA15, PB0-PB15, PC0-PC15, PD0-PD15, PE0-PE15, PF0-PF15, PG0-PG15 and PH0-PH4. Each of these pins features communication, control and data collection.

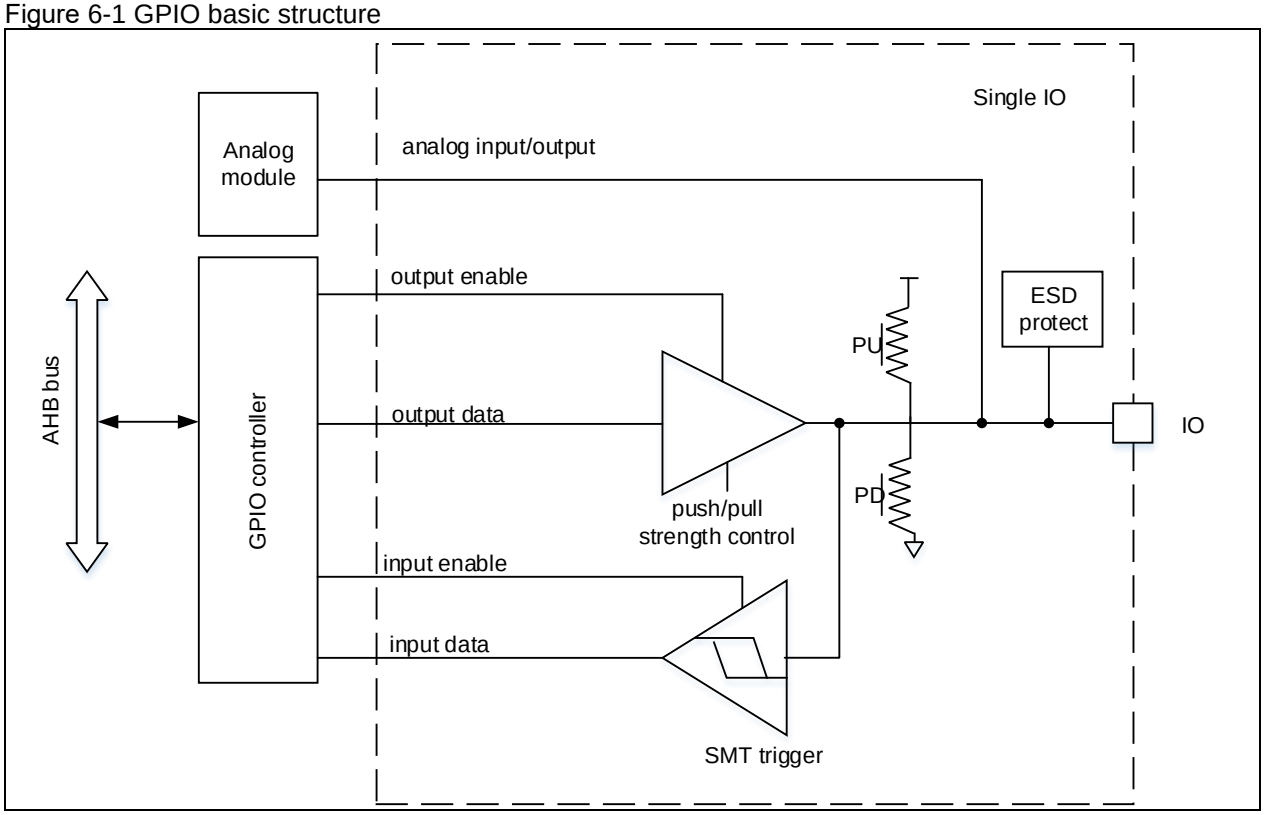
- Each pin can be configured by software as general-purpose input/output, multiplexed function and analog mode. In general-purpose output or multiplexed function mode, it can be configured as pushpull/open-drain output.
- Each pin has individual weak pull-up/pull-down capability.
- Each pin's output drive capability is configurable by software.
- Each pin can be configured as external interrupt input.
- Each pin supports lock mechanism.

6.2 Function overview

6.2.1 GPIO structure

I/O port registers can be accessed by half-words or bytes, and each I/O port bit can be programmed freely.

Figure 6-1 GPIO basic structure



6.2.2 GPIO reset status

After power-on or system reset, all pins are configured as analog mode except CPU debug related pins and BOOT1 pin. These pins are configured as follows:

PA13/SWDIO: multiplexed pull-up

PA14/SWCLK: multiplexed pull-down

PB3/SWO: multiplexed without pull-up/pull-down

PB2/BOOT1: general-purpose floating input mode

6.2.3 General-purpose input configuration

Mode	IOMC	PULL
Floating input	00	00 or 11
Pull-down input		10
Pull-up input		01

When an I/O pin is configured as general-purpose input:

- Get I/O states by reading the input data register.
- Schmitt-trigger input is activated.

Note: In floating input mode, it is recommended to set the unused pins as analog mode in order to avoid leakage caused by intereferences from unused pins in a complex environment

6.2.4 Analog mode configuration

Mode	IOMC	PULL
Analog	11	Unused

When an I/O pin is configured as analog mode:

- Schmitt-trigger input is disabled.
- Output data register is invalid.
- Pull-up/pull-down configuration is invalid.
- I/O states cannot be obtained by reading the input data register.

6.2.5 General-purpose output configuration

Mode	IOMC	OM	HDRV	ODRV[1:0]	PULL
Push-pull without pull-up/pull-down	01	0	000: Output mode, normal sourcing/sinking strength 001: Output mode, large sourcing/sinking strength 010: Output mode, normal sourcing/sinking strength 011: Output mode, normal sourcing/sinking strength 1xx: Output mode, maximum sourcing/sinking strength		00 or 11
Push-pull with pull-up	01	0			01
Push-pull with pull-down	01	0			10
Open-drain without pull-up/pull-down	01	1			00 or 11
Open-drain with pull-up	01	1			01
Open-drain with pull-down	01	1			10

When an I/O port is configured as general-purpose output:

- Schmitt-trigger input is activated.
- In open-drain mode, output 0 when the output data register is configured as 0. When the output data register is configured as 1, the pin is in high-impedance state if there is no internal pullup/pull-down, or the pin is pulled up to 1 or pulled down to 0 if there is internal pull-up or pull-down.

- In push-pull output mode, output data register is used to output 0/1.
- Get I/O states by reading the input data register.
- GPIO toggle/set/clear register is used to toggle/set/clear the corresponding GPIO data output registers.

Note: If both IOCB and IOSB bits are set in the GPIO set/ clear register, the IOSB takes priority.

6.2.6 GPIO port lock mechanism

Each I/O port supports lock mechanism. When lock mechanism is enabled, I/O port configuration cannot be modified until the next reset or power-on.

6.2.7 IOMUX structure

Several peripheral functions can be mapped on each IO pin. Peripheral input/output corresponding to an I/O pin is selected through IOMUX input/output table. Each I/O pin has up to 16 IOMUX mapping options for flexible selection, configured through the GPIOx_MUXL (for pin0 to pin7) and GPIOx_MUXH (for pin8 to pin15) registers.

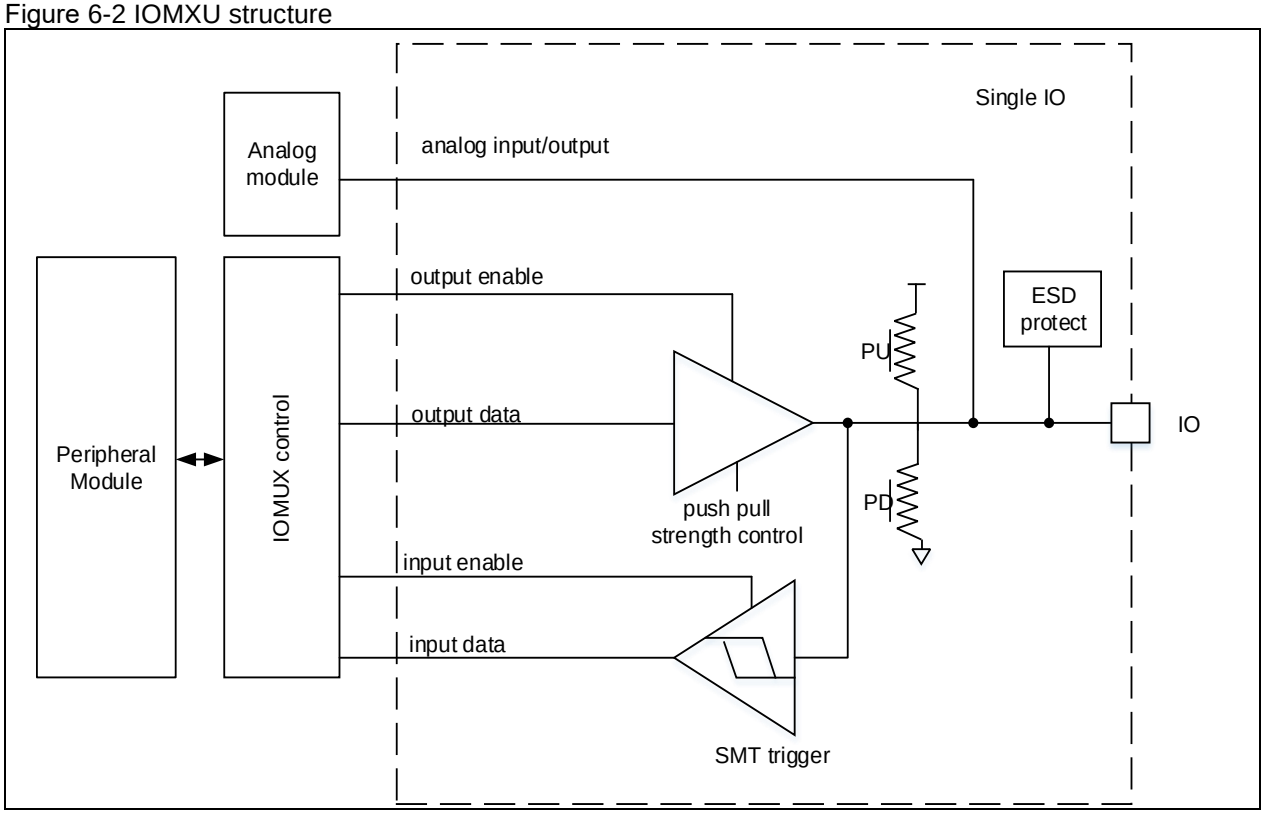
Each I/O pin is connected to only one peripheral's pin by setting the GPIOx_MUXL or GPIOx_MUXH register so that there can be no conflict between peripherals sharing the same pin.

While being used as multiplexed function input, the I/O pin is disconnected from GPIO controller and it is controlled by IOMUX controller.

In multiplexed input mode, the I/O pin can be configured as multiplexed mode by setting the GPIOx_CFGR register, and it can be configured as floating, pull-up/pull-down by setting the GPIOx_PULL register according to the peripheral characteristics.

When used as multiplexed output or bidirectional IOMUX, the I/O pin can be configured as multiplexed mode by setting the GPIOx_CFGR register, as push-pull/open-drain mode by setting the GPIOx_OMODE register or pull-up/pull-down by setting the GPIOx_PULL register according to the peripheral characteristics. Configure output drive capability by setting the GPIOx_ODRVR and GPIOx_HDRV register.

Figure 6-2 IOMXU structure



6.2.8 Multiplexed function configuration

Mode	IOMC	OM	HDRV	ODRV[1:0]	PULL
Push-pull without pull-up/pull-down	10	0	000: output mode, normal sourcing/sinking strength 001: output mode, large sourcing/sinking strength 010: output mode, normal sourcing/sinking strength 011: output mode, normal sourcing/sinking strength 1xx: output mode, maximum sourcing/sinking strength Valid for multiplexed function output or bidirectional multiplexed function only		00 or 11
Push-pull with pull-up	10	0			01
Push-pull with pull-down	10	0			10
Open-drain without pull-up/pull-down	10	1			00 or 11
Open-drain with pull-up	10	1			01
Open-drain with pull-down	10	1			10
Input without pull-up/pull-down	10	x			00 or 11
Input with pull-up	10	x			01
Input with pull-down	10	x			10

When an I/O port is configured as multiplexed function:

- Schmitt-trigger input is activated.
- In open-drain output mode, output 0 when the peripheral outputs 0. When the peripheral outputs 1, the pin is in high-impedance state if there is no internal pull-up/pull-down, or it is pulled up to 1 or pulled down to 0 if there is internal pull-up or pull-down.
- In push-pull output mode, output 0/1 according to the peripheral output.
- Get an I/O pin state by reading input data register.

6.2.9 IOMUX input/output

The multiplexed function of each I/O port line is configured through the GPIOx_MUXL (from pin0 to pin7) or GPIOx_MUXH (from pin8 to pin15) register.

Table 6-1 Port A multiplexed function configuration with GPIOA_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PA0		TMR2_CH1 TMR2_EXT	TMR5_CH1	TMR8_EXT	I2C2_SCL		USART2_RX	USART2_CTS
PA1		TMR2_CH2	TMR5_CH2	TMR9_CH1C	I2C2_SDA	SPI4_MOSI I2S4_SD	SPI3_CS I2S3_WS	USART2_RTS_DE
PA2		TMR2_CH3	TMR5_CH3	TMR9_CH1		I2SF5_CKIN		USART2_TX
PA3		TMR2_CH4	TMR5_CH4	TMR9_CH2		I2S2_MCK		USART2_RX
PA4					I2C1_SCL	SPI1_CS I2S1_WS	SPI3_CS I2S3_WS	USART2_CK
PA5		TMR2_CH1 TMR2_EXT		TMR8_CH1C		SPI1_SCK I2S1_CK	USART3_CK	USART3_RX
PA6		TMR1_BRK	TMR3_CH1	TMR8_BRK		SPI1_MISO	I2S2_MCK	USART3_CTS
PA7		TMR1_CH1C	TMR3_CH2	TMR8_CH1C	I2C3_SCL	SPI1_MOSI I2S1_SD		USART3_TX
PA8	CLKOUT	TMR1_CH1		TMR9_BRK	I2C3_SCL			USART1_CK
PA9	CLKOUT	TMR1_CH2			I2C3_SMBA	SPI2_SCK I2S2_CK		USART1_TX
PA10	ERTC_REFIN	TMR1_CH3				SPI2_MOSI I2S2_SD	I2S4_MCK	USART1_RX
PA11		TMR1_CH4			I2C2_SCL	SPI2_CS I2S2_WS	SPI4_MISO	USART1_CTS
PA12		TMR1_EXT			I2C2_SDA	SPI2_MISO	I2SF5_SDEXT	USART1_RTS_DE

PA13	SWDIO	IR_OUT			I2C1_SDA		SPI3_MISO	
PA14	SWCLK				I2C1_SMBA		SPI3_MOSI I2S3_SD	
PA15		TMR2_CH1 TMR2_EXT				SPI1_CS I2S1_WS	SPI3_CS I2S3_WS	USART1_TX

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PA0	USART4_TX	TMR9_CH2C		EMAC_MII_CRS				EVENTOUT
PA1	USART4_RX	QSPI1_IO3		EMAC_MII_RX_CLK EMAC_RMII_REF_CLK		I2SF5_SD	I2C1_SMBA	EVENTOUT
PA2		CAN2_RX	QSPI1_CS	EMAC_MDIO			XMC_D4	EVENTOUT
PA3		CAN2_TX		EMAC_MII_COL			XMC_D5	EVENTOUT
PA4	USART6_TX	TMR14_CH1				OTGFS1_OE	XMC_D6	EVENTOUT
PA5	USART6_RX	TMR13_CH1C					XMC_D7	EVENTOUT
PA6	USART3_RX	TMR13_CH1	QSPI1_MOSI_IO0		SDIO1_CMD	QSPI1_IO2		EVENTOUT
PA7		TMR14_CH1	QSPI1_MISO_IO1	EMAC_MII_RX_DV EMAC_RMII_CRS_DV	XMC_SDNWE			EVENTOUT
PA8	USART2_TX	USART7_RX	OTGFS1_SOF	CAN3_RX	SDIO1_D1		XMC_A4	EVENTOUT
PA9	I2C1_SCL	TMR14_BRK	OTGFS1_VBUS		SDIO1_D2			EVENTOUT
PA10	I2C1_SDA		OTGFS1_ID			I2SF5_MCK	I2SF5_SD	EVENTOUT
PA11	USART6_TX	CAN1_RX				USART4_RX	I2C1_SMBA	EVENTOUT
PA12	USART6_RX	CAN1_TX				USART4_TX		EVENTOUT
PA13			OTGFS1_OE					EVENTOUT
PA14	USART2_TX							EVENTOUT
PA15	USART2_RX	USART7_TX	QSPI1_IO2	CAN3_TX	XMC_NE2	USART4_RTS_DE		EVENTOUT

Table 6-2 Port B multiplexed function configuration with GPIOB_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PB0		TMR1_CH2C	TMR3_CH3	TMR8_CH2C		SPI1_MISO	USART2_RX	SPI3_MOSI I2S3_SD
PB1		TMR1_CH3C	TMR3_CH4	TMR8_CH3C		SPI1_MOSI I2S1_SD	SPI2_SCK I2S2_CK	USART2_CK
PB2		TMR2_CH4	TMR3_EXT		I2C3_SMBA			SPI3_MOSI I2S3_SD
PB3	SWO	TMR2_CH2			I2C2_SDA	SPI1_SCK I2S1_CK	SPI3_SCK I2S3_CK	USART1_RX
PB4		TMR1_CH4C	TMR3_CH1	TMR11_BRK	I2C3_SDA	SPI1_MISO	SPI3_MISO	I2S3_SDEXT
PB5			TMR3_CH2	TMR10_BRK	I2C1_SMBA	SPI1_MOSI I2S1_SD	SPI3_MOSI I2S3_SD	USART1_CK
PB6			TMR4_CH1	TMR10_CH1C	I2C1_SCL	I2S1_MCK	SPI4_CS I2S4_WS	USART1_TX
PB7			TMR4_CH2	TMR8_BRK	I2C1_SDA		SPI4_SCK I2S4_CK	USART1_RX
PB8		TMR2_CH1 TMR2_EXT	TMR4_CH3	TMR10_CH1	I2C1_SCL		SPI4_MISO	USART1_TX
PB9	IR_OUT	TMR2_CH2	TMR4_CH4	TMR11_CH1	I2C1_SDA	SPI2_CS I2S2_WS	SPI4_MOSI I2S4_SD	I2C2_SDA
PB10		TMR2_CH3			I2C2_SCL	SPI2_SCK I2S2_CK	I2S3_MCK	USART3_TX
PB11		TMR2_CH4	TMR5_CH4		I2C2_SDA	I2SF5_CKIN		USART3_RX
PB12		TMR1_BRK	TMR5_CH1	TMR12_BRK	I2C2_SMBA	SPI2_CS I2S2_WS	SPI4_CS I2S4_WS	SPI3_SCK I2S3_CK
PB13	CLKOUT	TMR1_CH1C		TMR12_CH1C	I2C3_SMBA	SPI2_SCK I2S2_CK	SPI4_SCK I2S4_CK	I2C3_SCL
PB14		TMR1_CH2C		TMR8_CH2C	I2C3_SDA	SPI2_MISO	I2S2_SDEXT	USART3_RTS_DE
PB15	ERTC_REFIN	TMR1_CH3C		TMR8_CH3C	I2C3_SCL	SPI2_MOSI I2S2_SD		

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PB0	USART3_CK		QSPI1_MOSI_IO0	EMAC_MII_RXD2	SDIO1_D1	I2SF5_CK		EVENTOUT
PB1	USART3_RTS_DE	QSPI1_SCK		EMAC_MII_RXD3	SDIO1_D2	I2SF5_WS	TMR14_CH1	EVENTOUT
PB2		QSPI1_SCK	CAN3_STB		SDIO1_CK		TMR14_CH1C	EVENTOUT
PB3	USART1_RTS_DE	USART7_RX	QSPI1_IO3	CAN3_RX		USART5_TX		EVENTOUT
PB4	USART1_CTS	USART7_TX	QSPI1_SCK	CAN3_TX	SDIO1_D0	USART5_RX		EVENTOUT
PB5	USART5_RX	CAN2_RX	QSPI1_MOSI_IO0	EMAC_PPS_OUT	XMC_SDCKE1	USART5_CK_ RTS_DE	SDIO1_D3	EVENTOUT
PB6	USART5_TX	CAN2_TX	QSPI1_CS	USART4_CK	XMC_SDNE1	I2SF5_WS	SDIO1_D0	EVENTOUT
PB7	USART4_CTS	TMR11_CH1C	QSPI1_MISO_IO1	CAN1_STB	XMC_NADV	I2SF5_CK	SDIO1_D0	EVENTOUT
PB8	USART5_RX	CAN1_RX		EMAC_MII_TXD3	SDIO1_D4	I2SF5_SDEXT	I2SF5_SD	EVENTOUT
PB9	USART5_TX	CAN1_TX	QSPI1_CS		SDIO1_D5	I2SF5_SD	I2S1_MCK	EVENTOUT
PB10		QSPI1_CS	QSPI1_MISO_IO1	EMAC_MII_RX_ER	SDIO1_D7		XMC_NOE	EVENTOUT
PB11		TMR13_BRK	QSPI1_MOSI_IO0	EMAC_MII_TX_EN EMAC_RMII_TX_EN		CAN2_STB	XMC_LB	EVENTOUT
PB12	USART3_CK	CAN2_RX		EMAC_MII_TXD0 EMAC_RMII_TXD0	USART5_RX	I2SF5_WS	XMC_D13	EVENTOUT
PB13	USART3_CTS	CAN2_TX		EMAC_MII_TXD1 EMAC_RMII_TXD1	USART5_TX	I2SF5_CK	XMC_UB	EVENTOUT
PB14		TMR12_CH1				SDIO1_D6	XMC_D0	EVENTOUT
PB15		TMR12_CH2				SDIO1_CK	TMR12_CH1C	EVENTOUT

Table 6-3 Port C multiplexed function configuration with GPIOC_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PC0					I2C3_SCL		I2C1_SCL	
PC1					I2C3_SDA	SPI3_MOSI I2S3_SD	I2C1_SDA	SPI2_MOSI I2S2_SD
PC2						SPI2_MISO	I2S2_SDEXT	
PC3						SPI2_MOSI I2S2_SD		
PC4				TMR9_CH1		I2S1_MCK		USART3_TX
PC5		TMR1_CH4C		TMR9_CH2	I2C1_SMBA			USART3_RX
PC6		TMR1_CH1	TMR3_CH1	TMR8_CH1	I2C1_SCL	I2S2_MCK		
PC7		TMR1_CH2	TMR3_CH2	TMR8_CH2	I2C1_SDA	SPI2_SCK I2S2_CK	I2S3_MCK	
PC8		TMR1_CH3	TMR3_CH3	TMR8_CH3		I2S4_MCK	I2SF5_MCK	USART8_TX
PC9	CLKOUT	TMR1_CH4	TMR3_CH4	TMR8_CH4	I2C3_SDA	I2SF5_CKIN		USART8_RX
PC10			TMR5_CH2				SPI3_SCK I2S3_CK	USART3_TX
PC11			TMR5_CH3			I2S3_SDEXT	SPI3_MISO	USART3_RX
PC12				TMR11_CH1	I2C2_SDA		SPI3_MOSI I2S3_SD	USART3_CK
PC13				TMR8_CH4C				
PC14								
PC15								

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PC0	USART6_TX	USART7_TX			XMC_SDNWE			EVENTOUT
PC1	USART6_RX	USART7_RX		EMAC_MDC				EVENTOUT
PC2	USART8_TX			EMAC_MII_TXD2	XMC_SDNE0		XMC_NWE	EVENTOUT
PC3	USART8_RX			EMAC_MII_TX_CLK	XMC_SDCKE0		XMC_A0	EVENTOUT
PC4		TMR13_CH1	QSPI1_IO2	EMAC_MII_RXD0 EMAC_RMII_RXD0	XMC_SDNE0		XMC_NE4	EVENTOUT
PC5		TMR13_CH1C	QSPI1_IO3	EMAC_MII_RXD1 EMAC_RMII_RXD1	XMC_SDCKE0		XMC_NOE	EVENTOUT
PC6	USART6_TX	USART7_TX	XMC_A0		SDIO1_D6		XMC_D1	EVENTOUT
PC7	USART6_RX	USART7_RX	XMC_A1		SDIO1_D7		XMC_NADV	EVENTOUT
PC8	USART6_CK	QSPI1_IO2	XMC_A2		SDIO1_D0			EVENTOUT
PC9	I2C1_SDA	QSPI1_MOSI_IO0	XMC_A3	OTGFS1_OE	SDIO1_D1			EVENTOUT
PC10	USART4_TX	QSPI1_MISO_IO1			SDIO1_D2			EVENTOUT
PC11	USART4_RX	QSPI1_CS			SDIO1_D3		XMC_D2	EVENTOUT
PC12	USART4_CK	USART5_TX			SDIO1_CK		XMC_D3	EVENTOUT
PC13								EVENTOUT
PC14								EVENTOUT
PC15								EVENTOUT

Table 6-4 Port D multiplexed function configuration with GPIOD_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PD0				TMR8_CH4C		SPI4_MISO	SPI3_MOSI I2S3_SD	SPI2_CS I2S2_WS
PD1				TMR8_CH4			SPI2_SCK I2S2_CK	SPI2_CS I2S2_WS
PD2			TMR3_EXT					USART3_RTS_DE
PD3						SPI2_SCK I2S2_CK	SPI2_MISO	USART2_CTS
PD4							SPI2_MOSI I2S2_SD	USART2_RTS_DE
PD5								USART2_TX
PD6						SPI3_MOSI I2S3_SD		USART2_RX
PD7								USART2_CK
PD8								USART3_TX
PD9								USART3_RX
PD10								USART3_CK
PD11					I2C2_SMBA			USART3_CTS
PD12			TMR4_CH1		I2C2_SCL			USART3_RTS_DE
PD13			TMR4_CH2		I2C2_SDA			
PD14			TMR4_CH3		I2C3_SCL			
PD15			TMR4_CH4		I2C3_SDA			

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PD0	USART4_RX	CAN1_RX	XMC_A5		XMC_D2			EVENTOUT
PD1	USART4_TX	CAN1_TX	XMC_A6		XMC_D3			EVENTOUT
PD2	USART5_RX		XMC_A7		SDIO1_CMD		XMC_NWE	EVENTOUT
PD3		QSPI1_SCK	XMC_A8		XMC_CLK			EVENTOUT
PD4			XMC_A9		XMC_NOE			EVENTOUT
PD5			XMC_A10		XMC_NWE			EVENTOUT
PD6			XMC_A11		XMC_NWAIT			EVENTOUT
PD7			XMC_A12		XMC_NE1			EVENTOUT
PD8		TMR12_CH2C		EMAC_MII_RX_DV EMAC_RMII_CRS_DV	XMC_D13			EVENTOUT
PD9				EMAC_MII_RXD0 EMAC_RMII_RXD0	XMC_D14			EVENTOUT
PD10	USART4_TX			EMAC_MII_RXD1 EMAC_RMII_RXD1	XMC_D15			EVENTOUT
PD11		QSPI1_MOSI_IO0	XMC_A14 XMC_SDBA0	EMAC_MII_RXD2	XMC_A16	CAN3_STB		EVENTOUT
PD12	USART8_CK_RTS_DE	QSPI1_MISO_IO1	XMC_A15 XMC_SDBA1	EMAC_MII_RXD3	XMC_A17	CAN3_RX		EVENTOUT
PD13	USART8_TX	QSPI1_IO3	XMC_SDCLK		XMC_A18	CAN3_TX		EVENTOUT
PD14	USART8_RX				XMC_D0			EVENTOUT
PD15		USART7_CK_RTS _DE			XMC_D1			EVENTOUT

Table 6-5 Port E multiplexed function configuration with GPIOE_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PE0			TMR4_EXT					
PE1		TMR1_CH2C						
PE2			TMR3_EXT	TMR9_BRK		SPI4_SCK I2S4_CK		
PE3			TMR3_CH1	TMR9_CH2C				
PE4	CLKOUT		TMR3_CH2	TMR9_CH1C		SPI4_CS I2S4_WS		
PE5			TMR3_CH3	TMR9_CH1		SPI4_MISO		
PE6			TMR3_CH4	TMR9_CH2		SPI4_MOSI I2S4_SD		
PE7		TMR1_EXT						
PE8		TMR1_CH1C						
PE9		TMR1_CH1						
PE10		TMR1_CH2C						
PE11		TMR1_CH2				SPI4_CS I2S4_WS		
PE12		TMR1_CH3C			SPI1_CS I2S1_WS	SPI4_SCK I2S4_CK		
PE13		TMR1_CH3			SPI1_SCK I2S1_CK	SPI4_MISO		
PE14		TMR1_CH4			SPI1_MISO	SPI4_MOSI I2S4_SD		
PE15		TMR1_BRK	TMR1_CH4C		SPI1_MOSI I2S1_SD			

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PE0	USART8_RX	TMR13_CH1			XMC_LB XMC_SDDQML			EVENTOUT
PE1	USART8_TX	TMR14_CH1			XMC_UB XMC_SDDQMH			EVENTOUT
PE2		QSPI1_IO2	XMC_SDNCAS	EMAC_MII_TXD3	XMC_A23	TMR14_CH1C		EVENTOUT
PE3		TMR14_BRK			XMC_A19			EVENTOUT
PE4					XMC_A20			EVENTOUT
PE5					XMC_A21			EVENTOUT
PE6			XMC_SDNRAS		XMC_A22			EVENTOUT
PE7	USART5_CK_ RTS _DE	USART7_RX	XMC_A13		XMC_D4			EVENTOUT
PE8	USART4_TX	USART7_TX	XMC_A16		XMC_D5			EVENTOUT
PE9	USART4_RX		XMC_A17		XMC_D6			EVENTOUT
PE10	USART5_TX		XMC_A18		XMC_D7			EVENTOUT
PE11	USART5_RX				XMC_D8			EVENTOUT
PE12					XMC_D9			EVENTOUT
PE13					XMC_D10			EVENTOUT
PE14					XMC_D11			EVENTOUT
PE15					XMC_D12			EVENTOUT

Table 6-6 Port F multiplexed function configuration with GPIOF_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PF0					I2C2_SDA			
PF1					I2C2_SCL			
PF2					I2C2_SMBA			
PF3								
PF4								
PF5								
PF6				TMR10_CH1				
PF7				TMR11_CH1				
PF8								
PF9								
PF10		TMR1_EXT	TMR5_CH4					
PF11				TMR8_EXT				
PF12				TMR8_BRK				
PF13					I2C3_SMBA			
PF14					I2C3_SCL			
PF15					I2C3_SDA			

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PF0					XMC_A0			EVENTOUT
PF1					XMC_A1			EVENTOUT
PF2					XMC_A2			EVENTOUT
PF3					XMC_A3			EVENTOUT
PF4					XMC_A4			EVENTOUT
PF5			CAN3_STB		XMC_A5			EVENTOUT
PF6	USART7_RX	QSPI1_IO3		CAN3_RX				EVENTOUT
PF7	USART7_TX	QSPI1_IO2		CAN3_TX				EVENTOUT
PF8	USART8_RX	TMR13_CH1	QSPI1_MOSI_IO0					EVENTOUT
PF9	USART8_TX	TMR14_CH1	QSPI1_MISO_IO1					EVENTOUT
PF10		QSPI1_SCK						EVENTOUT
PF11					XMC_SDNRAS			EVENTOUT
PF12					XMC_A6			EVENTOUT
PF13					XMC_A7			EVENTOUT
PF14					XMC_A8			EVENTOUT
PF15			CAN1_STB		XMC_A9			EVENTOUT

Table 6-7 Port G multiplexed function configuration with GPIOG_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PG0						SPI1_MISO		
PG1						SPI1_MOSI I2S1_SD		
PG2								
PG3								
PG4								
PG5								
PG6								
PG7								
PG8								
PG9								
PG10								
PG11							SPI4_SCK I2S4_CK	
PG12							SPI4_MISO	
PG13							SPI4_MOSI I2S4_SD	
PG14							SPI4_CS I2S4_WS	
PG15								

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PG0		CAN1_RX			XMC_A10			EVENTOUT
PG1		CAN1_TX			XMC_A11			EVENTOUT
PG2					XMC_A12			EVENTOUT
PG3					XMC_A13			EVENTOUT
PG4					XMC_A14 XMC_SDBA0			EVENTOUT
PG5					XMC_A15 XMC_SDBA1			EVENTOUT
PG6			QSPI1_CS					EVENTOUT
PG7	USART6_CK							EVENTOUT
PG8	USART6_RTS_DE			EMAC_PPS_OUT	XMC_SDCLK			EVENTOUT
PG9	USART6_RX	QSPI1_IO2		CAN3_TX	XMC_NE2			EVENTOUT
PG10				CAN3_RX	XMC_NE3			EVENTOUT
PG11		CAN2_RX		EMAC_MII_TX_EN EMAC_RMII_TX_EN				EVENTOUT
PG12	USART6_RTS_DE	CAN2_TX			XMC_NE4			EVENTOUT
PG13	USART6_CTS		CAN2_STB	EMAC_MII_TXD0 EMAC_RMII_TXD0	XMC_A24			EVENTOUT
PG14	USART6_TX	QSPI1_IO3		EMAC_MII_TXD1 EMAC_RMII_TXD1	XMC_A25			EVENTOUT
PG15	USART6_CTS				XMC_SDNCAS			EVENTOUT

Table 6-8 Port H multiplexed function configuration with GPIOH_MUX* register

Pin name	MUX0	MUX1	MUX2	MUX3	MUX4	MUX5	MUX6	MUX7
PH0		TMR1_CH1			I2C1_SDA			
PH1		TMR1_CH2C			I2C1_SCL	SPI2_CS I2S2_WS		
PH2		TMR2_CH1	TMR5_CH1		I2C2_SCL			
PH3		TMR2_CH2	TMR5_CH2		I2C2_SDA			
PH4						SPI2_SCK I2S2_CK		

Pin name	MUX8	MUX9	MUX10	MUX11	MUX12	MUX13	MUX14	MUX15
PH0								EVENTOUT
PH1								EVENTOUT
PH2	USART4_RX	USART7_RX	QSPI1_MOSI_IO0					EVENTOUT
PH3	USART4_TX	USART7_TX	QSPI1_MISO_IO1					EVENTOUT
PH4		USART7_CK_ RTS _DE						EVENTOUT

Note: EVENTOUT is the EVENTOUT signal of Cortex-M .

6.2.10 Peripheral MUX function configuration

IOMUX function configuration is as follows:

- To use a peripheral pin in MUX output, it is configured as multiplexed push-pull/open-drain output.
- To use a peripheral pin in MUX input, it is configured as floating input/pull-up/pull-down input.
- For ADC peripherals, the pins of analog channels should be configured as analog input/output mode.
- For I^2C peripherals that intend to use pins as bidirectional functions, open-drain mode is required.
- For the USB OTGFS1_ID pin, configure the corresponding IOMUX function and enable the corresponding CRM clock, without the need to configure GPIO states.

6.2.11 IOMUX mapping priority

The unique peripheral multiplexed function can be configured through the GPIOx_MUXL/GPIOx_MUXH register, except individual pins that may be directly owned by hardware.

Some pins have been directly owned by specific hardware feature, whatever GPIO configuration.

Table 6-9 Pins owned by hardware

Pin	Enable bit	Description
PA0	PWC_CTRL[16] = 1	Once enabled, PA0 pin acts as WKUP1 of PWC.
PC13	PWC_CTRL[17] = 1	Once enabled, PC13 pin acts as WKUP2 of PWC.
PA2	PWC_CTRL[19] = 1	Once enabled, PA2 pin acts as WKUP4 of PWC.
PC5	PWC_CTRL[20] = 1	Once enabled, PC5 pin acts as WKUP5 of PWC.
PB5	PWC_CTRL[21] = 1	Once enabled, PB5 pin acts as WKUP6 of PWC.
PB15	PWC_CTRL[22] = 1	Once enabled, PB15 pin acts as WKUP7 of PWC.
PC13	(ERTC_CTRL[23]=1)| (ERTC_CTRL[22:21]!=00)| (ERTC_CTRL[11]=1& ERTC_TAMP[17]=0)| (ERTC_TAMP[0]=1& ERTC_TAMP[16]=0)	Once enabled, PC13 pin acts as RTC channel.
PA0	(ERTC_CTRL[11]=1& ERTC_TAMP[17]=1)| (ERTC_TAMP[0]=1& ERTC_TAMP[16]=1)| (ERTC_TAMP[3]=1)	Once enabled, PA0 pin acts as TAMPER2_BPR.
PC14	CRM_BPDC[0]=1	Once enabled, PC14 pin acts as LEXT channel.
PC15	CRM_BPDC[0]=1 & CRM_BPDC[2]=0	Once enabled, PC15 pin acts as LEXT channel.
PA4	DAC_CTRL[2] =1	Once enabled, PA4 pin acts as DAC1 analog channel.
PA5	DAC_CTRL[18] =1	Once enabled, PA5 pin acts as DAC2 analog channel.
PH0	CRM_CTRL[16]=1	Once enabled, PH0 pin acts as HEXT channel.
PH1	CRM_CTRL[16]=1& CRM_CTRL[18]=0	Once enabled, PH1 pin acts as HEXT channel.
PA10	CRM_AHBEN2[7]=1 & GPIOA_CFGR[21:20]=10 & GPIOA_MUXH[11:8]=1010	Once enabled, PA10 pin is configured as multiplexed input pull-up mode automatically for OTGFS1_ID.
PA11	CRM_AHBEN2[7] & OTGFS_GCCFG[16]	Once enabled, PA11 pin acts as OTGFS_D- channel.
PA12	CRM_AHBEN2[7] & OTGFS_GCCFG[16]	Once enabled, PA12 pin acts as OTGFS_D+ channel.

Note: PA0 or PC13 cannot be used as TAMPER_BPR function or WKUP of PWC simultaneously.

6.2.12 External interrupt/wake-up lines

Each pin can be used as an external interrupt input. The corresponding pin should be configured as input mode.

6.3 GPIO registers

The table below lists GPIO register map and their reset values. These peripheral registers can be accessed by bytes (8 bits), half-words (16 bits) or words (32 bits).

Table 6-10 GPIO register map and reset values

Register	Offset	Reset value
GPIOA_CFGR	0x00	0xEBFF FFFF
GPIOx_CFGR(x =B,C,D,E,F,G,H)	0x00	0xFFFF FF8F(B) 0xFFFF FFFF
GPIOx_OMODER	0x04	0x0000 0000
GPIOx_ODRVR	0x08	0x0C00 0000(A) 0x0000 00C0(B) 0x0000 0000
GPIOA_PULL	0x0C	0x2400 0000(A)
GPIOx_PULL(x = B,C,D,E,F,G,H)	0x0C	0x0000 0000
GPIOx_IDT	0x10	0x0000 XXXX
GPIOx_ODT	0x14	0x0000 0000
GPIOx_SCR	0x18	0x0000 0000
GPIOx_WPR	0x1C	0x0000 0000
GPIOx_MUXL	0x20	0x0000 0000
GPIOx_MUXH	0x24	0x0000 0000
GPIOx_CLR	0x28	0x0000 0000
GPIOx_TOGR	0x2C	0x0000 0000
GPIOx_HDRV	0x3C	0x0000 0000

6.3.1 GPIO configuration register (GPIOx_CFGR) (x=A..H)

Address offset: 0x00

Reset value: 0xEBFF_FFFF for port A, 0xFFFF_FF8F for port B, and 0xFFFF_FFFF for other ports.

Bit	Name	Reset value	Type	Description
Bit 2y+1:2y	IOMCy	0xEBFF FFFF	rw	GPIOx mode configuration (y=0~15) This field is used to configure the GPIOx mode: 00: Input mode 01: General-purpose output 10: Multiplexed function mode 11: Analog mode (after reset)

6.3.2 GPIO output mode register (GPIOx_OMODE) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Always 0.
Bit 15:0	OM	0x0000	rw	GPIOx output mode configuration (y=0..15) This field is used to configure the output mode of GPIOx: 0: Push-pull (reset state) 1: Open-drain

6.3.3 GPIO drive capability register (GPIOx_ODRVR) (x=A..H)

Address offset: 0x08

Reset value: 0x0C00_0000 for port A, 0x0000_00C0 for port B, and 0x0000_0000 for other ports.

Bit	Name	Reset value	Type	Description
Bit 2y+1:2y	ODRVy	0x0000 0000	rw	GPIOx drive capability (y=0…15) This field is used to configure the I/O port drive capability. x0: Normal sourcing/sinking strength 01: Large sourcing/sinking strength 11: Normal sourcing/sinking strength

6.3.4 GPIO pull-up/pull-down register (GPIOx_PULL) (x=A..H)

Address offset: 0x0C

Reset value: 0x2400_0000 for port A, and 0x0000_0000 for other ports.

Bit	Name	Reset value	Type	Description
Bit 2y+1:2y	PULLy	0x2400 0000	rw	GPIOx pull-up/pull-down configuration (y=0…15) This field is used to configure the pull-up/pull-down of the I/O port. 00,11: No pull-up/pull-down 01: Pull-up 10: Pull-down

6.3.5 GPIO input data register (GPIOx_IDT) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Always 0.
Bit 15:0	IDT	0xXXXX	ro	GPIOx input data It indicates the input status of I/O port. Each bit corresponds to an I/O.

6.3.6 GPIO output data register (GPIOx_ODT) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Always 0.
Bit 15:0	ODT	0x0000	rw	GPIOx output data Each bit represents an I/O port. It indicates the output status of I/O port. 0: Low 1: High

6.3.7 GPIO set/clear register (GPIOx_SCR) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	IOCB	0x0000	wo	GPIOx clear bit The corresponding ODT register bit is cleared by writing “1” to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations. 0: No action to the corresponding ODT bits 1: Clear the corresponding ODT bits
Bit 15:0	IOSB	0x0000	wo	GPIOx set bit The corresponding ODT register bit is set by writing “1” to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations. If both IOCB and IOSB bits are set to 1, the IOSB takes the priority. 0: No action to the corresponding ODT bits 1: Set the corresponding ODT bits

6.3.8 GPIO write protection register (GPIOx_WPR) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:17	Reserved	0x0000	resd	Kept as its default value.
Bit 16	WPSEQ	0x0	rw	Write protect sequence Write protect enable sequence bit and WPEN bit must be enabled at the same time to achieve write protection for some I/O bits. Write protect enable bit is executed four times in the order below: write “1” -> write “0” -> write “1” -> read. Note that the value of WPEN bit cannot be modified during this period.
Bit 15:0	WPEN	0x0000	rw	Write protect enable Each bit corresponds to an I/O port 0: No effect 1: Write protection

6.3.9 GPIO multiplexed function low register (GPIOx_MUXL) (x=A..H)

Address offset: 0x20

Reset value: 0x00000000

Bit	Name	Reset value	Type	Description
Bit 4y+3:4y	MUXLy	0x0	rw	Multiplexed function select for GPIOx pin y (y=0…7) This field is used to configure multiplexed function I/Os. 0000: MUX0 0001: MUX1 0010: MUX2 0011: MUX3 0100: MUX4 0101: MUX5 0110: MUX6 0111: MUX7 1000: MUX8 1001: MUX9

				1010: MUX10 1011: MUX11 1100: MUX12 1101: MUX13 1110: MUX14 1111: MUX15

6.3.10 GPIO multiplexed function high register (GPIOx_MUXH) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 4(y-8)+3 :4(y-8)	MUXHy	0x0	rw	Multiplexed function select for GPIOx pin y (y=8…15) This field is used to configure multiplexed function I/Os. 0000: MUX0 0001: MUX1 0010: MUX2 0011: MUX3 0100: MUX4 0101: MUX5 0110: MUX6 0111: MUX7 1000: MUX8 1001: MUX9 1010: MUX10 1011: MUX11 1100: MUX12 1101: MUX13 1110: MUX14 1111: MUX15

6.3.11 GPIO port bit clear register (GPIOx_CLR) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept as its default value.
Bit 15:0	IOCB	0x0000	wo	GPIOx clear bit The corresponding ODT register bit is cleared by writing “1” to these bits. Otherwise, the corresponding ODT register bit remains unchanged, which acts as ODT register bit operations. 0: No action to the corresponding ODT bits 1: Clear the corresponding ODT bits

6.3.12 GPIO bit port toggle register (GPIOx_TOGR) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept as its default value.
Bit 15:0	IOTB	0x0000	wo	GPIOx pin y (y=0…15) toggle These bit are write-only. Reading them returns 0x0000. 0: No action to the corresponding ODTy bits 1: Toggle the corresponding ODTy bits

6.3.13 GPIO huge current control register (GPIOx_HDRV) (x=A..H)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept as its default value.
Bit 15:0	HDRV	0x0000	rw	Huge sourcing/sinking strength control 0: Not active 1: GPIO is configured as maximum sourcing/sinking strength

7 System configuration controller (SCFG)

7.1 Introduction

This device contains a set of system configuration registers. The system configuration controller is mainly set to:

- Manage the external interrupts connected to GPIOs
- Control the memory mapping mode
- Manage IRTMR GPIO configurations

7.2 SCFG registers

The table below shows SCFG register map and their reset values.

These peripheral registers must be accessed by words (32 bits).

Table 7-1 SCFG register map and reset value

Register	Offset	Reset value
SCFG_CFG1	0x00	0x0000 000X
SCFG_CFG2	0x04	0x0000 0000
SCFG_EXINTC1	0x08	0x0000 0000
SCFG_EXINTC2	0x0C	0x0000 0000
SCFG_EXINTC3	0x10	0x0000 0000
SCFG_EXINTC4	0x14	0x0000 0000
SCFG_UHDRV	0x2C	0x0000 0000

7.2.1 SCFG configuration register 1 (SCFG_CFG1)

Bit	Name	Reset value	Type	Description
Bit 31:13	Reserved	0x00000 0	resd	Kept at its reset value.
Bit 12	SWAP_QSPI	0x0	rw	QSPI address remapping 0: No QSPI address remap 1: QSPI address is remapped to 0x0900 0000 - 0x18FF FFFF. The remap address cannot be read before QSPI initialization, which may result in debugging error report when compiling code to the remap address. It is recommended to complete debugging before adding and re-compiling the remap code.
Bit 11	Reserved	0x0	resd	Kept at its reset value.
Bit 10	SWAP_XMC	0x0	rw	XMC address mapping swap 0: No XMC address mapping swap 1: SDRAM Bank0 address swaps to 0x6000 0000 -

				0x6FFF FFFF, and the NOR/ PSRAM /SRAM address swaps to 0xC000 00000 - 0xCFFF FFFF.
Bit 9:8	Reserved	0x0	resd	Kept at its reset value.
Bit 7:6	IR_SRC_SEL	0x0	rw	Infrared modulation envelope signal source selection This field is used to select the infrared modulation envelope signal source. 00: TMR10 01: Reserved 10: Reserved 11: Reserved
Bit 5	IR_POL	0x0	rw	Infrared output polarity selection 0: Infrared output (IR_OUT) is not inversed 1: Infrared output (IR_OUT) is inversed
Bit 4:2	Reserved	0x0	resd	Kept at its reset value.
Bit 1:0	MEM_MAP_SEL	0xX	ro	Boot mode status bit This bit is read-only, indicating the boot mode after reset. X0: Boot from main Flash memory 01: Boot from boot memory 11: Boot from embedded SRAM

7.2.2 SCFG configuration register 2 (SCFG_CFG2)

Bit	Name	Reset value	Type	Description
Bit 31:27	Reserved	0x0000 000	resd	Kept at its reset value.
Bit 26	CAN3_TST_SEL	0x0	rw	CAN3 timestamp counter source select 0: TMR3 1: TMR5
Bit 25	CAN2_TST_SEL	0x0	rw	CAN2 timestamp counter source select 0: TMR3 1: TMR5
Bit 24	CAN1_TST_SEL	0x0	rw	CAN1 timestamp counter source select 0: TMR3 1: TMR5
Bit 23	MII_RMII_SEL	0x0	rw	MII or RMII select This bit is used to select MII or RMII interface for Ethernet. 0: MII 1: RMII Note: This bit is valid for AT32F457 only.
Bit 22:9	Reserved	0x0000 000	resd	Kept at its reset value.
Bit 8	SRAM_OPERR_STS	0	rc_w1	SRAM odd parity error status This bit is cleared by writing “1”. 0: No SRAM odd parity error 1: SRAM odd parity error
Bit 7:3	Reserved	0x0000 000	resd	Kept at its reset value.
Bit 2	PVM_LK	0x0	rw	PVM lock enable 0: Disconnect the PVM interrupt with TMR1/TMR9

				/TMR10/11/12/13/14 brake input. The PVMSEL and PVMEN bits can be modified by software. 1: Connect the PVM interrupt with TMR1/TMR9 /TMR10/11/12/13/14 brake input. The PVMSEL and PVMEN bits are read-only and cannot be modified by software.
Bit 1	SRAM_OPERR_LK	0	rw	SRAM odd parity error lock enable 0: Disconnect SRAM odd parity error with TMR1/TMR9/10/11/13/14 brake input. 1: Connect SRAM odd parity error with TMR1/TMR9/10/11/13/14 brake input.
Bit 0	LOCKUP_LK	0x0	rw	CM4F LOCKUP bit enable

7.2.3 SCFG external interrupt configuration register 1 (SCFG_EXINTC1)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15:12	EXINT3	0x0	rw	EXINT3 input source configuration These bits are used to select the input source for the EXINT3 external interrupt. 0000: GPIOA pin 3 0001: GPIOB pin 3 0010: GPIOC pin 3 0011: GPIOD pin 3 0100: GPIOE pin 3 0101: GPIOF pin 3 0110: GPIOG pin 3 0111: GPIOH pin 3 Others: Reserved
Bit 11:8	EXINT2	0x0	rw	EXINT2 input source configuration These bits are used to select the input source for the EXINT2 external interrupt. 0000: GPIOA pin 2 0001: GPIOB pin 2 0010: GPIOC pin 2 0011: GPIOD pin 2 0100: GPIOE pin 2 0101: GPIOF pin 2 0110: GPIOG pin 2 0111: GPIOH pin 2 Others: Reserved
Bit 7:4	EXINT1	0x0	rw	EXINT1 input source configuration These bits are used to select the input source for the EXINT1 external interrupt. 0000: GPIOA pin 1 0001: GPIOB pin 1 0010: GPIOC pin 1 0011: GPIOD pin 1 0100: GPIOE pin 1

				0101: GPIOF pin 1 0110: GPIOG pin 1 0111: GPIOH pin 1 Others: Reserved
Bit 3:0	EXINT0	0x0	rw	EXINT0 input source configuration These bits are used to select the input source for the EXINT0 external interrupt. 0000: GPIOA pin 0 0001: GPIOB pin 0 0010: GPIOC pin 0 0011: GPIOD pin 0 0100: GPIOE pin 0 0101: GPIOF pin 0 0110: GPIOG pin 0 0111: GPIOH pin 0 Others: Reserved

7.2.4 SCFG external interrupt configuration register 2 (SCFG_EXINTC2)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15:12	EXINT7	0x0	rw	EXINT7 input source configuration These bits are used to select the input source for the EXINT7 external interrupt. 0000: GPIOA pin 7 0001: GPIOB pin 7 0010: GPIOC pin 7 0011: GPIOD pin 7 0100: GPIOE pin 7 0101: GPIOF pin 7 0110: GPIOG pin 7 Others: Reserved
Bit 11:8	EXINT6	0x0	rw	EXINT6 input source configuration These bits are used to select the input source for the EXINT6 external interrupt. 0000: GPIOA pin 6 0001: GPIOB pin 6 0010: GPIOC pin 6 0011: GPIOD pin 6 0100: GPIOE pin 6 0101: GPIOF pin 6 0110: GPIOG pin 6 Others: Reserved
Bit 7:4	EXINT5	0x0	rw	EXINT5 input source configuration These bits are used to select the input source for the EXINT5 external interrupt. 0000: GPIOA pin 5

				0001: GPIOB pin 5 0010: GPIOC pin 5 0011: GPIOD pin 5 0100: GPIOE pin 5 0101: GPIOF pin 5 0110: GPIOG pin 5 Others: Reserved
Bit 3:0	EXINT4	0x0	rw	EXINT4 input source configuration These bits are used to select the input source for the EXINT4 external interrupt. 0000: GPIOA pin 4 0001: GPIOB pin 4 0010: GPIOC pin 4 0011: GPIOD pin 4 0100: GPIOE pin 4 0101: GPIOF pin 4 0110: GPIOG pin 4 0111: GPIOH pin 4 Others: Reserved

7.2.5 SCFG external interrupt configuration register 3 (SCFG_EXINTC3)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15:12	EXINT11	0x0	rw	EXINT11 input source configuration These bits are used to select the input source for the EXINT11 external interrupt. 0000: GPIOA pin 11 0001: GPIOB pin 11 0010: GPIOC pin 11 0011: GPIOD pin 11 0100: GPIOE pin 11 0101: GPIOF pin 11 0110: GPIOG pin 11 Others: Reserved
Bit 11:8	EXINT10	0x0	rw	EXINT10 input source configuration These bits are used to select the input source for the EXINT10 external interrupt. 0000: GPIOA pin 10 0001: GPIOB pin 10 0010: GPIOC pin 10 0011: GPIOD pin 10 0100: GPIOE pin 10 0101: GPIOF pin 10 0110: GPIOG pin 10 Others: Reserved
Bit 7:4	EXINT9	0x0	rw	EXINT9 input source configuration

				These bits are used to select the input source for the EXINT9 external interrupt. 0000: GPIOA pin 9 0001: GPIOB pin 9 0010: GPIOC pin 9 0011: GPIOD pin 9 0100: GPIOE pin 9 0101: GPIOF pin 9 0110: GPIOG pin 9 Others: Reserved
Bit 3:0	EXINT8	0x0	rw	EXINT8 input source configuration These bits are used to select the input source for the EXINT8 external interrupt. 0000: GPIOA pin 8 0001: GPIOB pin 8 0010: GPIOC pin 8 0011: GPIOD pin 8 0100: GPIOE pin 8 0101: GPIOF pin 8 0110: GPIOG pin 8 Others: Reserved

7.2.6 SCFG external interrupt configuration register 4 (SCFG_EXINTC4)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15:12	EXINT15	0x0	rw	EXINT15 input source configuration These bits are used to select the input source for the EXINT15 external interrupt. 0000: GPIOA pin 15 0001: GPIOB pin 15 0010: GPIOC pin 15 0011: GPIOD pin 15 0100: GPIOE pin 15 0101: GPIOF pin 15 0110: GPIOG pin 15 Others: Reserved
Bit 11:8	EXINT14	0x0	rw	EXINT14 input source configuration These bits are used to select the input source for the EXINT14 external interrupt. 0000: GPIOA pin 14 0001: GPIOB pin 14 0010: GPIOC pin 14 0011: GPIOD pin 14 0100: GPIOE pin 14 0101: GPIOF pin 14 0110: GPIOG pin 14 Others: Reserved
Bit 7:4	EXINT13	0x0	rw	EXINT13 input source configuration These bits are used to select the input source for the EXINT13 external interrupt. 0000: GPIOA pin 13 0001: GPIOB pin 13 0010: GPIOC pin 13 0011: GPIOD pin 13 0100: GPIOE pin 13 0101: GPIOF pin 13 0110: GPIOG pin 13 Others: Reserved
Bit 3:0	EXINT12	0x0	rw	EXINT12 input source configuration These bits are used to select the input source for the EXINT12 external interrupt. 0000: GPIOA pin 12 0001: GPIOB pin 12 0010: GPIOC pin 12 0011: GPIOD pin 12 0100: GPIOE pin 12 0101: GPIOF pin 12 0110: GPIOG pin 12

				Others: Reserved

7.2.7 SCFG ultra high sourcing/sinking strength register (SCFG_UHDRV)

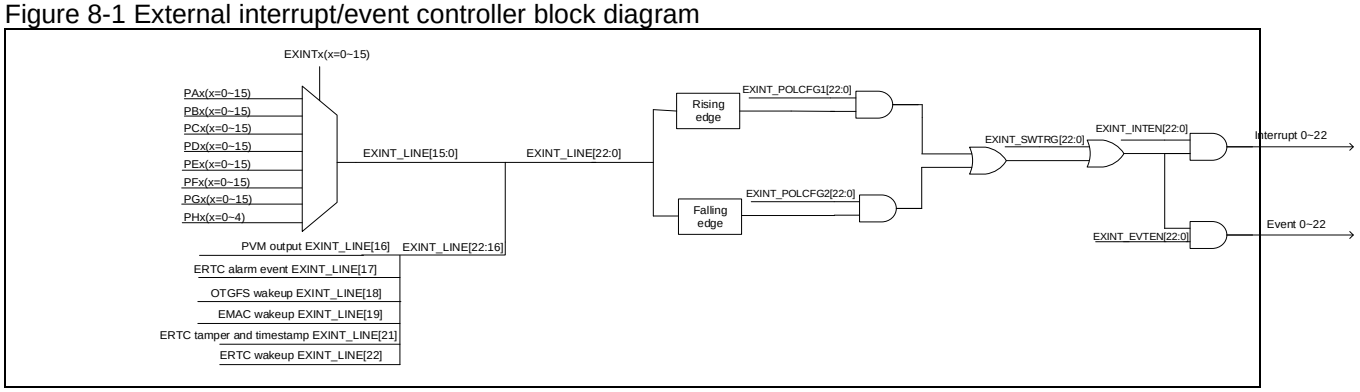
Bit	Name	Reset value	Type	Description
Bit 31:6	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 5	PD15_UH	0x0	rw	PD15 ultra high sourcing/sinking strength This bit is written by software to control PB15 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.
Bit 4	PD14_UH	0x0	rw	PD14 ultra high sourcing/sinking strength This bit is written by software to control PD14 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.
Bit 3	PD13_UH	0x0	rw	PD13 ultra high sourcing/sinking strength This bit is written by software to control PB10 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.
Bit 2	PD12_UH	0x0	rw	PD12 ultra high sourcing/sinking strength This bit is written by software to control PD12 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.
Bit 1	PB9_UH	0x0	rw	PB9 ultra high sourcing/sinking strength This bit is written by software to control PB9 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.
Bit 0	PB8_UH	0x0	rw	PB8 ultra high sourcing/sinking strength This bit is written by software to control PB8 PAD sourcing/sinking strength. 0: Not active 1: Corresponding GPIO is switched to ultra high sourcing/sinking strength When this bit is set, the control bits of GPIOx_ODRVR&GPIOx_HDRV become invalid.

8 External interrupt/event controller (EXINT)

8.1 EXINT introduction

EXINT consists of 22 interrupt lines EXINT_LINE[22:0] (in which bit [20] is reserved), each of which can generate an interrupt or event by edge detection trigger or software trigger. EXINT can enable or disable an interrupt or event independently through software configuration, and utilizes different edge detection modes (rising edge, falling edge or both edges) as well as trigger modes (edge detection, software trigger or both triggers) to respond to trigger source in order to generate an interrupt or event.

Figure 8-1 External interrupt/event controller block diagram



Main features:

- EXINT interrupt lines 0~15 mapping I/O can be configured independently
- Independent trigger selection on each interrupt line
- Independent enable bit for each interrupt
- Independent enable bit for each event
- Up to 22 software triggers that can be generated and cleared independently
- Independent status bit for each interrupt
- Each interrupt can be cleared independently

8.2 Function overview and configuration procedure

With up to 22 interrupt lines EXINT_LINE[22:0] (in which bit [20] is reserved), EXINT can detect not only GPIO external interrupt sources but also six internal sources such as PVM output, ERTC alarm events, ERTC tamper and timestamp events, ERTC wakeup events, OTGFS wakeup events and EMAC wakeup events through edge detection mechanism, where, GPIO interrupt sources can be selected with the SCFG_EXINTCx register. It should be noted that these input sources are mutually exclusive. For example, EXINT_LINE0 is allowed to select one of PA0/PB0/PC0/PD0 pins, instead of taking both PA0 and PB0 as the input sources at the same time.

EXINT supports multiple edge detection modes, including rising edge, falling edge or both edges, selected by EXINT_POLCFG1 and EXINT_POLCFG2 registers. Active edge trigger detected on the interrupt line can be used to generate an event or interrupt.

In addition, EXINT supports independent software trigger for the generation of an event or interrupt. This is achieved by setting the corresponding bits in the EXINT_SWTRG register.

EXINT can enable or disable an interrupt or event individually through software configuration such as EXINT_INTEN and EXINT_EVTEN registers, indicating that the corresponding interrupt or event must be enabled prior to either edge detection or software trigger.

EXINT also features an independent interrupt status bit. Reading access to EXINT_INTSTS register can obtain the corresponding interrupt status. The status flag is cleared by writing “1” to this register.

Interrupt initialization procedure

1. **Select an interrupt soure** by setting the SCFG_EXINTCx register (this is required when GPIO is used as an interrupt source).
2. **Select a trigger mode** by setting the EXINT_POLCFG1 and EXINT_POLCFG2 registers.
3. **Enable interrupt or event** by setting the EXINT_INTEN or EXINT_EVTEN register.
4. **Generate software trigger** by setting the EXINT_SWTRG register (this is applied to only software trigger input).

Note: To modify the interrupt source configuration, disable the EXINT_INTEN and EXINT_EVTEN registers and then restart interrupt initialization.

Interrupt clear procedure

- Writing “1” to the EXINT_INTSTS register to clear the interrupts generated, and the corresponding bits in the EXINT_SWTRG register will be cleared at the same time.

8.3 EXINT registers

The table below shows EXINT register map and their reset value.

These peripheral registers must be accessed by words (32 bits).

Table 8-1 External interrupt/event controller register map and reset value

Register	Offset	Reset value
EXINT_INTEN	0x00	0x0000 0000
EXINT_EVTEN	0x04	0x0000 0000
EXINT_ POLCFG1	0x08	0x0000 0000
EXINT_ POLCFG2	0x0C	0x0000 0000
EXINT_ SWTRG	0x10	0x0000 0000
EXINT_ INTSTS	0x14	0x0000 0000

8.3.1 Interrupt enable register (EXINT_INTEN)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	INTENx	0x000000	rw	Interrupt enable or disable on line x 0: Interrupt request is disabled 1: Interrupt request is enabled Note: Bit [20] is reserved. Unused.

8.3.2 Event enable register (EXINT_EVTEN)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	EVTENx	0x000000	rw	Event enable or disable on line x 0: Event request is disabled 1: Event request is enabled Note: Bit [20] is reserved. Unused.

8.3.3 Polarity configuration register 1 (EXINT_POLCFG1)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	RPx	0x000000	rw	Rising polarity configuration bit of line x These bits are used to select a rising edge to trigger an interrupt and event on line x. 0: Rising edge trigger on line x is disabled 1: Rising edge trigger on line x is enabled Note: Bit [20] is reserved. Unused.

8.3.4 Polarity configuration register 2 (EXINT_POLCFG2)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	FRx	0x000000	rw	Falling polarity event configuration bit of line x These bits are used to select a falling edge to trigger an interrupt and event on line x. 0: Falling edge trigger on line x is disabled 1: Falling edge trigger on line x is enabled Note: Bit [20] is reserved. Unused.

8.3.5 Software trigger register (EXINT_SWTRG)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	SWTx	0x000000	rw	Software triggle on line x If the corresponding bit in the EXINT_INTEN register is 1, the software writes this bit, and the hardware sets the corresponding bit in the EXINT_INTSTS register automatically to generate an interrupt. If the corresponding bit in the EXINT_EVTEN register is 1, the software writes this bit, and the hardware generates an event on the corresponding interrupt line automatically. 0: Default value 1: Software trigger generated Note: This bit is cleared by writing “1” to the corresponding bit in the EXINT_INTSTS register. Note: Bit [20] is reserved. Unused.

8.3.6 Interrupt status register (EXINT_INTSTS)

Bit	Name	Reset value	Type	Description
Bit 31:23	Reserved	0x000	resd	Forced to 0 by hardware.
Bit 22:0	LINEx	0x000000	rw1c	Line x state bit 0: No interrupt occurred 1: Interrupt occurred Note: This bit is cleared by writing “1”. Note: Bit [20] is reserved. Unused.

9 DMA controller (DMA)

9.1 Introduction

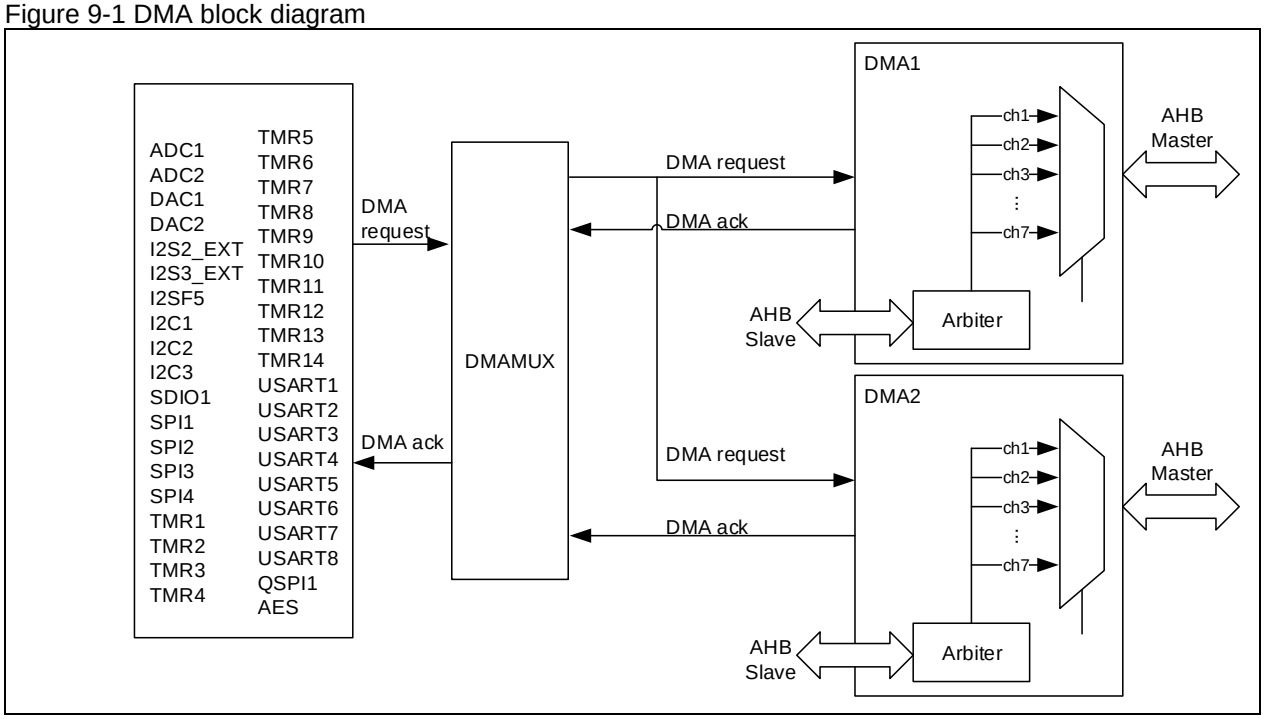
Direct memory access (DMA) controller is designed for high-speed data transmission between peripherals and memory or between memories. The data can be transmitted through DMA at a high speed without CPU interference, which saves CPU capacity.

There are two DMA controllers in the microcontroller. Each controller contains seven DMA channels that manage memory access requests from one or more peripherals. An arbiter is available for coordinating the priority of each DMA request.

9.2 Main features

- AMBA compliant (Rev. 2.0)
- Only support AHB OKAY and ERROR
- HBUSREQ and HGRANT of AHB master interface are not supported
- Support 7 channels
- Peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfers
- Support hardware handshake
- Support 8-bit, 16-bit and 32-bit data transfers
- Programmable amount of data to be transferred: up to 65535
- Support multiplexing

Figure 9-1 DMA block diagram



Note: The number of DMA peripherals may be different for different models.

9.3 Function overview

9.3.1 DMA configuration

1. Set the peripheral address in the DMA_CPBAx register

The initial peripheral address for data transfer remains unchanged during transmission.

2. Set memory address in the DMA_CMBAx register

The initial memory address for data transfer remains unchanged during transmission.

3. Configure the amount of the data to be transferred in the DMA_DTCNTx register

Programmable data transfer size is up to 0xFFFFFFFF. This value is decremented after each data transfer.

4. Configure the channel setting in the DMA_CHCTRLx register

Including channel priority, data transfer direction/width, address incremented mode, circular mode and interrupt mode

Channel priority (CHPL)

There are four levels, including very high priority, high priority, medium priority and low priority.
If the two channels have the same priority level, then the channel with lower number will get priority over the one with higher number. For example, channel 1 has priority over channel 2.

Data transfer direction (DTD)

Memory-to-peripheral (M2P) and peripheral-to-memory (P2M)

Address incremented mode (PINCM/MINCM)

In incremented mode, the subsequent transfer address is the previous address plus transfer width (PWIDTH/MWIDTH).

Circular mode (LM)

In circular mode, the contents in the DMA_DTCNTx register is automatically reloaded with the initially programmed value after the completion of the last transfer.

Memory-to-memory mode (M2M)

This mode indicates that DMA channels perform data transfer without requests from peripherals.
Circular mode and memory-to-memory mode cannot be used at the same time.

5. Enable DMA transfer by setting the CHEN bit in the DMA_CHCTRLx register

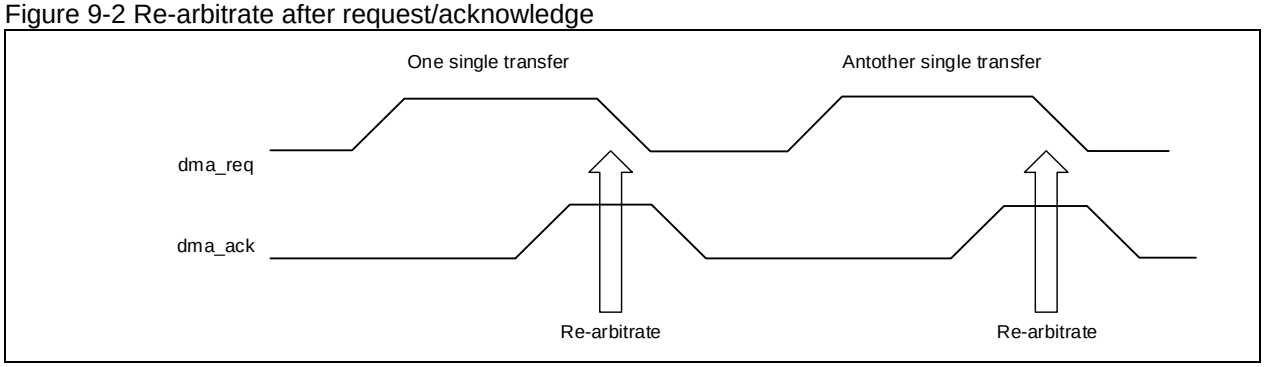
9.3.2 Handshake mechanism

In P2M and M2P mode, the peripherals need to send a request signal to the DMA controller. The DMA channel will send the peripheral transfer request (single) until the signal is acknowledged. After the completion of peripheral transmission, the DMA controller sends the acknowledge signal to the peripheral. The peripheral then releases its request as soon as it receives the acknowledge signal. At the same time, the DMA controller releases the acknowledge signal as well.

9.3.3 Arbiter

When several channels are enabled simultaneously, the arbiter will restart arbitration after full data transfer by the master controller. The channel with very high priority waits until the channel of the master controller has completed data transfers before taking control of it. The master controller will re-arbitrate to serve other channels as long as the channel completes a single transfer based on the master controller priority.

Figure 9-2 Re-arbitrate after request/acknowledge



9.3.4 Programmable data transfer width

Transfer width of the source data and destination data is programmable through the PWIDTH and MWIDTH bits in the DMA_CHCTRLx register. When PWIDTH is not equal to MWIDTH, it can be aligned according to the settings of PWIDTH/ MWIDTH.

Figure 9-3 PWIDTH: byte, MWIDTH: half-word

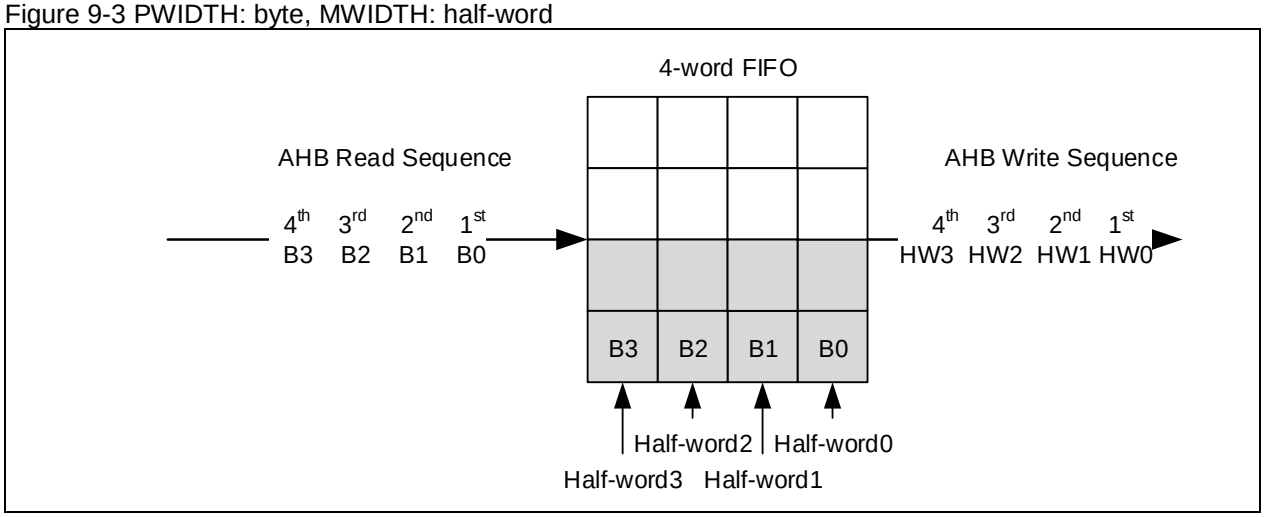


Figure 9-4 PWIDTH: half-word, MWIDTH: word

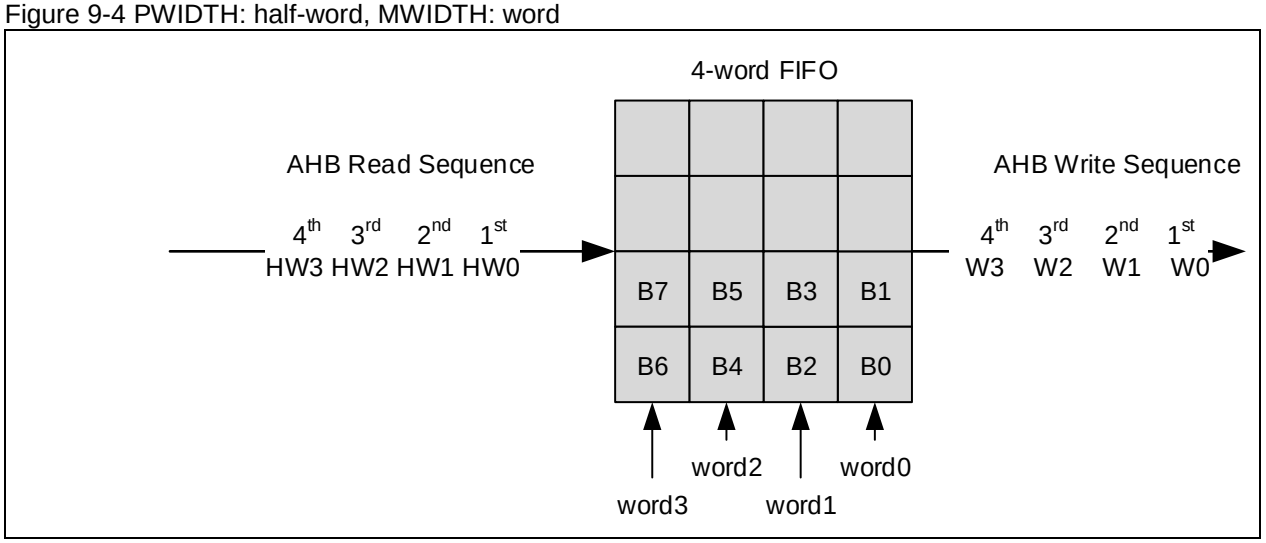
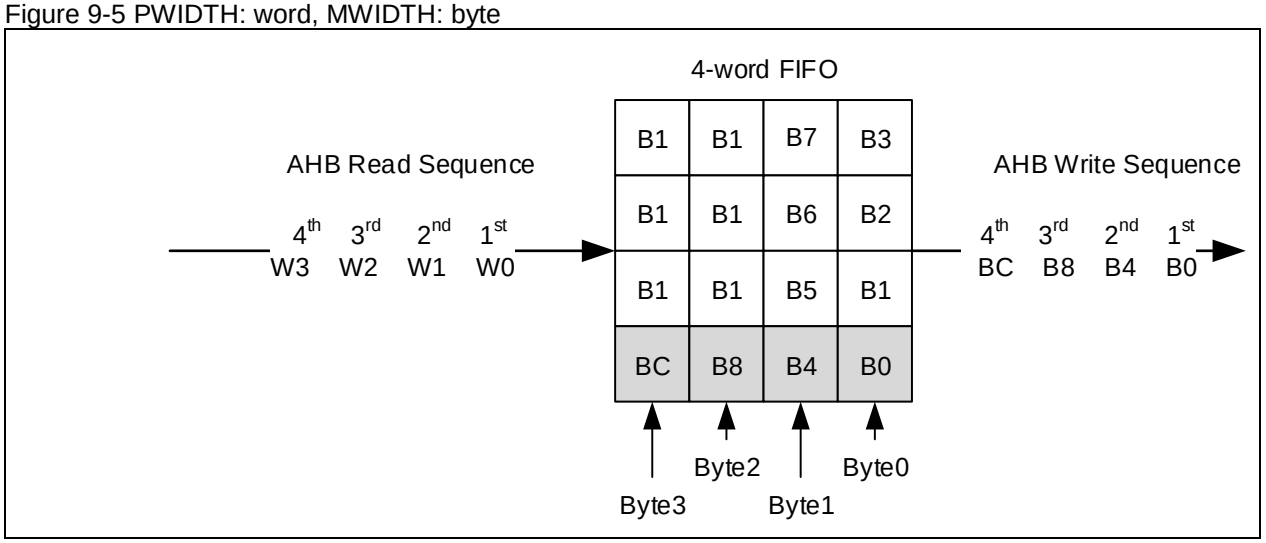


Figure 9-5 PWIDTH: word, MWIDTH: byte



9.3.5 Errors

Table 9-1 DMA error event

Error event	Description
Transfer error	AHB response error occurred during DMA read/write access

9.3.6 Interrupts

An interrupt can be generated on a DMA half-transfer, transfer complete and transfer error. Each channel has its specific interrupt flag, clear and enable bits, as shown in the table below.

Table 9-2 DMA interrupts

Interrupt event	Event flag bit	Clear control bit	Enable control bit
Half-transfer	HDTF	HDTFC	HDTIEN
Transfer completed	FDTF	FDTFC	FDTIEN
Transfer error	DTERRF	DTERRFC	DTERRIEN

9.4 DMA multiplexer (DMAMUX)

DMAMUX manages DMA requests/acknowledge between peripherals and DMA controller.

The DMA controller selects the DMA mapping table with the TBL_SEL bit in the DMA_MUXSEL register. Each DMA controller stream selects only one DMA request from the flexible mapping table. In flexible mapping mode, each channel can bypass or synchronize 127 possible channel requests from peripherals or generators through the REQSEL [6:0] bit in the DMA_MUXCxCTRL register.

9.4.1 DMAMUX function overview

The DMAMUX consists of a request generator and a request multiplexer.

Each of the DMAMUX generator channel x has a GEN enable bit in the DMA_MUXGxCTRL register. The SIGSEL bit is used to select the trigger input pof DMAMUX generator. Typically, the number of DMA requests equals GREQCNT + 1. The GPOL bit in the DMA_MUXGxCTRL register is used to select a trigger event that can be on a rising edge, falling edge or either of them.

Each of the DMAMUX stream x comes from all_req [127:1].

In flexible mapping mode, the SYNCEN bit in the DMA_MUXSxCTRL register is used to synchronize the selected DMA request input. In synchronous mode, the SYNCSEL bit in the DMA_MUXSxCTRL register is used to select synchronized input. The selected DMA request input will be transferred to chx_mux_req [7:0] as soon as a valid edge of the synchronized input is detected by the SYNCPOL [1:0] in the DMA_MUXSxCTRL register. In addition, when the EVTGEN bit in the DMA_MUXCxCTRL register is set, the programmable request counter (REQCNT) is used to generate a request output and event output.

Figure 9-6 DMAMUX block diagram

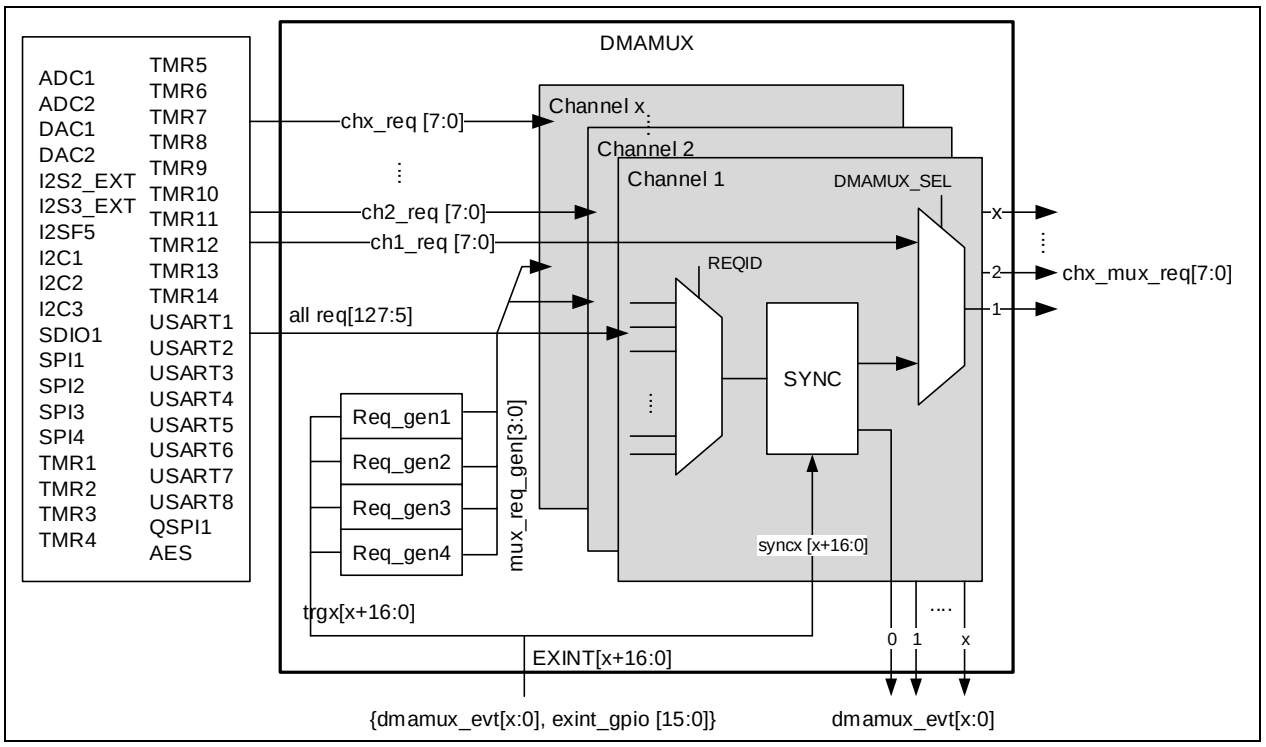

DMAMUX
ADC1
ADC2
DAC1
DAC2
I2S2_EXT
I2S3_EXT
I2SF5
I2C1
I2C2
I2C3
SDIO1
SPI1
SPI2
SPI3
SPI4
TMR1
TMR2
TMR3
TMR4
TMR5
TMR6
TMR7
TMR8
TMR9
TMR10
TMR11
TMR12
TMR13
TMR14
USART1
USART2
USART3
USART4
USART5
USART6
USART7
USART8
QSPI1
AES
chx_req [7:0]
ch2_req [7:0]
ch1_req [7:0]
all req[127:5]
Channel x
Channel 2
Channel 1
DMAMUX_SEL
REQID
SYNC
x
2
1
chx_mux_req[7:0]
Req_gen1
Req_gen2
Req_gen3
Req_gen4
mux_req_gen[3:0]
syncx [x+16:0]
trgx[x+16:0]
EXINT[x+16:0]
0
1
x
{dmamux_evt[x:0], exint_gpio [15:0]}
dmamux_evt[x:0]

Table 9-3 Flexible DMA1 / DMA2 request mapping

CHx_SRC	Request source	CHx_SRC	Request source	CHx_SRC	Request source	CHx_SRC	Request source
1	DMA_MUXREQG1	33	USART5_TX	65	TMR3_OVERFLOW	97	TMR12_TRIG
2	DMA_MUXREQG2	34	reserved	66	TMR3_TRIG	98	TMR12_HALL
3	DMA_MUXREQG3	35	reserved	67	TMR4_CH1	99	reserved
4	DMA_MUXREQG4	36	ADC2	68	TMR4_CH2	100	reserved
5	ADC1	37	reserved	69	TMR4_CH3	101	reserved
6	DAC1	38	reserved	70	TMR4_CH4	102	reserved
7	reserved	39	SDIO1	71	TMR4_OVERFLOW	103	reserved
8	TMR6_OVERFLOW	40	QSPI1	72	TMR5_CH1	104	reserved
9	TMR7_OVERFLOW	41	DAC2	73	TMR5_CH2	105	reserved
10	SPI1_RX	42	TMR1_CH1	74	TMR5_CH3	106	SPI4_RX
11	SPI1_TX	43	TMR1_CH2	75	TMR5_CH4	107	SPI4_TX
12	SPI2_RX	44	TMR1_CH3	76	TMR5_OVERFLOW	108	I2SF5_RX
13	SPI2_TX	45	TMR1_CH4	77	TMR5_TRIG	109	I2SF5_TX
14	SPI3_RX	46	TMR1_OVERFLOW	78	TMR9_CH1	110	I2S2EXT_RX
15	SPI3_TX	47	TMR1_TRIG	79	TMR9_OVERFLOW	111	I2S2EXT _TX
16	I2C1_RX	48	TMR1_HALL	80	TMR9_TRIG	112	I2S3EXT _RX
17	I2C1_TX	49	TMR8_CH1	81	TMR9_HALL	113	I2S3EXT _TX
18	I2C2_RX	50	TMR8_CH2	82	TMR10_CH1	114	USART6_RX
19	I2C2_TX	51	TMR8_CH3	83	TMR10_OVERFLOW	115	USART6_TX
20	I2C3_RX	52	TMR8_CH4	84	TMR11_CH1	116	USART7_RX
21	I2C3_TX	53	TMR8_OVERFLOW	85	TMR11_OVERFLOW	117	USART7_TX
22	reserved	54	TMR8_TRIG	86	reserved	118	USART8_RX
23	reserved	55	TMR8_HALL	87	reserved	119	USART8_TX
24	USART1_RX	56	TMR2_CH1	88	reserved	120	TMR13_CH1
25	USART1_TX	57	TMR2_CH2	89	reserved	121	TMR13_OVERFLOW
26	USART2_RX	58	TMR2_CH3	90	reserved	122	TMR14_CH1
27	USART2_TX	59	TMR2_CH4	91	AES_IN	123	TMR14_OVERFLOW
28	USART3_RX	60	TMR2_OVERFLOW	92	AES_OUT	124	TMR9_CH2
29	USART3_TX	61	TMR3_CH1	93	reserved	125	TMR12_CH2
30	USART4_RX	62	TMR3_CH2	94	reserved	126	TMR2_TRIG
31	USART4_TX	63	TMR3_CH3	95	TMR12_CH1	127	TMR4_TRIG
32	USART5_RX	64	TMR3_CH4	96	TMR12_OVERFLOW		

Table 9-4 DMAMUX EXINT LINE for trigger input and synchronized input

EXINT LINE	Source	EXINT LINE	Source	EXINT LINE	Source	EXINT LINE	Source
0	exint_gpio[0]	8	exint_gpio[8]	16	DMA_MUXevt1	24	reserved
1	exint_gpio[1]	9	exint_gpio[9]	17	DMA_MUXevt2	25	reserved
2	exint_gpio[2]	10	exint_gpio[10]	18	DMA_MUXevt3	26	reserved
3	exint_gpio[3]	11	exint_gpio[11]	19	DMA_MUXevt4	27	reserved
4	exint_gpio[4]	12	exint_gpio[12]	20	DMA_MUXevt5	28	reserved
5	exint_gpio[5]	13	exint_gpio[13]	21	DMA_MUXevt6	39	reserved
6	exint_gpio[6]	14	exint_gpio[14]	22	DMA_MUXevt7	30	reserved
7	exint_gpio[7]	15	exint_gpio[15]	23	reserved	31	reserved

9.4.2 DMAMUX overflow interrupts

During DMAMUX request generation, when a new trigger input occurs before the GREQCNT underflows, the TRGOVFx bit will be set in the DMA_MUXGSTS register. It is cleared by setting TRGOVFCx =1 in the DMA_MUXGCLR register. an interrupt will be generated if the interrupt enable bit TRGOVIEN is set in the DMA_MUXGxCTRL register.

In DMAMUX synchronous mode, when a new synchronized input occurs before the REQCNT underflows, the SYNCOVFx bit will bt set in the DMA_MUXSYNCSTS register. It is cleared by setting SYNCOVFCx=1 in the DMA_MUXSYNCCLR register. An interrupt will be generated if the interrupt enable bit SYNCOVIEN is set in the DMA_MUXSxCTRL register.

Figure 9-7 DMAMUX request synchronized mode

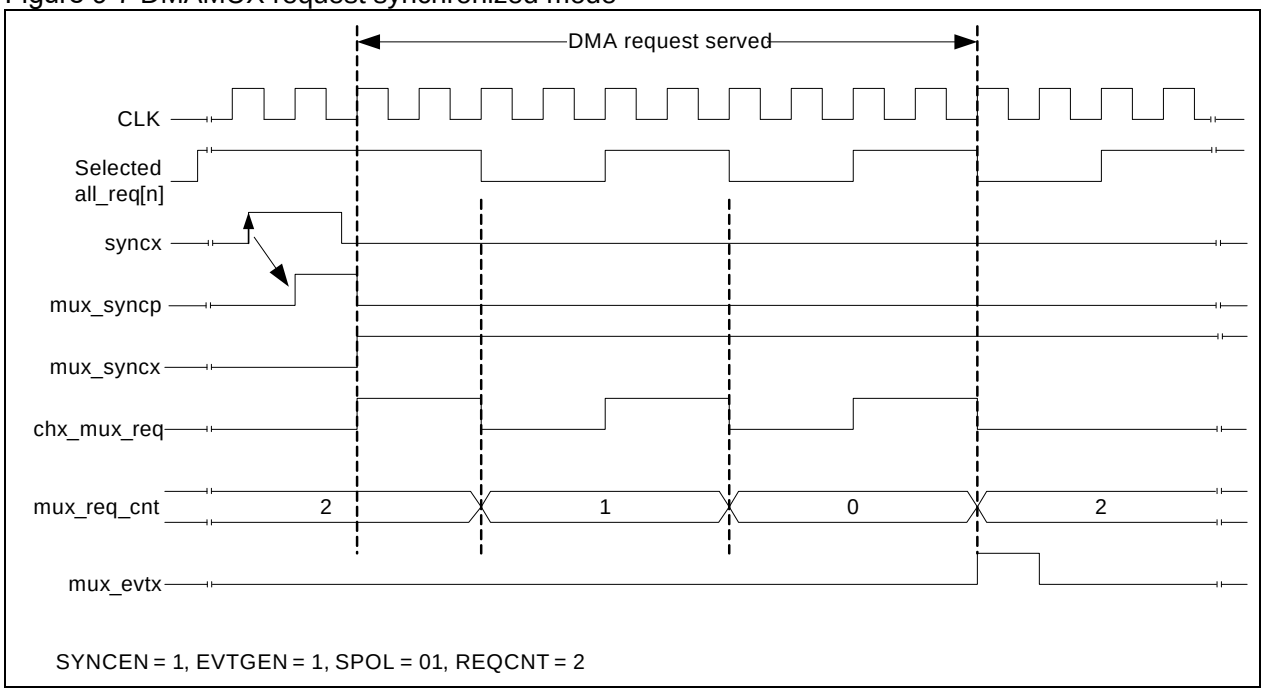
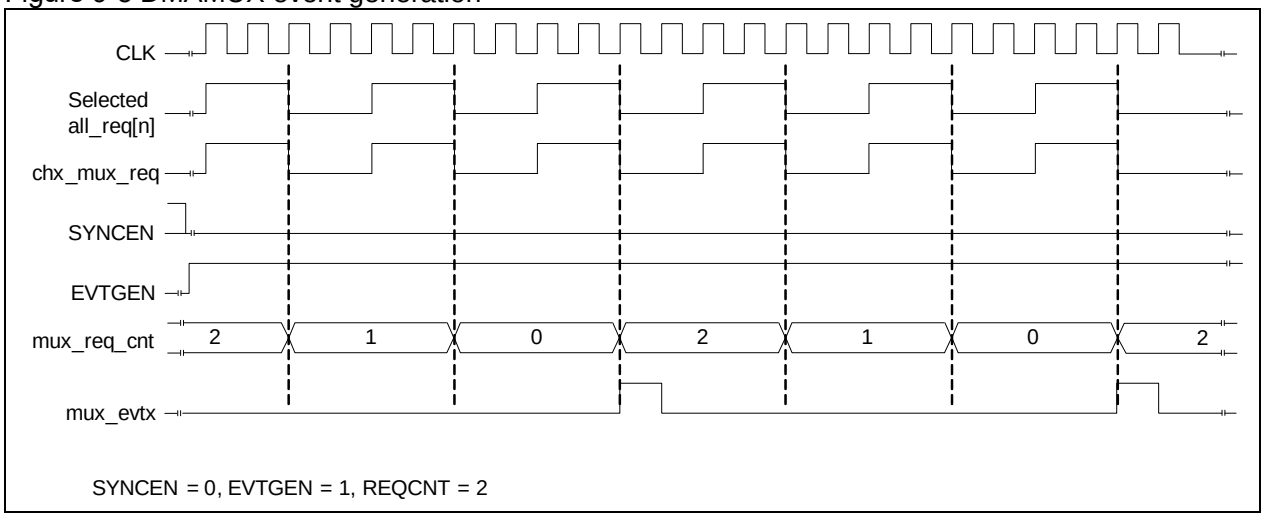


Figure 9-8 DMAMUX event generation



9.5 DMA registers

The table below lists DMA register map and their reset values. These peripheral registers can be accessed by bytes (8 bits), half-words (16 bits) or words (32 bits).

Table 9-5 DMA register map and reset value

Register	Offset	Reset value
DMA_STS	0x00	0x0000 0000
DMA_CLR	0x04	0x0000 0000
DMA_C1CTRL	0x08	0x0000 0000
DMA_C1DTCNT	0x0c	0x0000 0000
DMA_C1PADDR	0x10	0x0000 0000
DMA_C1MADDR	0x14	0x0000 0000
DMA_C2CTRL	0x1c	0x0000 0000
DMA_C2DTCNT	0x20	0x0000 0000
DMA_C2PADDR	0x24	0x0000 0000
DMA_C2MADDR	0x28	0x0000 0000
DMA_C3CTRL	0x30	0x0000 0000
DMA_C3DTCNT	0x34	0x0000 0000
DMA_C3PADDR	0x38	0x0000 0000
DMA_C3MADDR	0x3c	0x0000 0000
DMA_C4CTRL	0x44	0x0000 0000
DMA_C4DTCNT	0x48	0x0000 0000
DMA_C4PADDR	0x4c	0x0000 0000
DMA_C4MADDR	0x50	0x0000 0000
DMA_C5CTRL	0x58	0x0000 0000
DMA_C5DTCNT	0x5c	0x0000 0000
DMA_C5PADDR	0x60	0x0000 0000
DMA_C5MADDR	0x64	0x0000 0000
DMA_C6CTRL	0x6c	0x0000 0000
DMA_C6DTCNT	0x70	0x0000 0000

DMA_C6PADDR	0x74	0x0000 0000
DMA_C6MADDR	0x78	0x0000 0000
DMA_C7CTRL	0x80	0x0000 0000
DMA_C7DTCNT	0x84	0x0000 0000
DMA_C7PADDR	0x88	0x0000 0000
DMA_C7MADDR	0x8c	0x0000 0000
DMA_MUXSEL	0x100	0x0000 0000
DMA_MUXC1CTRL	0x104	0x0000 0000
DMA_MUXC2CTRL	0x108	0x0000 0000
DMA_MUXC3CTRL	0x10c	0x0000 0000
DMA_MUXC4CTRL	0x110	0x0000 0000
DMA_MUXC5CTRL	0x114	0x0000 0000
DMA_MUXC6CTRL	0x118	0x0000 0000
DMA_MUXC7CTRL	0x11c	0x0000 0000
DMA_MUXG1CTRL	0x120	0x0000 0000
DMA_MUXG2CTRL	0x124	0x0000 0000
DMA_MUXG3CTRL	0x128	0x0000 0000
DMA_MUXG4CTRL	0x12c	0x0000 0000
DMA_MUXSYNCSTS	0x130	0x0000 0000
DMA_MUXSYNCCLR	0x134	0x0000 0000
DMA_MUXGSTS	0x138	0x0000 0000
DMA_MUXGCLR	0x13c	0x0000 0000

9.5.1 DMA interrupt status register (DMA_STS)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:28	Reserved	0x0	resd	Kept at its default value.
Bit 27	DTERRF7	0x0	ro	Channel 7 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 26	HDTF7	0x0	ro	Channel 7 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 25	FDTF7	0x0	ro	Channel 7 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 24	GF7	0x0	ro	Channel 7 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 23	DTERRF6	0x0	ro	Channel 6 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 22	HDTF6	0x0	ro	Channel 6 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 21	FDTF6	0x0	ro	Channel 6 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 20	GF6	0x0	ro	Channel 6 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 19	DTERRF5	0x0	ro	Channel 5 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 18	HDTF5	0x0	ro	Channel 5 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 17	FDTF5	0x0	ro	Channel 5 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 16	GF5	0x0	ro	Channel 5 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 15	DTERRF4	0x0	ro	Channel 4 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred

Bit 14	HDTF4	0x0	ro	Channel 4 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 13	FDTF4	0x0	ro	Channel 4 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 12	GF4	0x0	ro	Channel 4 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 11	DTERRF3	0x0	ro	Channel 3 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 10	HDTF3	0x0	ro	Channel 3 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 9	FDTF3	0x0	ro	Channel 3 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 8	GF3	0x0	ro	Channel 3 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 7	DTERRF2	0x0	ro	Channel 2 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 6	HDTF2	0x0	ro	Channel 2 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 5	FDTF2	0x0	ro	Channel 2 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 4	GF2	0x0	ro	Channel 2 global event flag 0: No transfer error, half-transfer or transfer complete
Bit 3	DTERRF1	0x0	ro	Channel 1 data transfer error event flag 0: No transfer error occurred 1: Transfer error occurred
Bit 2	HDTF1	0x0	ro	Channel 1 half transfer event flag 0: No half-transfer event occurred 1: Half-transfer event occurred
Bit 1	FDTF1	0x0	ro	Channel 1 transfer complete event flag 0: No transfer complete event occurred 1: Transfer complete event occurred
Bit 0	GF1	0x0	ro	Channel 1 global event flag 0: No transfer error, half-transfer or transfer complete

9.5.2 DMA interrupt flag clear register (DMA_CLR)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:28	Reserved	0x0	resd	Kept at its default value.
Bit 27	DTERRFC7	0x0	rw1c	Channel 7 data transfer error flag clear 0: No effect 1: Clear the DTERRF7 flag in the DMA_STS
Bit 26	HDTFC7	0x0	rw1c	Channel 7 half transfer flag clear 0: No effect 1: Clear the HDTF7 flag in the DMA_STS register
Bit 25	FDTFC7	0x0	rw1c	Channel 7 transfer complete flag clear 0: No effect 1: Clear the FDTF7 flag in the DMA_STS register
Bit 24	GFC7	0x0	rw1c	Channel 7 global flag clear 0: No effect 1: Clear the DTERRF7, HDTF7, FDTF7 and GF7
Bit 23	DTERRFC6	0x0	rw1c	Channel 6 data transfer error flag clear 0: No effect 1: Clear the DTERRF6 flag in the DMA_STS
Bit 22	HDTFC6	0x0	rw1c	Channel 6 half transfer flag clear 0: No effect 1: Clear the HDTF6 flag in the DMA_STS register
Bit 21	FDTFC6	0x0	rw1c	Channel 6 transfer complete flag clear 0: No effect 1: Clear the FDTF6 flag in the DMA_STS register
Bit 20	GFC6	0x0	rw1c	Channel 6 global flag clear 0: No effect 1: Clear the DTERRF6, HDTF6, FDTF6 and GF6
Bit 19	DTERRFC5	0x0	rw1c	Channel 5 data transfer error flag clear 0: No effect 1: Clear the DTERRF5 flag in the DMA_STS
Bit 18	HDTFC5	0x0	rw1c	Channel 5 half transfer flag clear 0: No effect 1: Clear the HDTF5 flag in the DMA_STS register
Bit 17	FDTFC5	0x0	rw1c	Channel 5 transfer complete flag clear 0: No effect 1: Clear the FDTF5 flag in the DMA_STS register
Bit 16	GFC5	0x0	rw1c	Channel 5 global flag clear 0: No effect 1: Clear the DTERRF5, HDTF5, FDTF5 and GF5
Bit 15	DTERRFC4	0x0	rw1c	Channel 4 data transfer error flag clear 0: No effect 1: Clear the DTERRF4 flag in the DMA_STS

Bit 14	HDTFC4	0x0	rw1c	Channel 4 half transfer flag clear 0: No effect 1: Clear the HDTF4 flag in the DMA_STS register
Bit 13	FDTFC4	0x0	rw1c	Channel 4 transfer complete flag clear 0: No effect 1: Clear the FDTF4 flag in the DMA_STS register
Bit 12	GFC4	0x0	rw1c	Channel 4 global flag clear 0: No effect 1: Clear the DTERRF4, HDTF4, FDTF4 and GF4
Bit 11	DTERRFC3	0x0	rw1c	Channel 3 data transfer error flag clear 0: No effect 1: Clear the DTERRF3 flag in the DMA_STS
Bit 10	HDTFC3	0x0	rw1c	Channel 3 half transfer flag clear 0: No effect 1: Clear the HDTF3 flag in the DMA_STS register
Bit 9	FDTFC3	0x0	rw1c	Channel 3 transfer complete flag clear 0: No effect 1: Clear the FDTF3 flag in the DMA_STS register
Bit 8	GFC3	0x0	rw1c	Channel 3 global flag clear 0: No effect 1: Clear the DTERRF3, HDTF3, FDTF3 and GF3
Bit 7	DTERRFC2	0x0	rw1c	Channel 2 data transfer error flag clear 0: No effect 1: Clear the DTERRF2 flag in the DMA_STS
Bit 6	HDTFC2	0x0	rw1c	Channel 2 half transfer flag clear 0: No effect 1: Clear the HDTF2 flag in the DMA_STS register
Bit 5	FDTFC2	0x0	rw1c	Channel 2 transfer complete flag clear 0: No effect 1: Clear the FDTF2 flag in the DMA_STS register
Bit 4	GFC2	0x0	rw1c	Channel 2 global flag clear 0: No effect 1: Clear the DTERRF2, HDTF2, FDTF2 and GF2
Bit 3	DTERRFC1	0x0	rw1c	Channel 1 data transfer error flag clear 0: No effect 1: Clear the DTERRF1 flag in the DMA_STS
Bit 2	HDTFC1	0x0	rw1c	Channel 1 half transfer flag clear 0: No effect 1: Clear the HDTF1 flag in the DMA_STS register
Bit 1	FDTFC1	0x0	rw1c	Channel 1 transfer complete flag clear 0: No effect 1: Clear the FDTF1 flag in the DMA_STS register

Bit 0	GFC1	0x0	rw1c	Channel 1 global flag clear 0: No effect 1: Clear the DTERRF1, HDTF1, FDTF1 and GF1

9.5.3 DMA channel-x configuration register (DMA_CxCTRL) (x = 1…7)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:15	Reserved	0x00000	resd	Kept at its default value.
Bit 14	M2M	0x0	rw	Memory-to-memory mode 0: Disabled 1: Enabled
Bit 13:12	CHPL	0x0	rw	Channel priority level 00: Low 01: Medium 10: High 11: Very high
Bit 11:10	MWIDTH	0x0	rw	Memory data bit width 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved
Bit 9:8	PWIDTH	0x0	rw	Peripheral data bit width 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved
Bit 7	MINCM	0x0	rw	Memory address increment mode 0: Disabled 1: Enabled
Bit 6	PINCM	0x0	rw	Peripheral address increment mode 0: Disabled 1: Enabled
Bit 5	LM	0x0	rw	Circular mode 0: Disabled 1: Enabled
Bit 4	DTD	0x0	rw	Data transfer direction 0: Read from peripherals 1: Read from memory
Bit 3	DTERRIEN	0x0	rw	Data transfer error interrupt enable 0: Disabled 1: Enabled
Bit 2	HDTIEN	0x0	rw	Half transfer interrupt enable 0: Disabled

				1: Enabled
Bit 1	FDTIEN	0x0	rw	Transfer complete interrupt enable 0: Disabled 1: Enabled
Bit 0	CHEN	0x0	rw	Channel enable 0: Disabled 1: Enabled

9.5.4 DMA channel-x number of data register (DMA_CxDTCNT) (x = 1…7)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:0	CNT	0x0000 0000	rw	Number of data to transfer The number of data to transfer is from 0x0 to 0xFFFFFFFF. This register can only be written when CHEN=0 in the corresponding channel register. The value is decremented by 1 after each DMA transfer.

9.5.5 DMA channel-x peripheral address register (DMA_CxPADDR) (x = 1…7)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:0	PADDR	0x0000 0000	rw	Peripheral base address Base address of peripheral data register is the source or destination of data transfer

9.5.6 DMA channel-x memory address register (DMA_CxMADDR) (x = 1…7)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:0	MADDR	0x0000 0000	rw	Memory base address Memory address is the source or destination of data transfer

9.5.7 DMAMUX select register (DMA_MUXSEL)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	TBL_SEL	0x0	rw	Multiplexer table select 0x1: Flexible mapping table

9.5.8 DMAMUX channel-x control register (DMA_MUXCxCTRL) (x = 1…7)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:25	Reserved	0x00	resd	Kept at its default value.
Bit 28:24	SYNCSEL	0x00	rw	Synchronization select
Bit 23:19	REQCNT	0x00	rw	DMA request count These bits indicate the number of DMA requests sent to the DMA controller after synchronization is enabled, and/or DMA request count before event output is generated. These bits are reserved when both SYNCEN and EVTGEN bits are LOW.
Bit 18:17	SYNCPOL	0x0	rw	Synchronization polarity It is used to define the polarity of the selected synchronization input 0x0: No event 0x1: Rising edge 0x2: Falling edge 0x3: Rising edge and falling edge
Bit 16	SYNCEN	0x0	rw	Synchronization enable 0: Disabled 1: Enabled
Bit 15:10	Reserved	0x00	resd	Kept at its default value.
Bit 9	EVTGEN	0x0		Event generate enable 0: Disabled 1: Enabled
Bit 8	SYNCOVIEN	0x0		Synchronization overrun interrupt enable 0: Interrupt disabled 1: Interrupt enabled
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6:0	REQSEL	0x00		DMA request select It is used to select the input DMA request. Refer to the DMAMUX table for the configuration of multiplexer input.

9.5.9 DMAMUX generator-x control register (DMA_MUXGxCTRL) (x = 1…4)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	resd	Kept at its default value.
Bit 23:19	GREQCNT	0x00	rw	DMA request generation count It is used to define the number of DMA request (GNBREQ + 1) to be generated after trigger event. This field is reserved only when the GEN bit is disabled.
Bit 18:17	GPOL	0x0	rw	DMA request generation polarity This field defines the polarity of the selected trigger input. 0x0: No event 0x1: Rising edge 0x2: Falling edge 0x3: Rising edge and falling edge
Bit 16	GEN	0x0	rw	DMA request generation enable 0: Disabled 1: Enabled
Bit 15:9	Reserved	0x00	resd	Kept at its default value.
Bit 8	TRGOVIEN	0x0	rw	Trigger overrun interrupt enable 0: Interrupt disabled 1: Interrupt enabled
Bit 7:5	Reserved	0x0	resd	Kept at its default value.
Bit 4:0	SIGSEL	0x00	rw	Signal select It is used to select the DMA trigger input for DMA request generator.

9.5.10 DMAMUX channel synchronization status register (DMA_MUXSYNCSTS)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:7	Reserved	0x0000 00	resd	Kept at its default value.
Bit 6:0	SYNCOVF	0x00	ro	Synchroniztion overrun interrupt flag The synchronization overrun interrupt occurs when the DMA request counter is below REQCNT. This flag is set when a new synchronization event occurs.

9.5.11 DMAMUX channel interrupt flag clear register (DMA_MUXSYNCCLR)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:7	Reserved	0x0000 00	resd	Kept at its default value.
Bit 6:0	SYNCOVFC	0x00	rw1c	Synchronization overrun interrupt flag clear Writing 1 to the corresponding bit can clear the SYNCOVF flag in the MUXSYNCSTS register.

9.5.12 DMAMUX generator interrupt status register (DMA_MUXGSTS)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:4	Reserved	0x0000 000	resd	Kept at its default value.
Bit 3:0	TRGOVF	0x00	ro	Trigger overrun interrupt flag When the DMA request count is lower than GREQCNT, this field is set while a new trigger event occurs.

9.5.13 DMAMUX generator interrupt flag clear register (DMA_MUXGCLR)

Access: 0 wait state, accessible by bytes, half-words or words

Bit	Name	Reset value	Type	Description
Bit 31:4	Reserved	0x0000 000	resd	Kept at its default value.
Bit 3:0	TRGOVFC	0x00	rw1c	Trigger overrun interrupt flag clear Writing 1 to the corresponding bit can clear the TRGOVF flag in the DMA_MUXGSTS register.

10 CRC calculation unit (CRC)

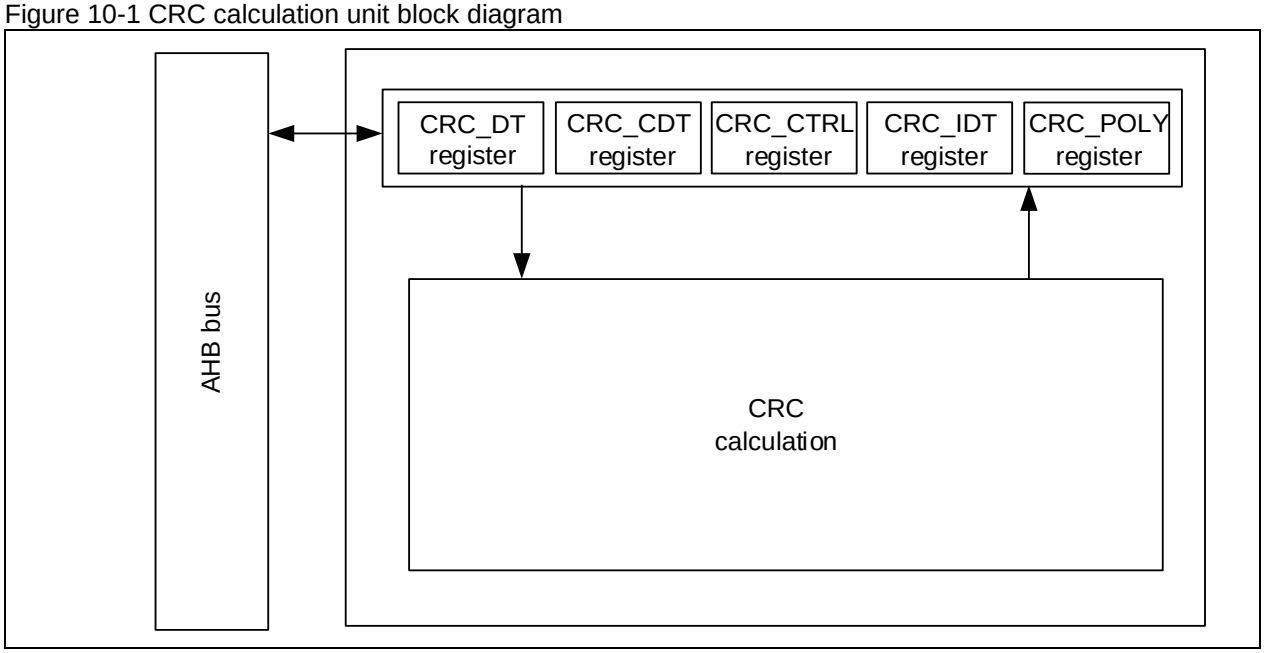
10.1 CRC introduction

The Cyclic Redundancy Check (CRC) is an independent peripheral with CRC check feature. It follows CRC32/MPEG-2 standard.

The CRC_CTRL register is used to select output data toggle (word, REVOD=1) or input data toggle (byte, REVID=01, half-word: REVID=10, word: REVID=11). The CRC calculation unit also has initialization function. After each CRC reset, the value in the CRC_IDT register is written into the CRC_DT register by CRC. The CRC_POLY register can be used to program the polynomial coefficient, and the polynomial size can be set to 7-bit, 8-bit, 16-bit or 32-bit by setting the POLY_SIZE bit in the CRC_CTRL register.

Users can write the data to go through CRC check and read the calculated result through CRC_DT register. Note that the calculation result is the combination of the previous result and the current value to be calculated.

Figure 10-1 CRC calculation unit block diagram



Main features:

- Use CRC-32 code
- Programmable polynomial
- 4x HCLK cycles for each CRC calculation
- Support input/output data format toggle
- Perform write/read operation through CRC_DT register
- Set an initialization value with the CRC_IDT register. The value is loaded into CRC_DT register after each CRC reset.

10.2 CRC function edscription

In CRC calculation, the input data is used as the dividend and the generating polynomial as the divisor for Modulo-2 Division, and the remainder obtained is the CRC value.

CRC calculation procedure

- Toggle input, that is, toggle the input data according to the REVID value in the CRC_CTRL register.
- Initialize: perform XOR with the initial value set in the CRC_IDT for the first time of calculation (if it is not the first time, the initial value should be the previously calculated result).
- CRC calculation: perform Modulo-2 Division with the generating polynomial (0x4C11DB7), and the remainder obtained is the CRC value.
- Toggle output: determine whether to perform toggle (word) according to the REVOD value in the CRC_CTRL register before output.
- Perform XOR calculation for the result, and the XOR-ed value is fixed to 0x0000 0000.

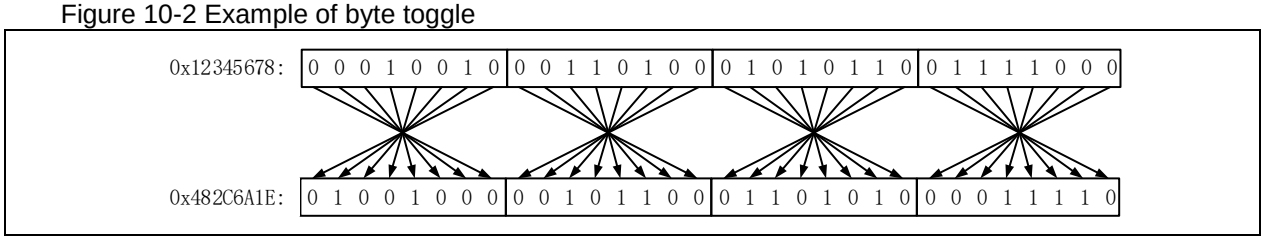
CRC-32/MPEG-2 parameters

- Generating polynomial: 0x4C11DB7,
  $$X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}+X^{7}+X^{5}+X^{4}+X^{2}+X+1$$
- Initial value is 0xFFFF FFFF to avoid obtaining the same calculation result for 1-byte 0x00 and multibyte 0x00.
- XOR-ed value: 0x0000 0000, indicating the CRC result requires an additional XOR calculation.

Toggle function

- Select byte toggle: 8 bits as a group and in reverse order. As shown in the figure below, if the original data is 0x12345678, the toggled data is 0x482C6A1E.
- Select half-word toggle: 16 bits as a group and in reverse order.
- Select word toggle: 32 bits as a group and in reverse order.

Figure 10-2 Example of byte toggle



10.3 CRC registers

Table 10-1 CRC register map and reset value

Register	Offset	Reset value
CRC_DT	0x00	0xFFFF FFFF
CRC_CDT	0x04	0x0000 0000
CRC_CTRL	0x08	0x0000 0000
CRC_IDT	0x10	0xFFFF FFFF
CRC_POLY	0x14	0x04C1 1DB7

10.3.1 Data register (CRC_DT)

Bit	Name	Reset value	Type	Description
Bit 31:0	DT	0xFFFF FFFF	rw	Data value It is used as input register when writing new data into the CRC calculator. It returns CRC calculation results when it is read.

10.3.2 Common data register (CRC_CDT)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	resd	Kept at its default value.
Bit 7:0	CDT	0x00	rw	Common 8-bit data value This field is used to store one byte of data temporarily. This register is not affected by the CRC reset generated by the RST bit in the CRC_CTRL register

10.3.3 Control register (CRC_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	resd	Kept at its default value.
Bit 7	REVOD	0x0	resd	Reverse output data It is set and cleared by software. This bit is used to control whether or not to reverse output data. 0: No effect 1: Word reverse
Bit 6:5	REVID	0x0	rw	Reverse input data It is set and cleared by software. This bit is used to control how to reverse input data. 00: No effect 01: Byte reverse 10: Half-word reverse 11: Word reverse
Bit 4:3	POLY_SIZE	0x0	rw	Polynomial size This bit is used to set the polynomial size. It works with the CRC_POLY register. 00: 32-bit 01: 16-bit 10: 8-bit 11: 7-bit
Bit 2:1	Reserved	0x0	resd	Kept at its default value.
Bit 0	RST	0x0	wo	Reset CRC calculation unit It is set by software and cleared by hardware. To reset CRC calculation unit, the data register is set as 0xFFFF FFFF. 0: No effect 1: Reset

10.3.4 Initialization register (CRC_IDT)

Bit	Name	Reset value	Type	Description
Bit 31:0	IDT	0xFFFF FFFF	rw	Initial data value When CRC reset is triggered by the RST bit in the CRC_CTRL register, the value in the initialization register is written into the CRC_DT register as an initial value.

10.3.5 Polynomial register (CRC_POLY)

Bit	Name	Reset value	Type	Description
Bit 31:0	POLY	0x04C1 1DB7	rw	Polynomial coefficient The generating polynomial is used as the division in CRC calculation. In CRC32 algorithm, the polynomial coefficient is set to 0x4C11DB7. The polynomial can also be programmed by users.

11 I²C interface

11.1 I²C introduction

I²C (inter-integrated circuit) bus interface manages the communication between the microcontroller and serial I²C bus. It supports master and slave modes, with up to 1 Mbit/s of communication speed (fast mode plus).

11.2 I²C main features

- I²C bus
  - Master and slave modes
  - Multimaster capability
  - Standard mode (100 KHz), fast mode (400 KHz) and fast mode plus (1 MHz)
  - 7-bit and 10-bit address mode
  - Two 7-bit slave addresses (2 addresses, one of them can be masked)
  - Broadcast call mode
  - Programmable data setup and hold time
  - Clock stretching capability
- Support DMA transfer
- Programmable digital noise filter
- Support SMBus 2.0 protocol
  - PEC generation and verification
  - Acknowledgement control for command and data
  - ARP (address resolution protocol)
  - Master capability
  - Device capability
  - SMBus reminder capability
  - Timeout detection
  - Idle detection
- PMBus

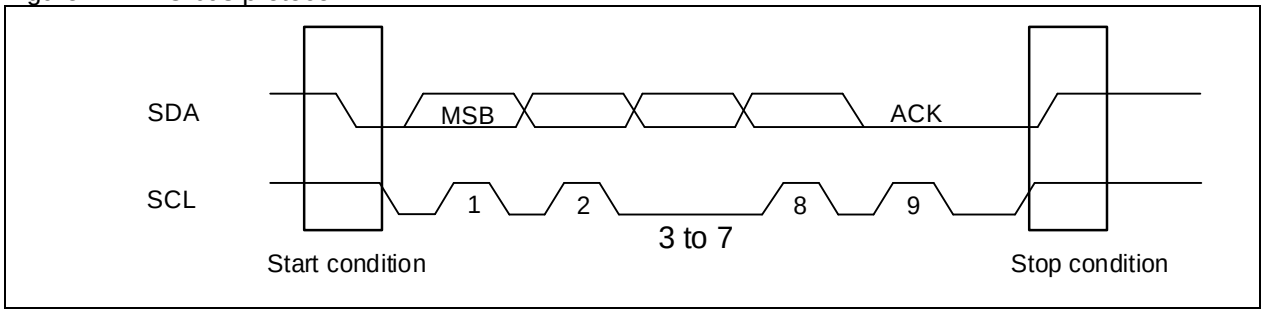
11.3 I²C function overview

I²C bus consists of a data line (SDA) and clock line (SCL). It can achieve a maximum of 100 KHz speed in standard mode, up to 400 KHz in fast mode, and up to 1 MHz in fast mode plus. A frame of data transmission begins with a Start condition and ends with a Stop condition. The bus is kept in busy state after receiving the Start condition, and becomes idle as long as it receives the Stop condition.

Start condition: SDA switches from high to low when SCL is set high

Stop condition: SDA switches from low to high when SCL is set high

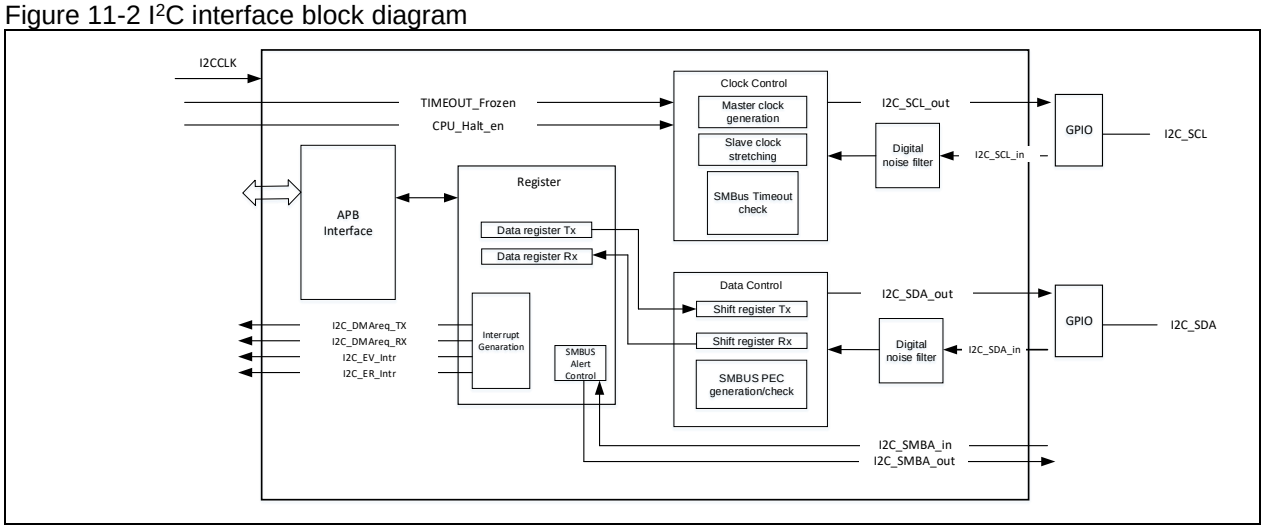
Figure 11-1 I²C bus protocol



11.4 I²C interface

The figure below shows the block diagram of I²C interfae.

Figure 11-2 I²C interface block diagram



1. Operating mode

I²C bus interface can operate both in master mode and slave mode. Switching from master mode to slave mode, vice versa, is supported as well. By default, the interface operates in slave mode. When the Start condition is activated, the I²C bus interface switches from slave mode to master mode, and returns to slave mode automatically at the end of data transfer (Stop condition is triggered).

2. Communication process

- Master mode communication:
  1. Start condition generation
  2. Address transmission
  3. Data Tx or Rx
  4. Stop condition generation
  5. End of communication
- Slave mode communication:
  1. Wait until the address is matched
  2. Data Tx or Rx
  3. Wait for the generation of Stop condition
  4. End of communication

3. Digital filter capability

The digital filter is avail on both SCL and SDA lines. It is enabled by setting the DFLT[3:0] bit (0~15) in the I2C_CTRL1 register to reduce noise on bus on a large scale. The filter time is DFLT x T_{I2C_CLK}. The digital filter is not allowed to be altered when the I²C is enabled.

4. Address control

Both master and slave support 7-bit and 10-bit addressing modes.

Slave address mode:

- In 7-bit address mode (ADDR1MODE=0)
  - ADDR1EN=1, ADDR2EN=0 stands for a single address mode: only matches OADDR1
  - ADDR1EN=1, ADDR2EN=1 stands for dual address mode: matches OADDR1 and OADDR2
- In 10-bit address mode (ADDR1MODE=1)
  - Only supports a single address mode (ADDR1EN=1, ADDR2EN=0), matches OADDR1

Slave address masking capability

The slave address 2 (OADDR2) is maskable, which is done by setting the ADDR2MASK[2:0].

- 0: Address bit [7:1]
- 1: Address bit [7:2]
- 2: Address bit [7:3]
- 3: Address bit [7:4]
- 4: Address bit [7:5]
- 5: Address bit [7:6]
- 6: Address bit [7]
- 7: All addresses, excluding those reserved by I²C

Support special slave address:

- Broadcast call address (0b0000000x): This address is enabled when GCAEN=1.
- SMBus device default address (0b1100001x): This address is enabled for SMBus address resolution protocol in SMBus device mode (DEVADDREN = 1).
- SMBus master default address (0b0001000x): This address is enabled for SMBus master notification protocol in SMBus master mode (HADDREN = 1).
- SMBus alert address (0b0001100x): This address is enabled for SMBus alert response address protocol in SMBus device mode when SMBALERT = 1.

Refer to SMBus2.0 protocol for more information.

Slave address matching proedure:

- Receive a Start condition
- Address matching
- The salve sends an ACK if address is matched
- ADDRF=1, with SDIR indicating the transmission direction
  - When SDIR=0, slave enters receiver mode, starting receiving data
  - When SDIR=1, slave enters transmitter mode, starting transmitting data

5. Clock stretching capability

Clock stretching is enabled by default (STRETCH=0 in the I2C_CTRL1 register). The slave can hold the SCL line low for software operation. If the clock stretching capability is not supported by master, then the STRETCH must be set to 1 in the I2C_CTRL1 register. It should be noted that the clock stretching capability of I²C slave must be configured before the I²C peripherals are enabled.

Clock stretching capability enabled

I²C slave stretches the SCL clock in one of the following conditions:

- Address reception: When the address received by slave matches the local address enabled (ADDRF=1 in the I2C_STS register), the SCL line is pulled down until the ADDRF is cleared by setting the ADDRC bit in the I2C_CLR register.
- Data reception: When the shift register has received another new byte before the data in the I2C_RXDT register is read, the SCL line will be pulled low until the data in the I2C_RXDT register is read.
- Data transmission: If no data is written when the ADDRF is cleared, and TDBE= 1 in the I2C_STS register, then SCL line will be pulled low until data is written to the I2C_TXDT register.
- Data transmission: If no data is written to the I2C_TXDT register after the completion of the previous data transfer, the SCL line will be pulled low until data is written to the I2C_TXDT register.
- When slave byte control mode is selected (SCTRL=1 in the I2C_CTRL1 register) and RLDEN=1 in the I2C_CTRL2 register, if TCRLD = 1, indicating the completion of the last data transfer, then the TCRLD will be cleared by hardware so as to release the SCL line after a non-zero value is written to the CNT bit of I2C_CTRL2 register.

Clock stretching capability disabled

The SCL clock stretching is disabled when STRETCH=1 in the I2C_CTRL1 register, with the following conditions worth notice:

- Address reception: The SCL clock is not stretched when the address received by slave matches the local address enabled (ADDRF=1 in the I2C_STS register).
- Data reception: If no data is read from the I2C_RXDT register before the next ACK signal, an overflow will occur, and the OUF bit in the I2C_STS register will be set.

- Data transmission: If no data is written to the I2C_TXDT register after the completion of the previous data transfer, an underflow will occur, and the OUF bit in the I2C_STS register will be set.

11.4.1 I²C timing control

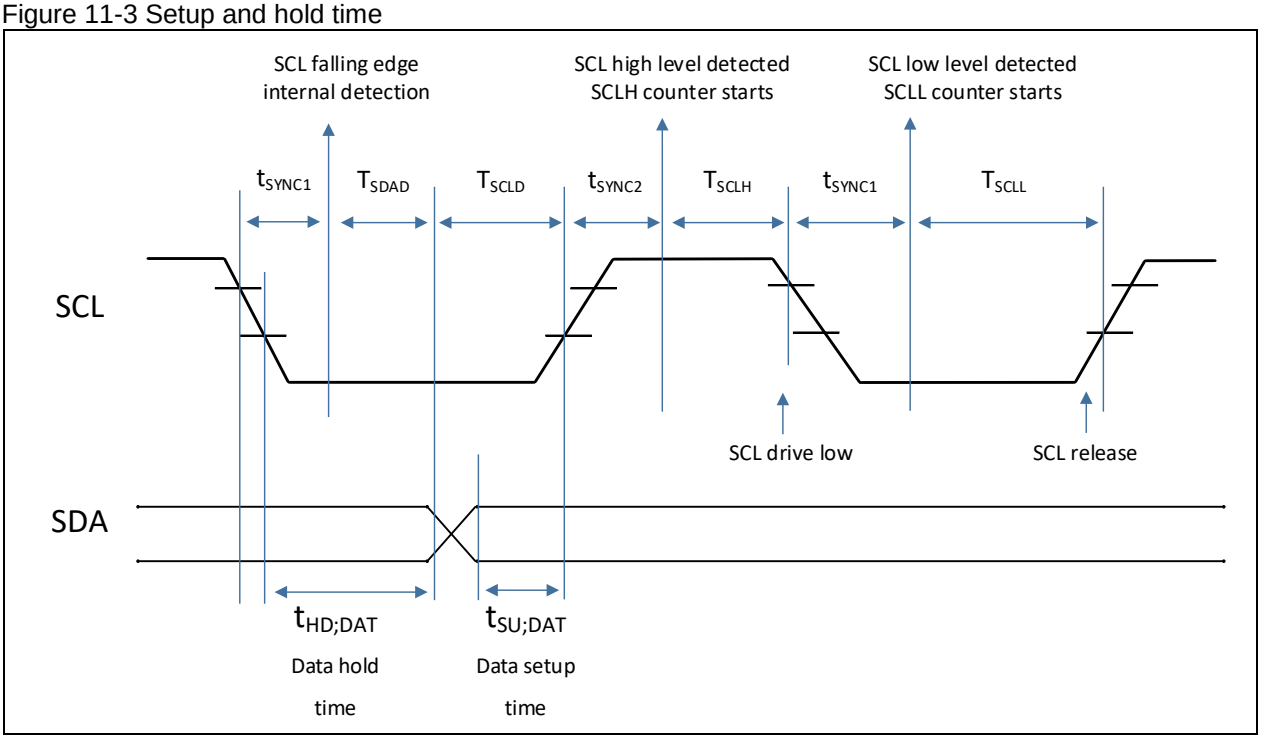
I²C core is clocked by I2C_CLK, whereas the I2C_CLK is clocked by PCLK1. The PCLK1 should be set to be less than 4/3 SCL cycles.

The corresponding bits in the I2C_CLKCTRL register are used for timing configuration.

— DIV[7:0]: I²C clock divider
— SDAD[3:0]: Data hold time ($t_{HD;DAT}$)
— SCLD[3:0]: Data setup time ($t_{SU;DAT}$)
— SCLH[7:0]: SCL high
— SCLL[7:0]: SCL low

Note: Timing configuration cannot be modified once the I²C is enabled.

Figure 11-3 Setup and hold time



It is possible to configure data hold time ($t_{HD;DAT}$) and data setup time ($t_{SU;DAT}$) freely by setting the DIV[7:0], SDAD[3:0] and SCLD[3:0] bits in the I2C_CLKCTRL register.

- Data hold time ($t_{HD;DAT}$): refers to the duration from SCLfalling edge to SDA output

  $t_{HD;DAT} = T_{SDAD} + t_{SYNC}$

  $T_{SDAD} = SDAD \times (DIV + 1) \times T_{I2C\_CLK}$

  $t_{SYNC} = (DFLT + 3) \times T_{I2C\_CLK} - t_f$

  t_{SYNC} consists of three parts:
  — SCL falling edge time t_f
  — Digital filter input latency ($DFLT \times T_{I2C\_CLK}$)
  — Synchronization delay between SCL and I2C_CLK (2~3 I2C_CLK cycles)

- Data setup time ($t_{SU;DAT}$): refers to the duration from SDA output to SCL rising edge

  $t_{SU;DAT} = SCLD \times (DIV+1) \times T_{I2C\_CLK} - t_r$

In master mode, the width of SCL signals (high and low) can be configured freely by setting the DIV[7:0], SCLH[7:0] and SCLL[7:0] bits in the I2C_CLKCTRL register.

SCL low: When the SCL low signal is detected, the internal SCLL counter starts counting until it reaches

the SCLL value. At this point, the SCL line is released and becomes high.

SCL high: When the SCL high signal is detected, the internal SCLH counter starts counting. When the counter value reaches the SCLH value, the SCL line is pulled low. In the process of SCL remaining high, if it is pulled low by external bus, the internal SCLH counter will stop counting and start counting in SCL low mode, laying the foundation for clock synchronization.

- SCL high signal width:

  $t_{HIGH} = T_{SCLH} = (SCLH + 1) \times (DIV + 1) \times T_{I2C\_CLK}$
- SCL low signal width:

  $t_{LOW} = T_{SCLL} = (SCLL + 1) \times (DIV + 1) \times T_{I2C\_CLK}$

Table 11-1 I²C timing specifications

Parameter		Standard mode		Fast mode		Fast mode plus		SMBus	
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
f_{SCL} (kHz)	SCL clock frequency		100		400		1000		100
t_{LOW} (us)	SCL clock low	4.7		1.3		0.5		4.7	
t_{HIGH} (us)	SCL clock high	4.0		0.6		0.26		4.0	50
$t_{HD;DAT}$ (us)	Data hold time	0		0	0.9	0	0.45	300	
$t_{SU;DAT}$ (ns)	Data setup time	250		100		50		250	
tr (ns)	SCL/SDA rising edge		1000		300		120		1000
tf (ns)	SCL/SDA falling edge		300		300		120		300

11.4.2 Data transfer management

Data transfer counter is available in the I²C interface to control communication flow. It is mainly used for:

- NACK transmission: master reception mode
- STOP transmission: master reception/ transmission modes
- RESTART generation: master reception/ transmission modes
- ACK control: slave mode (SMBus)
- PEC transmission/reception: master/slave modes

Generally, the data transfer management counter (by setting the CNT[7:0] bit in the I2C_CTRL2 register) is applicable to master mode, and it is disabled in slave mode. This counter is used only in SMBus mode for the ACK control and PEC reception of each byte by the slave. In SMBus mode, the slave enables data counter with the SCTRL bit in the I2C_CTRL1 register.

Byte control through master

The CNT[7:0] bit in the I2C_CTRL2 register is used to configure the number of bytes to be transferred, ranging from 1 to 255. If the number of data to be transferred is greater than 255, then the RLDEN bit in the I2C_CTRL2 register has to be set to 1 to enable reload mode. The following configuration processes are described in two aspects:

- ≤255 bytes, for example, the number of data to be transferred is 100 bytes
  - Step 1: Disable reload mode by setting RLDEN=0;
  - Step 2: Set CNT[7:0]=100.
- >255 bytes, for example, the number of data to be transferred is 600 bytes
  - Step 1: Enable reload mode by setting RLDEN=1;
  - Step 2: Set CNT[7:0]=255, then the remaining bytes are 600-255=345;
  - Step 3: After the completion of 255-byte data transfer, the TCRLD=1 in the I2C_STS register, and then set CNT[7:0]=255 for continuous transfer, and the remaining bytes are 345-255=90;
  - Step 4: After the completion of the second 255-byte data transfer, the TCRLD=1 in the I2C_STS register, and then set RLDEN=0 to disable reload mode before setting CNT[7:0]=90 for continuous transfer.

There are two ways to stop the last data transfer (RLDEN=0, reload mode disabled):

- Stop data transfer automatically (ASTOPEN=1 in the I2C_CTRL2 register)
  - When the number of data programmed in the CNT[7:0] bit has been fully transferred, the master will automatically send a STOP condition.
- Stop data transfer by software (ASTOPEN=0 in the I2C_CTRL2 register)
  - When the number of data programmed in the CNT[7:0]bit has been fully transferred, the TDC=1 in the I2C_STS register, the SCL will be pulled low at this point, and an interrupt is generated if the TDCIEN is enabled. In this case, it is possible to send a STOP condition by setting GENSTOP=1 in the I2C_CTRL2 register, or send a RESTART condition by setting GENSTART=1 in the I2C_CTRL2 register, before clearing TDC flag by software.

Byte control through slave

This feature is enabled by setting the SCTRL bit in the I2C_CTRL1 register so that the slave is able to control ACK/NACK of each byte independently.

- Proceed as below:
  - Set SCTRL=1 to enable Byte Control Through Slave.
  - After the slave address is matched (ADDRF=1), enable reload mode by setting RLDEN=1, and then set CNT[7:0]=1.
  - When a byte is received, the TCRLD=1 in the I2C_STS register, and the slave will pull the SCL bus low between the 8th and 9th clock edges. At this point, the user can read the RXDT register and generate an ACK or NACK signal through the NACKEN bit in the I2C_CTRL2 register.
  - When an NACK signal is generated, it indicates the end of communication.
  - When an ACK signal is generated, the communication flow keeps going on. At this point, set CNT[7:0]=1, the TCRLD flag is cleared automatically by hardware, and the SCL bus is released for the reception of the next byte.

As we know, the value in the CNT[7:0] bit is not limited to 1. If you want to receive 8 data, for example, but just want to control the ACK/NACK signals of the 8th data, proceed as below: set CNT[7:0]=8, the slave will receive 7 consecutive data, with ACK signals sent. Once the 8th data reception is completed, the SCL us is pulled low, and then proceed as above to select whether to send an ACK or NACK.

It should be noted that the clock stretching capability must be enabled (STRETCH =0 in the I2C_CTRL1 register) before selecting Byte Control Mode Through Slave.

Table 11-2 I^{2}C configuration

Description	RLDEN	ASTOPEN	SCTRL
Master transmit/receive RESTART	0	0	×
Master transmit/receiveSTOP	0	1	×
Slave receive (control ACK/NACK of each byte)	1	×	1
Slave transmit/receive (ACK response to all bytes)	×	×	0

11.4.3 I^{2}C master communication flow

1. **I^{2}C clock initialization (by setting the I2C_CLKCTRL register)**
   - I^{2}C clock divider: DIV[7:0]
   - Data hold time ($t_{HD;DAT}$): SDAD[3:0]
   - Data setup time ($t_{SU;DAT}$): SCLD[3:0]
   - SCL high duration: SCLH[7:0]
   - SCL low duration: SCLL[7:0]

   The register can be configured by means of Artery_I2C_Timing_Configuration tool.

2. **Set the number of bytes to be transferred**
   - ≤255 bytes

     Set RLDEN=0 in the I2C_CTRL2 register to disable reload mode;

     Set CNT[7:0]=N in the I2C_CTRL2 register.

- >255 bytes

  Set RLDEN=1 in the I2C_CTRL2 register to enable reload mode;

  Set CNT[7:0]=255 in the I2C_CTRL2 register;

  Remaining bytes N=N-255.

3. End of data transfer

- ASTOPEN=0: stop data transfer by software. After the completion of data transfer, the TDC is set in the I2C_STS register, and GENSTOP=1 or GENSTART=1 is written by software to send a STOP or START condition.
- ASTOPEN=1: data transfer is stopped automatically. A STOP condition is sent at the end of data transfer.

4. Set slave address

- Set slave address value by setting the SADDR bit in the I2C_CTRL2 register.
- Set slave address mode by setting the ADDR10 bit in the I2C_CTRL2 register.

  ADDR10=0: 7-bit address mode

  ADDR10=1: 10-bit address mode

5. Set transfer direction (by setting the DIR bit in the I2C_CTRL2 register)

- DIR=0: Master reception
- DIR=1: Master transmission

6. Start data transfer

When GENSTART=1 in the I2C_CTRL2 register, the master starts sending a START condition and slave address. After receiving the ACK from the slave, ADDRF=1 is asserted in the I2C_STS register. The ADDRF flag can be cleared by setting ADDRC=1 in the I2C_CLR register, and then data transfer starts.

7. Master transmit

1. I2C_TXDT data register is empty, the shift register is empty, and TDIS=1 in the I2C_STS register;
2. Write 1 to the TXDT register, and data is immediately moved to the shift register;
3. TXDT register becomes empty, and TDIS=1 again;
4. Write 2 to the TXDT register, and TDIS is cleared;
5. Repeat steps 2 and 3 until the data in the CNT[7:0] is sent;
6. If TCRLD=1 (reload mode) in the I2C_STS register, the following two circumstances should be noted:

   Remaining bytes N>255: Writing 255 to the CNT bit, N=N-255, TCRLD is cleared, and data transfer continues;

   Remaining bytes N≤255: Disable reload mode (RLDEN=0), write N to the CNT bit, TCRLD is cleared, and data transfer continues.

8. Master receive

1. After the slave address is matched, ADDRF=1 in the I2C_STS register, clear the ADDRF flag by setting ADDRC=1 in the I2C_CLR register, and then it starts sending data;
2. After the reception of data, RDBF=1; read the RXDT register will clear the RDBF automatically;
3. Repeat step 2 until the data in the CNT[7:0] is received;
4. If TCRLD=1 (reload mode) in the I2C_STS register, the following two circumstances should be noted:

   Remaining bytes N>255: Writing 255 to the CNT bit, N=N-255, TCRLD is cleared, and data transfer continues;

   Remaining bytes N≤255: Disable reload mode (RLDEN=0), write N to the CNT bit,TCRLD is cleared, and data transfer continues.
5. After the reception of the last data, an NACK signal will be sent by master.

9. Stop condition

- STOP condition generation:

  ASTOPEN=0: TDC=1 in the I2C_STS register, set GENSTOP=1 to generate a STOP condition

  ASTOPEN=1: A STOP condition is generated automatically
- Wait for the generation of a STOP condition. When a STOP condition is generated, STOPF=1

is asserted in the I2C_STS register. The STOPF flag can be cleared by setting STOPC=1 in the I2C_CLR register, and then transfer stops.

When the master receives an NACK signal during transmission, then the ACKFAIL is set in the I2C_STS register, and a STOP condition is sent to stop communication, whatever mode (either ASTOPEN=0 or ASTOPEN=1).

Master transmitter

Figure 11-4 I²C master transmission flow

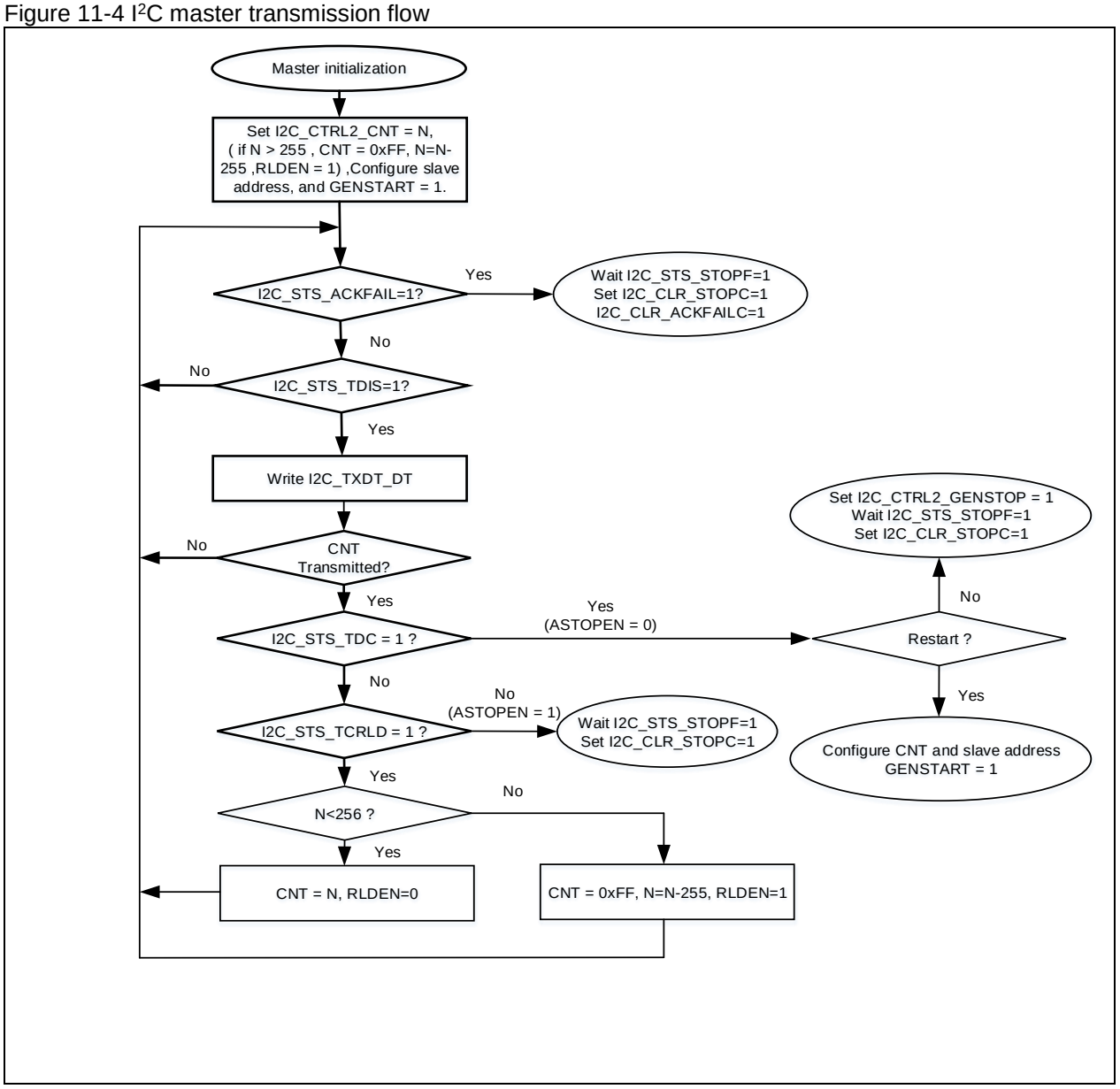
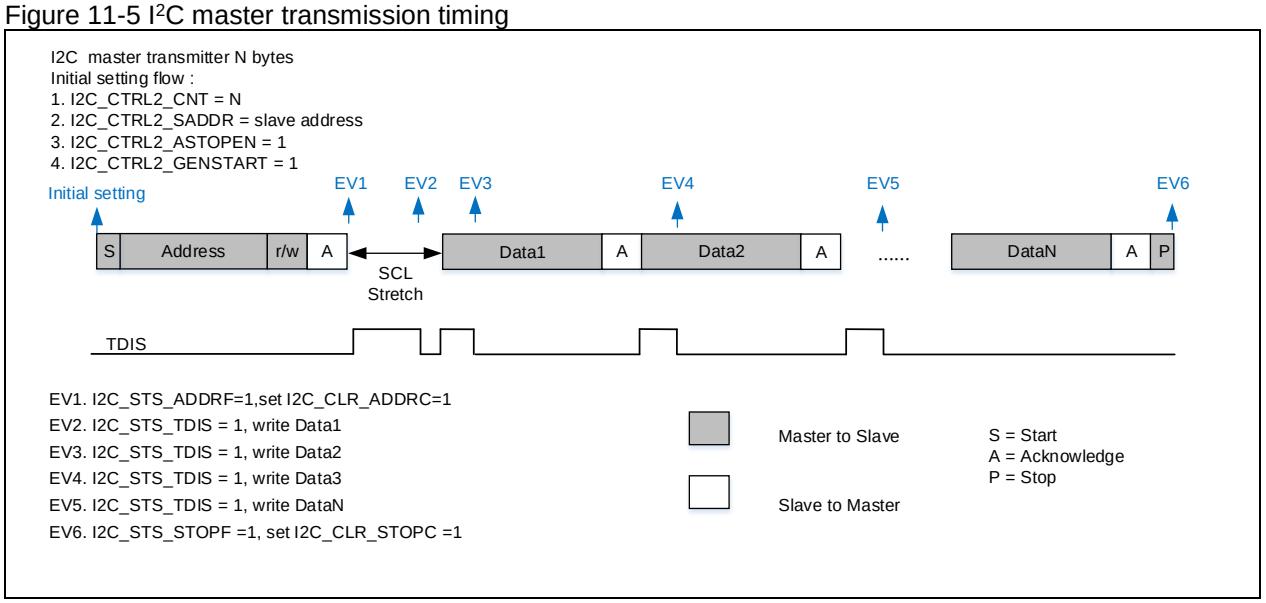


Figure 11-5 I²C master transmission timing



Master receiver

Figure 11-6 I²C master receive flow

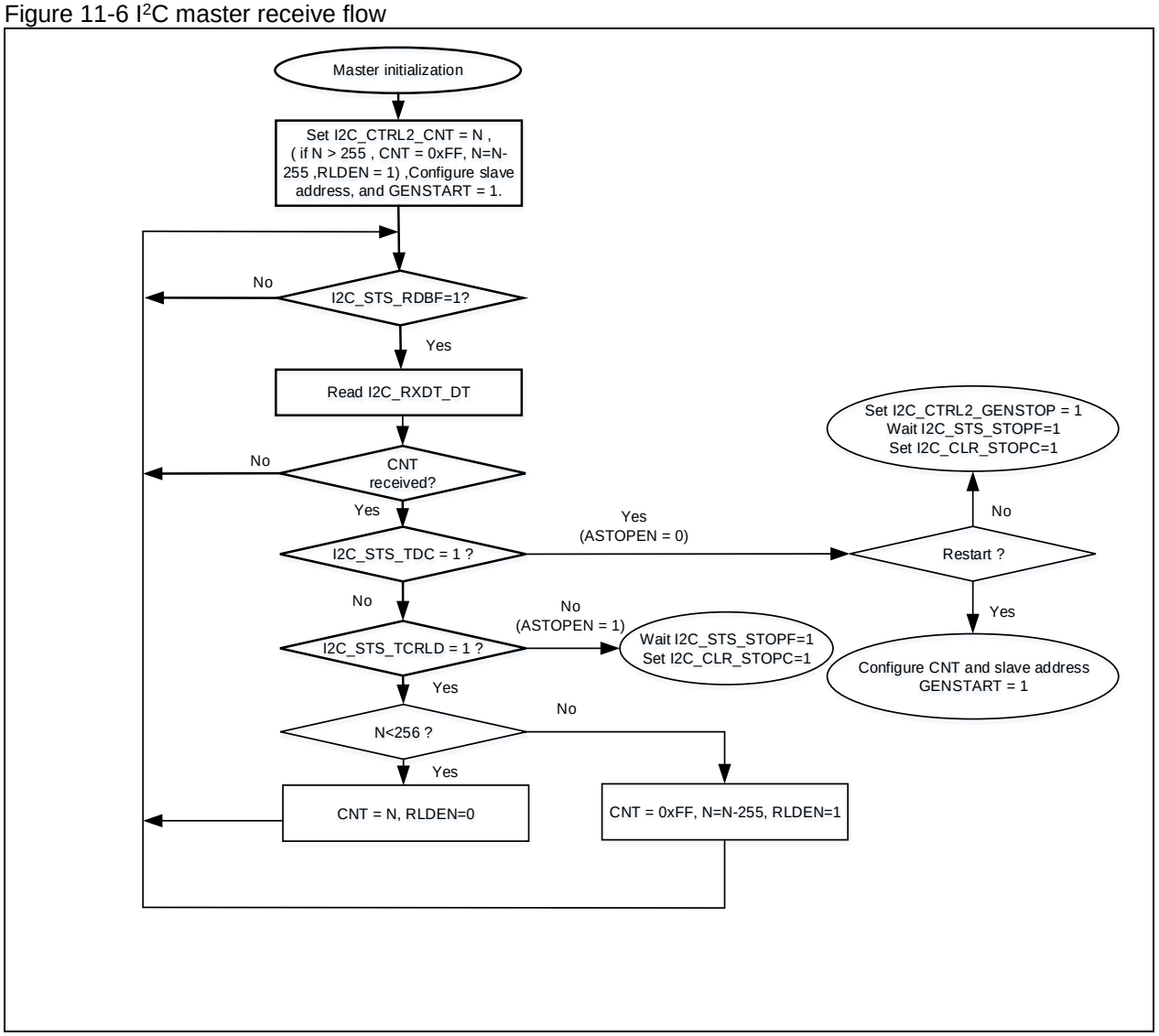
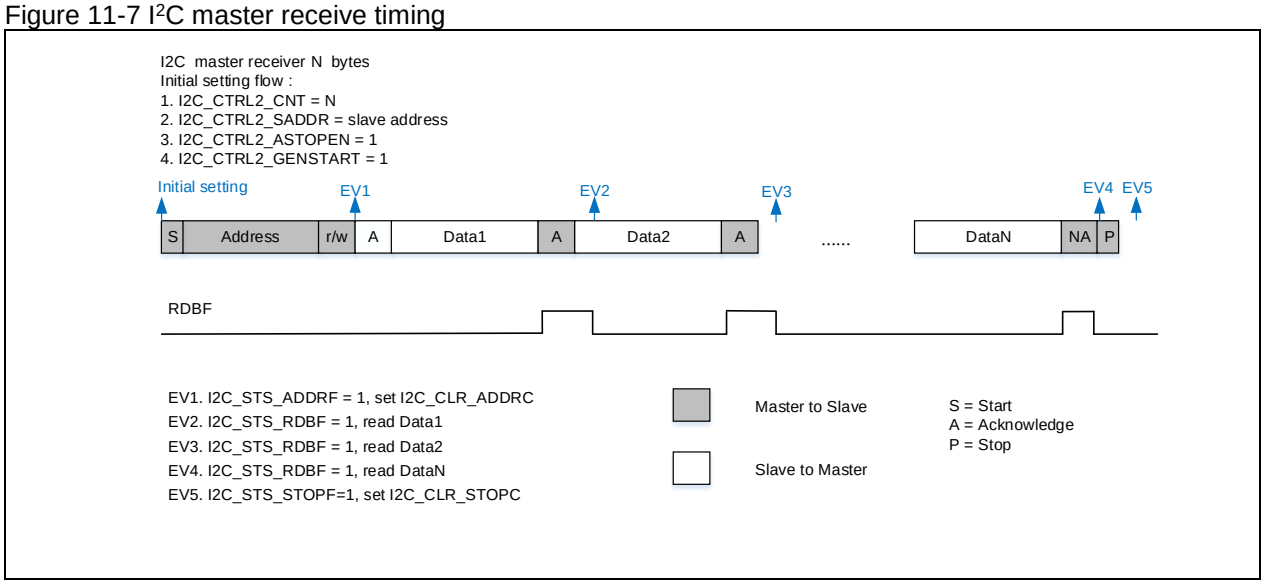


Figure 11-7 I²C master receive timing



Master special transfer sequence

In 10-bit addressing mode, the READH10 bit in the I2C_CTRL2 register is used to generate a special timing. When READH10=1, the master sends data to the slave before read access to the slave, as shown in the figure below.

Operating method:

When ASTOPEN = 0, data is transferred from the master to the slave. At the end of data transfer, READH10=0 is asserted, and then the master starts receiving data from the slave.

Figure 11-8 10-bit address read access when READH10=1

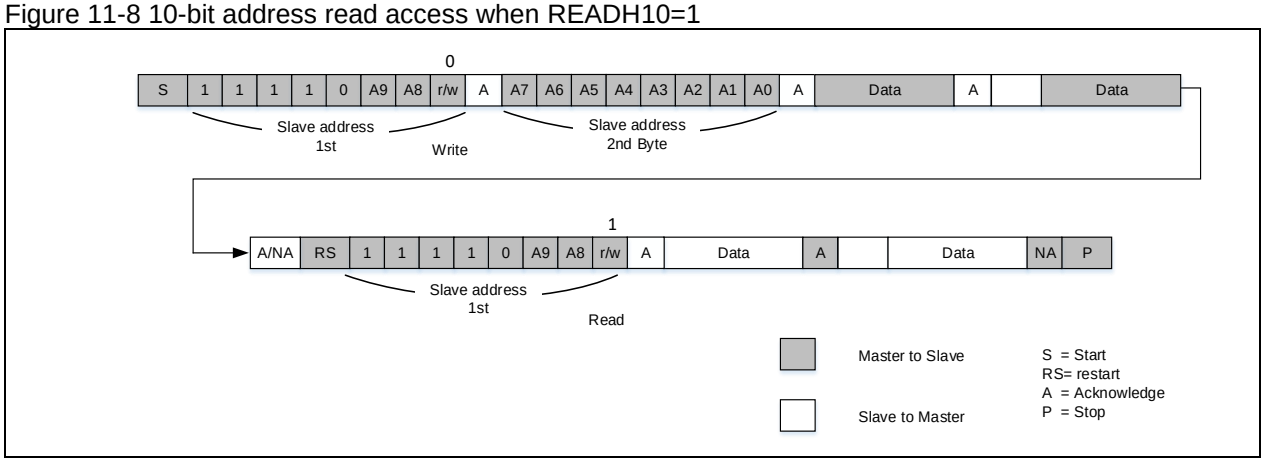
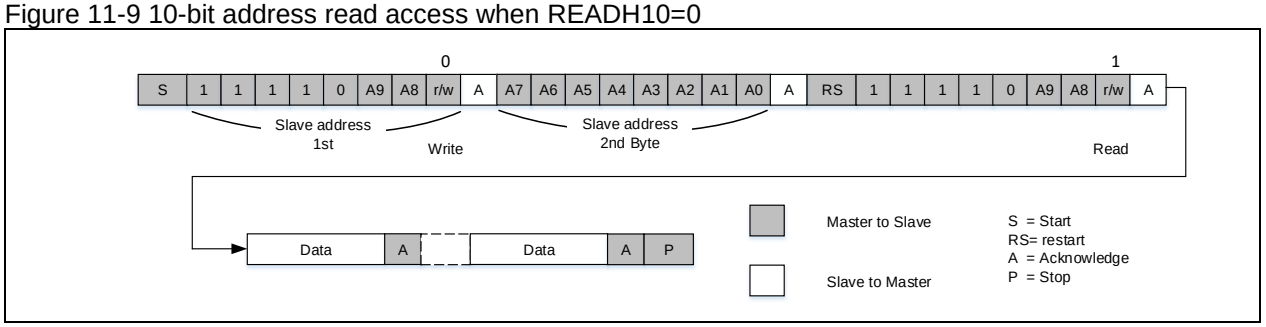


Figure 11-9 10-bit address read access when READH10=0



11.4.4 I²C slave communication flow

1. **I²C clock initialization (by setting the I2C_CLKCTRL register)**
   - I²C clock divider: DIV[7:0]
   - Data hold time ($t_{HD;DAT}$): SDAD[3:0]
   - Data setup time ($t_{SU;DAT}$): SCLD[3:0]

   The register can be configured by means of Artery_I2C_Timing_Configuration tool.

2. **Set local address 1**
   - Set address mode:

     7-bit address: by setting ADDR1MODE = 0 in the I2C_OADDR1 register

     10-bit address: by setting ADDR1MODE = 1 in the I2C_OADDR1 register
   - Set address 1: by setting the ADDR1 bit in the I2C_OADDR1 register
   - Enable address 1: by setting ADDR1EN=1 in the I2C_OADDR1 register

3. **Set local address 2**
   - Set address 2: by setting the ADDR2 bit in the I2C_OADDR2 register
   - Set address 2 mask bit: by setting the ADDR2MASK bit in the I2C_OADDR2 register
   - Enable address 2: by setting ADDR2EN=1 in the I2C_OADDR2 register

   Note: Only 7-bit address mode is available in the address 2 mode. The ADDR2MASK bit is used to mask some address bits freely so that the slave can respond to some specific addresses. Refer to Section 14.2 for more information about the ADDR2MASK bit.

   In the case of using only one address, only address 1 needs to be configured, without the need of address 2 mode.

4. Wait for address matching

When the local address is received, the ADDRF bit is set in the I2C_STS register. The data transfer direction can be obtained by read access to the SDIR bit in the I2C_STS register. When SDIR=0, it indicates that the slave is receiving data, whereas SDIR=1 indicates that the slave is sending data. The ADDR[6:0] bit in the I2C_STS register indicates what kind of address has been received, which is particularly helpful in the case when the dual address mode is used and the address 2 mode mask bit is set.

Data transfer starts when the ADDRF flag is cleared by setting ADDRC=1 in the I2C_CLR register.

5. Data transfer (slave transmission, clock stretching enabled, STRETCH=0)

After address matching:

1. I2C_TXDT data register becomes empty, the shift register becomes empty, and TDIS=1 in the I2C_STS register;
2. Write 1 to the TXDT register, and data is immediately moved to the shift register;
3. TXDTT register becomes empty, and TDIS=1 again;
4. Write 2 to the TXDT register, and the TDIS is cleared;
5. Repeat steps 3 and 4 until the end of data transfer;
6. Wait for the generation of an NACK signal. Once received, the ACKFAILF is set in the I2C_STS register. The ACKFAILF flag is cleared by setting ACKFAILC=1 in the I2C_CLR register.
7. Wait for the generation of a STOP condition. Once received, the STOPF bit is set in the I2C_STS register. The STOPF flag is cleared by setting STOPC=1 in the I2C_CLR register, and transmission ends.

Note: In the case of the clock stretching being disabled (STRETCH=1), if data has not yet been written to the TXDT register before the transmission of the first bit of the to-be-transferred data (that is, before the generation of SDA edge), an underrun error may occur, and the OUF bit is set in the I2C_STS register, sending 0xFF to the bus.

In order to write data in time, data must be written to the DT register first before communication, in two different ways:

- Write operation through software: Clear the TXDT register by setting TDBE=1 through software; then write the first data to the TXDT register, and the TDBE bit is cleared.
- Write operation through interrupts or DMA: Clear the TXDT register by setting TDBE=1 through software; then set TDIS=1 to generate a TDIS event, which generates an interrupt or DMA request. At this point, data is written to the TXDT register using DMA or interrupt functions.

6. Data transfer (slave receive, clock stretching enabled, STRETCH=0)

After address matching:

1. I2C_RXDT register becomes empty, the shift register becomes empty, and RDBF=0 in the I2C_STS register;
2. After the reception of data, RDBF=1; read the RXDT register will clear the RDBF automatically;
3. Repeat step 2 until the completion of all data transfer;
4. Wait for the generation of a STOP condition. Once received, the STOPF bit in the I2C_STS register is set. The STOPF flag is cleared by setting STOPC=1 in the I2C_CLR register, and transmission ends.

In slave receive mode, the slave byte control mode (SCTRL=1) can be used for data reception. This mode allows to control ACK/NACK signals of each byte received. This mode is typically available in SMBus protocol. Refer to 11.4.2 Data transfer management for more information about this mode.

Note that the slave must read the received data in the case of the clock stretching being disabled (STRETCH=1). If one-byte data has been received and data is not read yet before the end of the next data reception, an overrun error occurs, setting the OUF bit in the I2C_STS register and sending NCAK.

An interrupt will be generated if the corresponding interrupt enable bit is enabled. For more information about interrupt generation, refer to the interrupt chapter.

Slave transmitter

Figure 11-10 I²C slave transmission flow

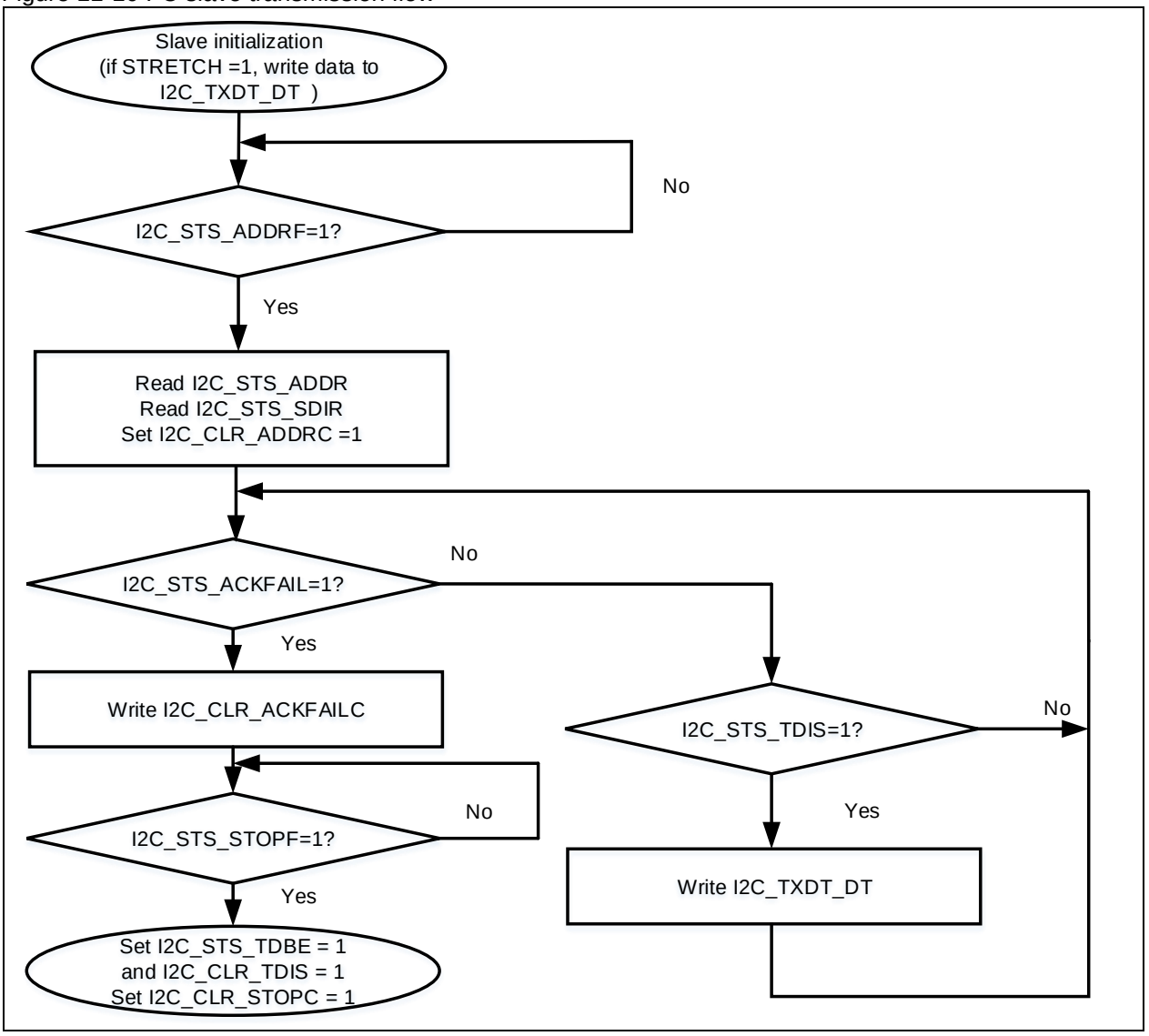
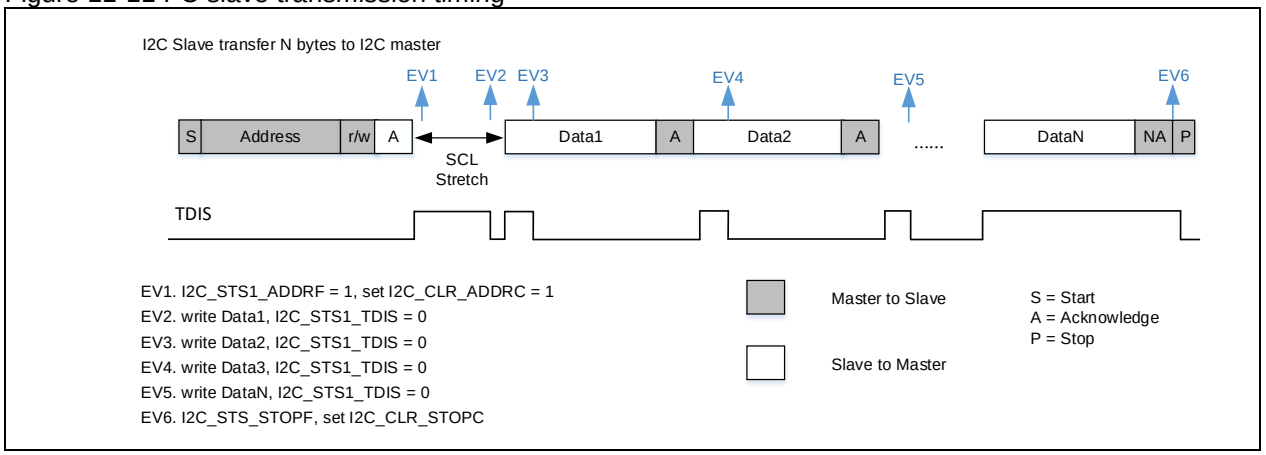


Figure 11-11 I²C slave transmission timing



Slave receiver

Figure 11-12 I²C slave receive flow

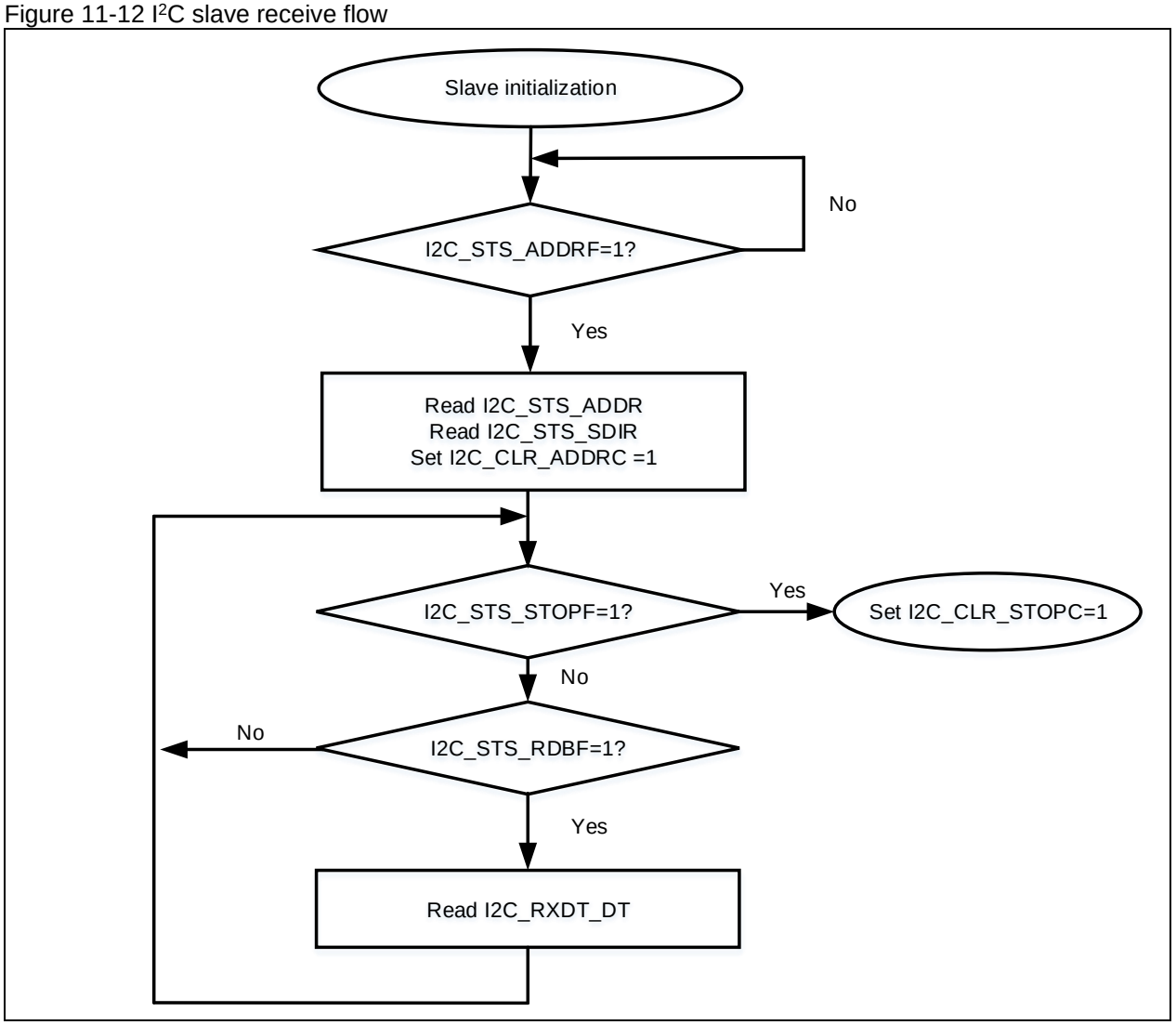
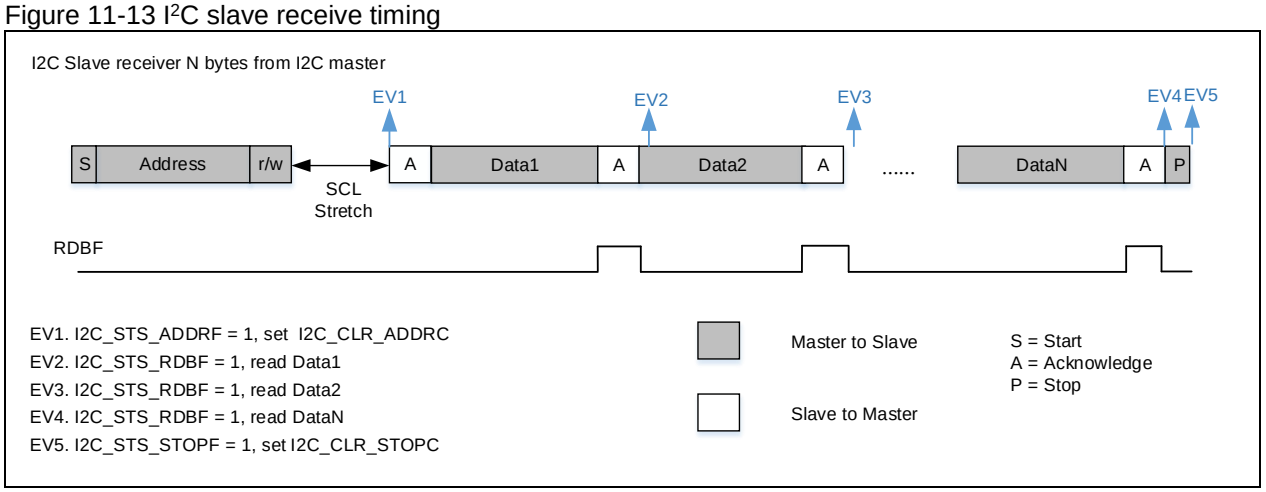


Figure 11-13 I²C slave receive timing



11.4.5 SMBus

The System Management Bus (SMBus) is a two-wire interface through which various devices can communicate with each other. It is based on I²C. With SMBus, the device can provide manufacturer information, tell the system its model/part number, report different types of errors and accept control parameters and so on. For more information, refer to SMBus 2.0 protocol.

Difference between SMBus and I²C

1. SMBus requires a minimum speed of 10 KHz for the purpose of management and monitor. It is quite easy to know whether the bus is in Idle state or not as long as a parameter is input while running on a certain transmission speed, without the need of detecting the STOP signals one after another, or even keeping STOP and other parameter monitor. I²C does not support such function.
2. SMBus transmission speed ranges from 10 KHz to100 KHz. I²C has no minimum requirement, and its maximum speed varies from one mode to another, namely, 100 KHz in standard mode and 400 KHz in fast mode.
3. After reset, SMBus needs timeout, but there is no limit for I²C in this regard.

SMBus address resolution protocol (ARP)

ARP address conflicts can be resolved by dynamically assigning a new unique address to each device. Refer to SMBus 2.0 protocol for more information about ARP.

Setting the DEVADDREN bit in the I2C_CTRL1 register can enable the I²C interface to recognize the default device address (0b1100001x). However, unique device identifier (UDID) and the detailed protocol implementation should be handled by software.

SMBus host notify protocol

The slave device can send data to the master device through SMBus host notify protocol. For example, the slave can notify the host to implement ARP with this protocol. Refer to SMBus 2.0 protocol for details on SMBus host notify protocol.

In host mode (HADDREN=1), the I²C interface is enabled to recognize the default host address (0b0001000x).

SMBus Alert

SMBALERT is an optional signal that connects the ALERT pin between the host and the salve. With this signal, the slave notifies the host to access the slave. SMBALERT is a wired-AND signal. For more information about SMBus Alert, refer to SMBus2.0 protocol.

The detailed sequences are as follows:

SMBus host

1. Enable SMBus Alert mode by setting SMBALERT=1;
2. Enable ALERT interrupt if necessary;
3. When an alert event occurs on the ALERT pin (ALERT pin changes from high to low);
4. The host will generate ALERT interrupt if enabled;
5. The host then processes the interrupt and accesses to all devices through ARA (Alert Response Address) (0001100x) so as to get the slave addresses. Only the devices with pulled-down SMBALERT can acknowledge ARA.
6. The host then continues to operate based on the slave addresses available.

SMBus slave

1. When an alert event occurs and the ALERT pin changes from high to low (SMBALERT=1), the slave responses to ARA (Alert Response Address) (0001100x);
2. Wait until the host gets the slave addresses through ARA;
3. Report its own address, but it continues to wait if the arbitration is lost;
4. Address is reported properly, and the ALERT pin is released (SMBALERT=0).

Packet error checking (PEC)

Packet error checking (PEC) is used to guarantee the correctness and integrity of data transfer. This is done by using CRC-8 polynomial:

$C(x) = x^8 + x^2 + x + 1$

PEC calculation is enabled when PECEN=1 in the I2C_CTRL1 register to check address and data.

PEC transfer is enabled when PECTEN=1 in the I2C_CTRL2 register. If the data to be transferred reaches N-1 (CNT=N):

- Host sends a PEC automatically;
- Slave considers the Nth data as a PEC and check it. A NACK will be sent is the PEC checking result is incorrect, and the PECERR will be set in the I2C_STS register. In case of slave transmission mode, a NACK must follow the PEC whatever the checking result.

SMBus timeout

The SMBus protocol specifies three timeout detection modes:

- Low level timeout ($T_{TIMEOUT}$): The time duration when the SCL is kept low in a single mode (taking into account master/slave device, however actively or passively pulled low).
- Cumulative timeout for a slave device at low level ($T_{LOW:SEXT}$): The cumulative time duration when the SCL is pulled low by a slave device during the period from a START condition to a STOP condition.
- Cumulative timeout for a master device at low level ($T_{LOW:MEXT}$): The cumulative time duration when the SCL is pulled low by a master device during the period from the ACK of the last byte to the 8th bit of the next byte (a single byte).

It should be noted that both $T_{LOW:SEXT}$ and $T_{LOW:MEXT}$ only deal with the time when they set themselves low level, excluding the time when they are pulled low by external sources. In contrast, both of these cases are considered in the calculation of $T_{TIMEOUT}$.

Table 11-3 SMBus timeout specification

Type of timeout	Min.	Max.	Unit
$T_{TIMEOUT}$	25	35	ms
$T_{LOW:SEXT}$	-	25	ms
$T_{LOW:MEXT}$	-	10	ms

The I^2C peripherals embeds two counters for timeout detection, which can be configured through the I2C_TIMEOUT register. When a timeout event occurs, the TMOUT is set in the I2C_STS register. The TMOUT bit can be cleared by setting TMOUTC=1 in the I2C_CLR register.

- EXTTIME: This is used to the cumulative timeout detection for master/slave devices at low level.

  Timeout detection = (EXTTIME + 1) x 2048 x T_{I2C_CLK}
- TOTIME: This is used for clock level timeout detection, selected through the TOMODE bit.

  TOMODE=0: Low level timeout detection, timeout duration=(TOTIME + 1) x 2048 x T_{I2C_CLK}

  TOMODE=1: High level timeout detection, timeout duration=(TOTIME + 1) x 4 x T_{I2C_CLK}

Table 11-4 SMBus timeout detection configuration

Type of timeou	Other configuratio	Enable bit	Timeout calculation
$T_{TIMEOUT}$	TOMODE=0	TOEN=1	(TOTIME + 1) x 2048 x T_{I2C_CLK}
$T_{LOW:SEXT}$	-	EXTEN=1	(EXTTIME + 1) x 2048 x T_{I2C_CL}
$T_{LOW:MEXT}$	-	EXTEN=1	(EXTTIME + 1) x 2048 x T_{I2C_CL}

Slave receive byte control

In slave receive mode, the slave receive byte control mode (SCTRL=1) can be used to control ACK/NACK signals of each received byte. Refer to the 11.4.2 Data transfer management for more information.

Table 11-5 SMBus mode configuration

mode	PECEN	PECTEN	RLDEN	ASTOPEN	SCTRL
Master transmit/receive +STOP	1	1	0	1	-
Master transmit/receive +RESTART	1	1	0	0	-
Slave receive	1	1	1	-	1
Slave transmit	1	1	0	-	-

How to use this interface in SMBus mode

1. Set SMBus default address acknowledgement:

   HADDREN=1: Master default address acknowledged (0b0001000x)

   DEVADDREN=1: Device default address acknowledged (0b1100001x)
2. Configure PEC
3. Slave receive byte control mode can be enabled (with the SCTRL bit in the I2C_CTRL1 register) in slave mode, if necessary
4. Other configurations follow the I^2C

However, the detailed SMBus protocol implementation should be handled by software, since the I^2C interface is only enabled to recognize the addresses of SMBus protocols.

11.4.6 SMBus master communication flow

The SMBus is similar to the I^2C in terms of master communication flow.

1. I^2C clock initialization (by setting the I2C_CLKCTRL register)
- I^2C clock divider: DIV[7:0]
- Data hold time ($t_{HD;DAT}$): SDAD[3:0]
- Data setup time ($t_{SU;DAT}$): SCLD[3:0]
- SCL high duration: SCLH[7:0]
- SCL low duration: SCLL[7:0]

The register can be configured by means of Artery_I2C_Timing_Configuration tool.

2. SMBus-related initialization
- Select SMBus host: host default address acknowledged (0b0001000x) by setting HADDREN=1
- Enable PEC calculation: set PECEN=1 in the I2C_CTRL1 register
- Enable PEC transfer: set PECTEN=1 in the I2C_CTRL2 register

3. Set the number of bytes to be transferred
- The number of bytes to be transferred in SMBus mode is <255 at one time. Set RLDEN=0 and CNT[7:0]=N in the I2C_CTRL2 register.

4. End of data transfer
- ASTOPEN=0: stop data transfer by software. After the completion of data transfer, the TDC is set in the I2C_STS register, and GENSTOP=1 or GENSTART=1 is written by software to send a STOP or START condition.
- ASTOPEN=1: data transfer is stopped automatically. A STOP condition is sent at the end of data transfer.

5. Set slave address
- Set slave address (by setting the SADDR bit in the I2C_CTRL2 register)
- Set 7-bit slave address mode (by setting the ADDR10=0 bit in the I2C_CTRL2 register)

6. Set transfer direction (by setting the DIR bit in the I2C_CTRL2 register)
- DIR=0: Master reception
- DIR=1: Master transmission

7. Start data transfer

Set GENSTART=1 in the I2C_CTRL2 register, and the master starts sending a START condition and slave address. After receiving the ACK from the slave, ADDRF=1 is asserted in the I2C_STS register. The ADDRF flag is cleared by setting ADDRC=1 in the I2C_CLR register, and then data transfer starts.

8. Master transmit
1. I2C_TXDT register is empty, the shift register is empty, and TDIS=1 in I2C_STS register;
2. Write 1 to the TXDT register, and data is immediately moved to the shift register;
3. TXDT register is empty, and TDIS=1 again;
4. Write 2 to the TXDT register, and TDIS is cleared;
5. Repeat steps 2 and 3 until the specified data (N-1) is sent;
6. The master will sutomatically transmit the Nth data, that is, PEC.

9. Master recevice

1. After the reception of data, RDBF=1; read the RXDT register will clear the RDBF automatically;
2. Repeat step 1 until the reception of the specified data (N). The Nth data is set as PEC. A NACK is automatically sent after the reception of the Nth data (PEC) whatever the PEC result.

10. STOP condition

- STOP condition generation:

  ASTOPEN=0: TDC=1 in the I2C_STS register, and set GENSTOP=1 to generate a STOP condition;

  ASTOPEN=1: TDC=1 in the I2C_STS register, and the hardware generates a STOP condition automatically.
- Wait for the generation of a STOP condition. When a STOP condition is generated, STOPF=1 is asserted in the I2C_STS register. The STOPF flag can be cleared by setting STOPC=1 in the I2C_CLR register, and then transfer stops.

SMBus master transmitter

Figure 11-14 SMBus master transmission flow

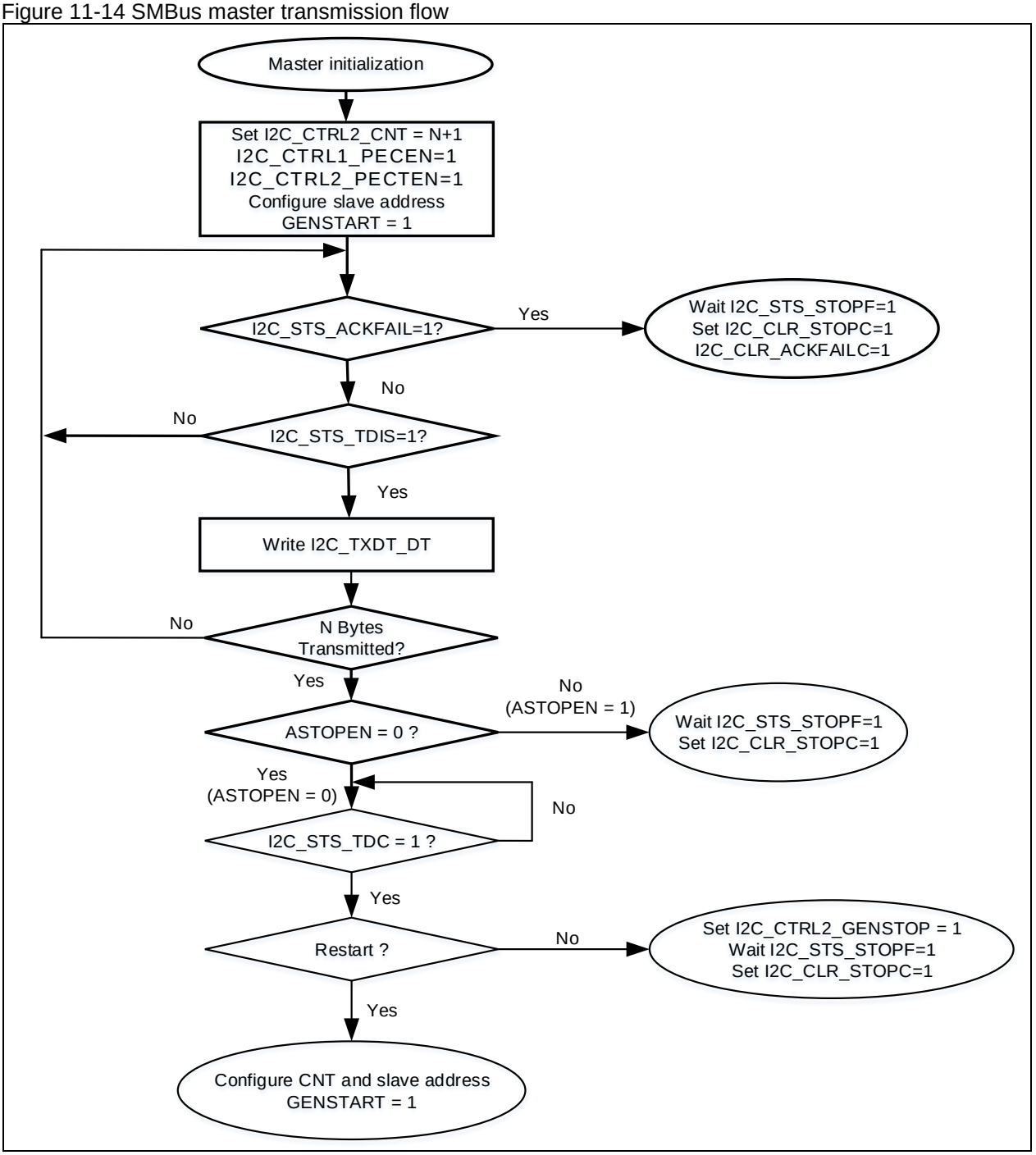
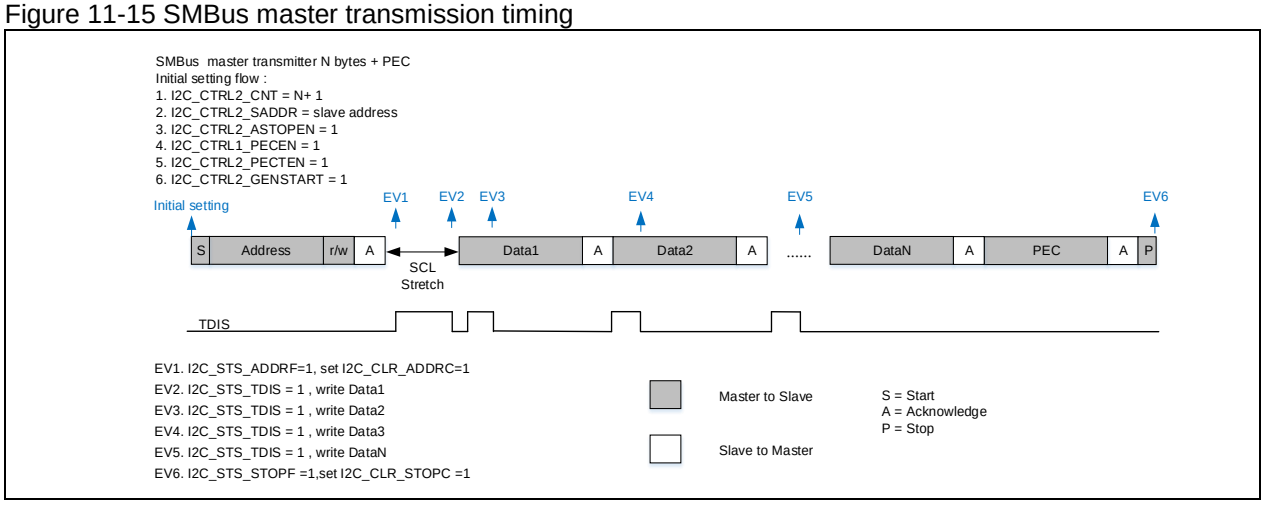


Figure 11-15 SMBus master transmission timing



SMBus master receiver

Figure 11-16 SMBus master receive flow

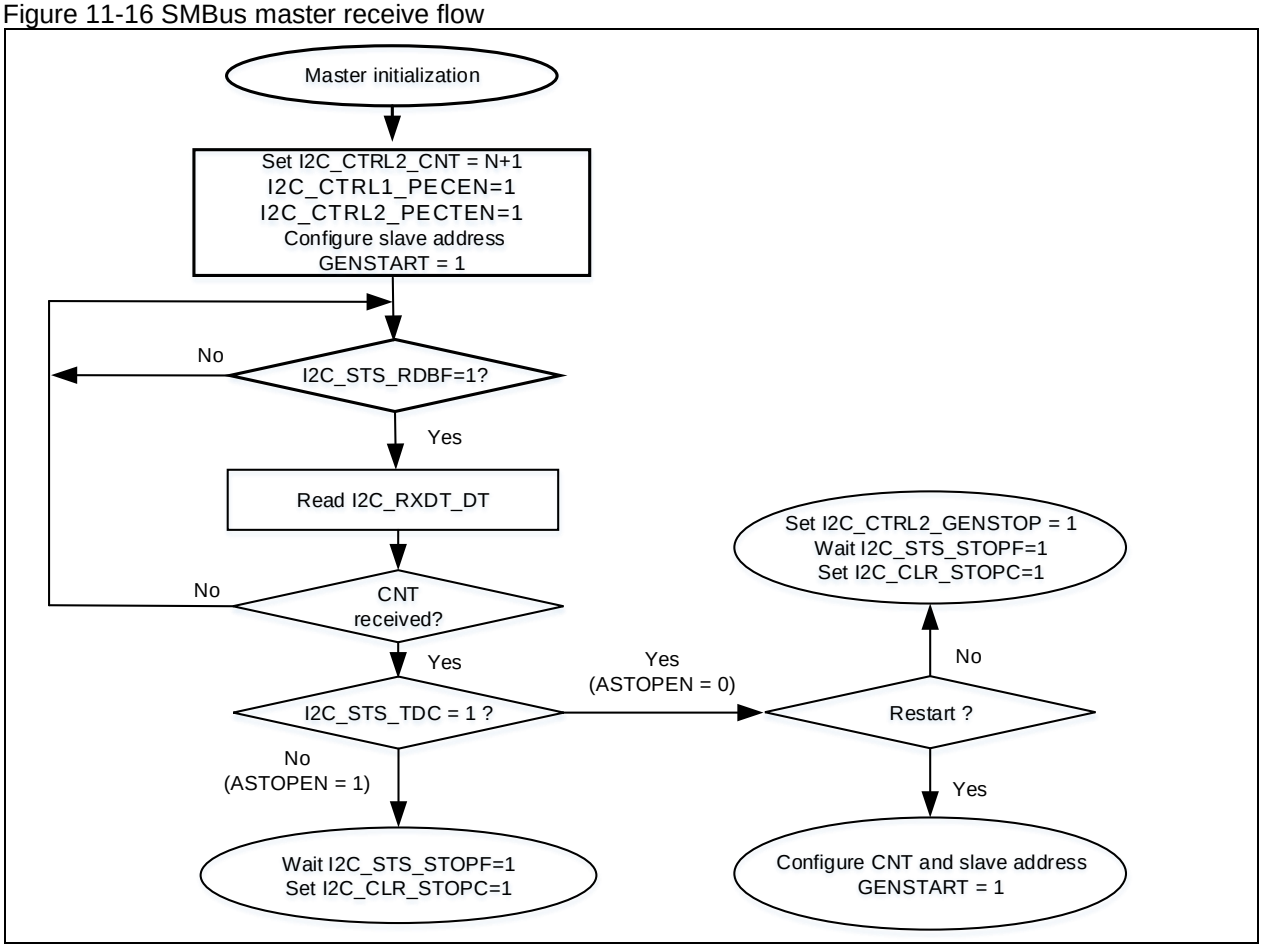
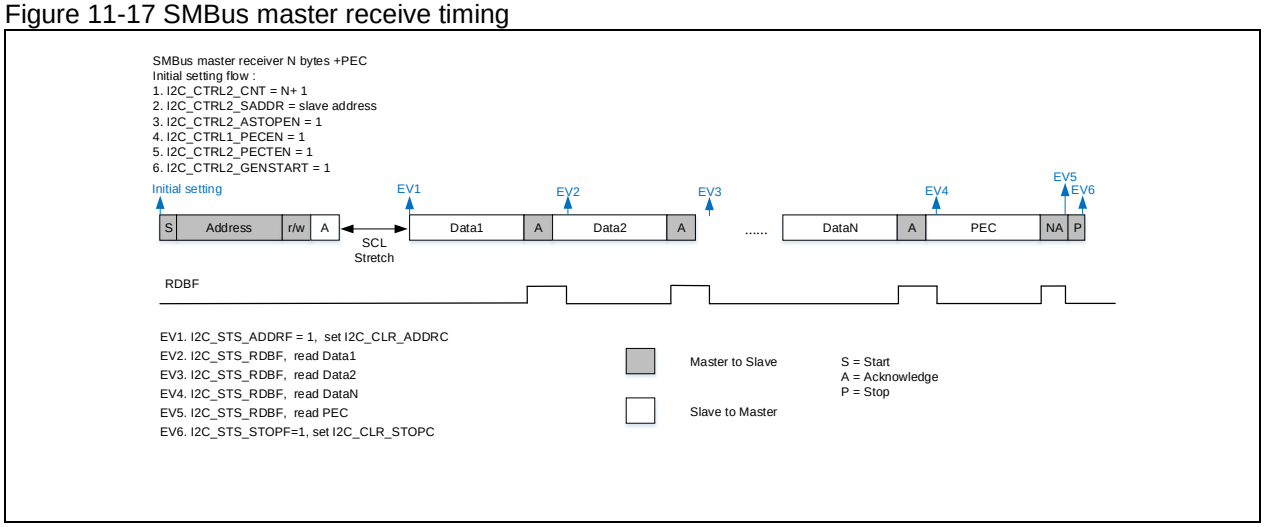


Figure 11-17 SMBus master receive timing



11.4.7 SMBus slave communication flow

The SMBus is similar to the I²C in terms of slave communication flow.

1. I²C clock initialization (by setting the I2C_CLKCTRL register)

- I²C clock divider: DIV[7:0]
- Data hold time ($t_{HD;DAT}$): SDAD[3:0]
- Data setup time ($t_{SU;DAT}$): SCLD[3:0]

The register can be configured by means of Artery_I2C_Timing_Configuration tool.

2. Set local address

- Set 7-bit address mode: by setting ADDR1MODE = 0 in the I2C_OADDR1 register
- Set address 1: by setting the ADDR1 bit in the I2C_OADDR1 register
- Enable address 1: by setting ADDR1EN=1 in the I2C_OADDR1 register

3. SMBus-relate initialization

- Select SMBus device: device default address acknowledged (0b1100001x) by setting DEVADDREN=1
- Enable PEC calculation: by setting PECEN=1 in the I2C_CTRL1 register
- Set slave byte control mode:

  Slave transmit: disable byte control mode by setting SCTRL=0 in the I2C_CTRL1 register

  Slave receive: enable byte control mode by setting SCTRL=1 in the I2C_CTRL1 register

4. Wait for address matching

When the local address is received, the ADDRF bit is set in the I2C_STS register. The data transfer direction can be obtained by read access to the SDIR bit in the I2C_STS register. When SDIR=0, it indicates that the slave is receiving data, whereas SDIR=1 indicates that the slave is sending data. The ADDR[6:0] bit in the I2C_STS register indicates what kind of address has been received.

Enable PEC transfer: by setting PECTEN=1 in the I2C_CTRL2 register

Set the number of data to be transferred:

- Slave transmit: by setting CNT=N in the I2C_CTRL2 register
- Slave receive: by setting CNT=1 in the I2C_CTRL2 register

Set reload mode:

- Slave transmit: by setting RLDEN=0 in the I2C_CTRL2 register
- Slave receive: by setting RLDEN=1 in the I2C_CTRL2 register

The ADDRF flag can be cleared by setting ADDRC=1 in the I2C_CLR register, and then data transfer starts.

5. Data transfer (slave transmission, clock stretching enabled, STRETCH=0)

After address matching:

1. I2C_TXDT register becomes empty, the shift register becomes empty, and TDIS=1 in the I2C_STS register;
2. Write 1 to the TXDT register, and data is immediately moved the shift register;
3. TXDT register is empty, and TDIS=1 again;
4. Write 2 to the TXDT register, and TDIS is cleared;
5. Repeat steps 3 and 4 until the data (N-1) is sent;
6. The slave will automatically transmit the Nth data, that is, PEC;
7. Wait for the generation of a NACK signal. Once received, the ACKFAILF is set in the I2C_STS register. The ACKFAILF flag is cleared by writing 1 to the ACKFAILC bit;
8. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C_STS register. The STOPF is cleared by writing 1 to the STOPC bit in the I2C_CLR register, and transmission ends.

6. Data transfer (slave receive, clock stretching enabled, STRETCH=0)

After address matching:

1. I2C_RXDT register becomes empty, the shift register becomes empty, and RDBF=0 in the I2C_STS register;
2. Upon the receipt of one-byte data, RDBF=1 and TCRLD=1, then the SCL is pulled by the slave;
3. Read the RXDT register, and the RDBF is cleared automatically;
4. The NACKEN bit in the I2C_CTRL2 register can be configured to generate an ACK or NACK, if needed:

   If a NACK is detected, it indicates the completion of communication;

   If an ACK is detected, communication continues. Writing CNT=1 will automatically clear the TCRLD flag by hardware, and the SCL is released by the slave for the reception of the next data.
5. Repeat steps 2, 3 and 4 until the data (N-1) is received;
6. Set RLDEN=0 in the I2C_CTRL2 register to disable reload mode. Set CNT=1 and repeat steps 2 and 3 to receive a PEC. The PECERR bit will be set if a PEC error occurs.
7. Wait for the generation of a STOP condition. Once received, the STOPF is set in the I2C_STS register. The STOPF can be cleared by writing 1 to the STOPC bit in the I2C_CLR register, and transfer ends.

SMBus slave transmitter

Figure 11-18 SMBus slave transmission flow

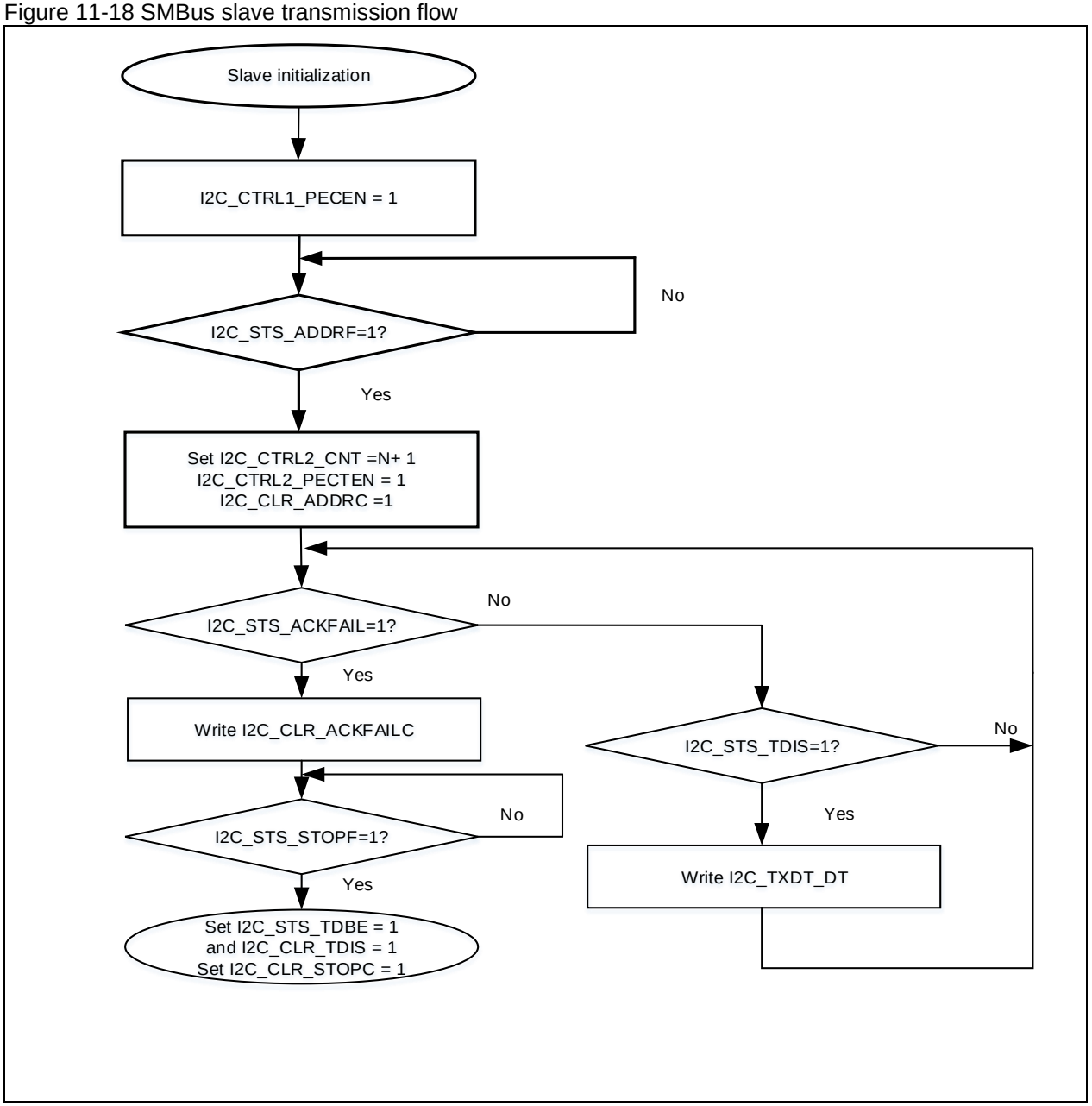
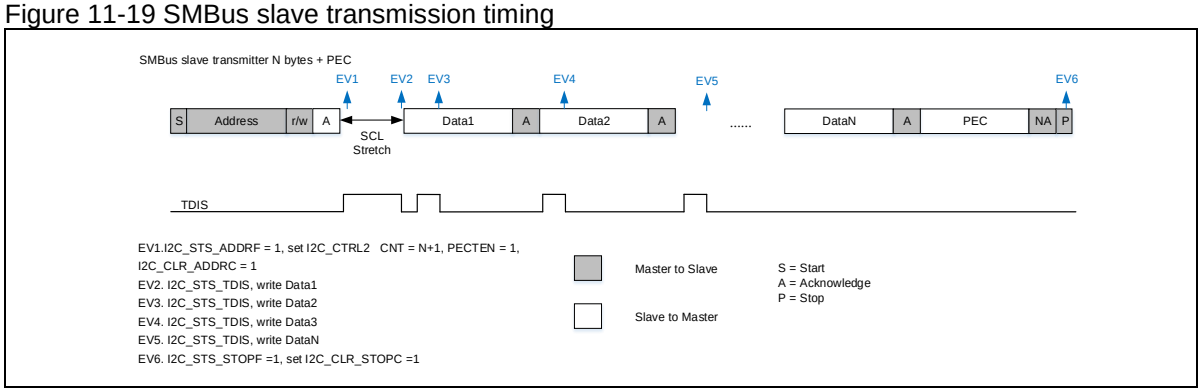


Figure 11-19 SMBus slave transmission timing



SMBus slave receiver

Figure 11-20 SMBus slave receive flow

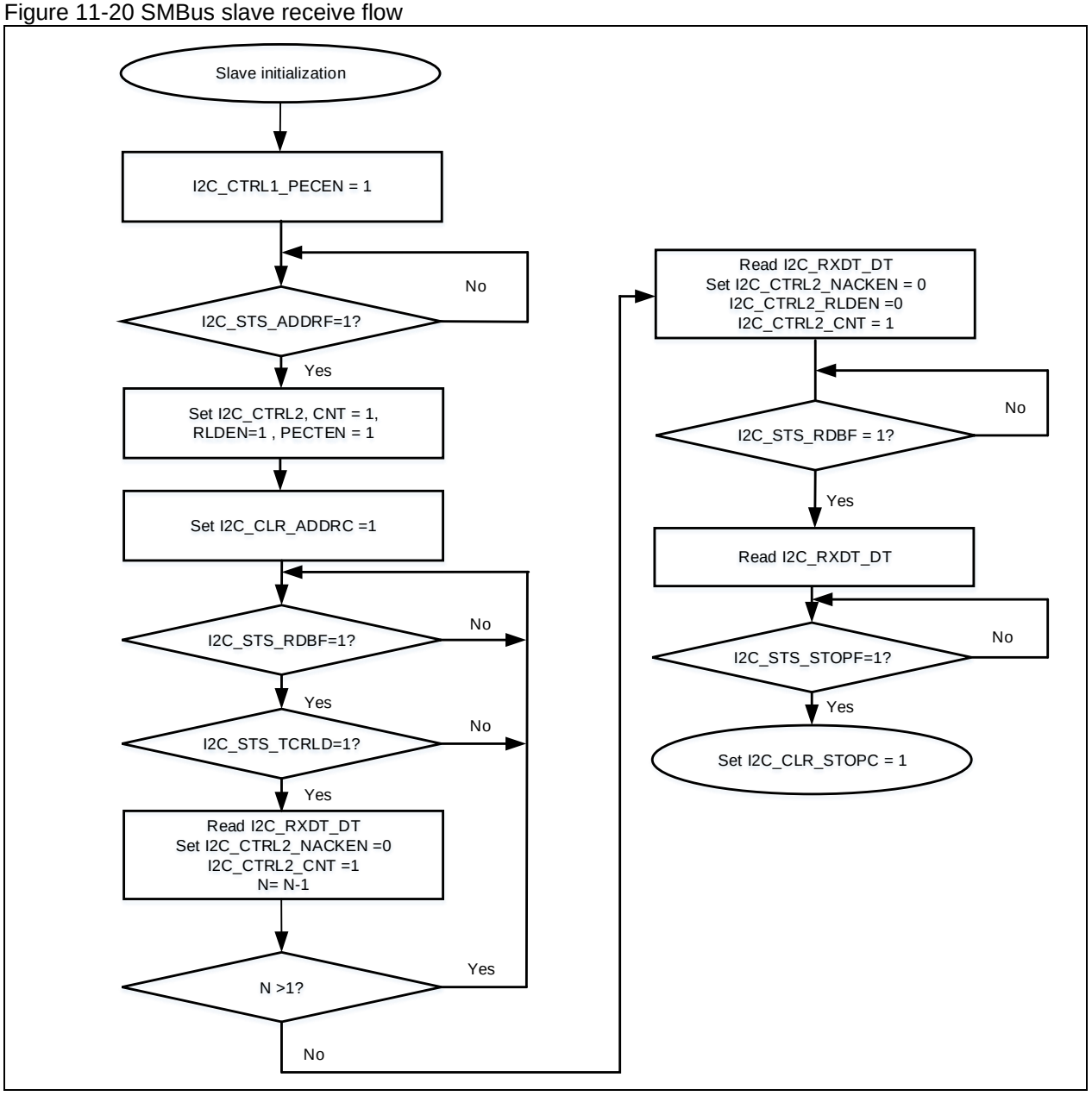
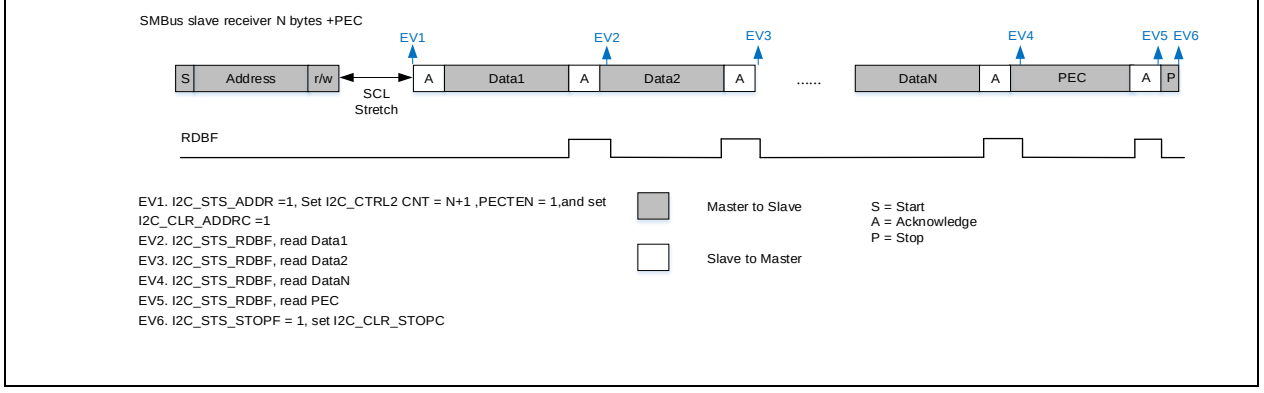


Figure 11-21 SMBus slave receive timing



11.4.8 Data transfer using DMA

I²C data transfer can be done using DMA controller so as to reduce the burden on the CPU. The TDIEN and RDIEN must be set to 0 when using DMA for data transfer.

Transmission using DMA (DMATEN=1)

1. Set the peripheral address (DMA_CxPADDR= I2C_TXDT address)
2. Set the memory address (DMA_CxMADDR=data memory address)
3. The transmission direction is set from memory to peripheral (DTD=1 in the DMA_CHCTRL register);
4. Configure the total number of bytes to be transferred in the DMA_CxDTCNT register;
5. Configure other parameters (such as priority, memory data width, peripheral data width, interrupts, etc.) in the DMA_CHCTRL register;
6. Enable the DMA channel by setting CHEN=1 in the DMA_CxCTRL register;
7. Enable I²C DMA request by setting DMATEN=1 in the I2C_CTRL1 register. Once the TDIS bit in the I2C_STS1 register is set, the data is loaded from the programmed memory to the I2C_TXDT register through DMA;
8. When the number of data transfers, programmed in the DMA controller, is reached (DMA_CxDTCNT), the data transfer is complete (an interrupt is generated if enabled);
9. Master transmitter: refer to the I²C master communication flow section for STOP condition.

   Slave transmitter: refer to the I²C slave communication flow section for STOP condition.

Reception using DMA (DMAREN=1)

1. Set the peripheral address (DMA_CxPADDR= I2C_RXDT address);
2. Set the memory address (DMA_CxMADDR=data memory address);
3. The transmission direction is set from peripheral to memory (DTD=0 in the DMA_CHCTRL);
4. Configure the total number of bytes to be transferred in the DMA_CxDTCNT register;
5. Configure other parameters (such as priority, memory data width, peripheral data width, interrupts, etc.) in the DMA_CHCTRL register;
6. Enable the DMA channel by setting CHEN=1 in the DMA_CxCTRL register;
7. Enable I²C DMA request by setting DMAREN=1 in the I2C_CTRL1 register. Once the RDBF bit is set in the I2C_STS1 register, the data is loaded from the I2C_DT register to the programmed memory through DMA;
8. When the number of data transfers, programmed in the DMA controller, is reached (DMA_TCNTx=0), the data transfer is complete (an interrupt is generated if enabled).
9. Master receiver: refer to the I²C master communication flow section for STOP condition.

   Slave receiver: refer to the I²C slave communication flow section for STOP condition.

11.4.9 Error management

The error management feature included in the I²C provides a guarantee for the reliability of communication. The manageable error events are listed below:

Table 11-6 I²C error events

Error event	Event flag	Enable control bit	Clear bit
SMBus Alert	ALERTF	ERRIEN	ALERTC
Timeout error	TMOUT	ERRIEN	TMOUTC
PEC error	PECERR	ERRIEN	PECERRC
Overrun/underrun	OUF	ERRIEN	OUFC
Arbitration lost	ARLOST	ERRIEN	ARLOSTC
Bus error	BUSERR	ERRIEN	BUSERRC

Overrun/Underrun (OUF)

In slave mode, an underrun/overrun may appear if the clock stretching feature is disabled (STRETCH=1 in the I2C_CTRL1 register).

In slave transmit mode: if data has not yet been written to the TXDT register before the transmission of the first bit of the to-be-transferred data (that is, before the generation of SDA edge), an underrun error may occur, and the OUF bit is set in the I2C_STS register, sending 0xFF to the bus.

In slave receive mode: The slave must read the received data as soon as it receives the data. If one-byte data has been received and data is not read yet before the end of the next data reception, an overrun error occurs, setting OUF=1 in the I2C_STS register, and sending NCAK.

Arbitration lost (ARLOST)

An arbitration lost may occur when the device controls the SDA line to output high level but the actual bus output is low.

- Master transmit: An arbitration may occur during an address transfer and a data transfer
- Master receive: An arbitration may occur during an address transfer and an ACK response
- Slave transmit: An arbitration may occur during a data transfer
- Slave receive: An arbitration may occur during an ACK response

Once an arbitration lost is detected, the ARLOST is set by hardware in the I2C_STS register. The SCL and SDA buses will be released and go automatically back to slave mode.

Bus error (BUSERR)

The SDA line, during a data transfer, must be kept in a stable state when the SCL is in high level. The SDA can be changed only when the SCL signal becomes low; otherwise, a bus error may appear.

- SDA changes from 1 to 0: a misplaced START condition
- SDA changes from 0 to 1: a misplaced STOP condition

Both of these conditions above may trigger a bus error. Once it occurs, the BUSERR is set by hardware in the I2C_STS register.

PEC error (PECERR)

The PEC is available only in SMBus mode. In master receive and slave receive modes, a PEC error may appear if the received PEC is not equal to the internally calculated PEC. In this case, the PECERR bit is set by hardware in the I2C_STS register.

In slave receive mode, an NACK is sent when a PEC error is detected.

In master receive mode, an NACK is always sent, whatever the PEC check result.

SMBus alert (ALERTF)

The SMBus alert feature is present when HADDREN=1 (SMBus master mode) and SMBALERT=1 (SMBus alert mode). Once an alert event is detected on the ALERT pin (ALERT pin changes from high to low), the ALERTF bit is set by hardware in the I2C_STS register.

Timeout error (TMOUT)

SMBus defines a timeout mechanism for the improvement of the system stability, preventing the bus from being pulled down in the case of a master or slave failure. Once a timeout event (defined in SMBus chapter) is detected, the TMOUT is set by hardware in the I2C_STS register. If a timeout error occurs in slave mode, the SCL and SDA buses are immediately released; if a timeout error occurs in master mode, a STOP condition is automatically by host to abort the communication.

11.5 I²C interrupt requests

The following table lists all the I²C interrupt requests.

Table 11-7 I²C interrupt requests

Interrupt event	Event flag	Enable control bit
Address matched	ADDRF	ADDRIEN
Acknowledge failure	ACKFAIL	ACKFAILIEN
Stop condition received	STOPF	STOPIEN
Transmit interrupt state	TDIS	TDIEN
Receive data buffer full	RDBF	RDIEN
Transfer complete, wait for loading data	TCRLD	TDCIEN
Data transfer complete	TDC	
SMBus alert	ALERTF	ERRIEN
Timeout error	TMOUT	
PEC error	PECERR	
Overrun/underrun	OUF	
Arbitration lost	ARLOST	
Bus error	BUSERR	

11.6 I²C debug mode

When the microcontroller enters debug mode (Cortex®-M4F halted), the SMBUS timeout either continues to work or stops, depending on the I2Cx_SMBUS_TIMEOUT configuration bit in the DEBUG module.

11.7 I²C registers

These peripheral registers must be accessed by words (32 bits).

Table 11-8 I²C register map and reset value

Register	Offset	Reset value
I2C_CTRL1	0x00	0x00000000
I2C_CTRL2	0x04	0x00000000
I2C_OADDR1	0x08	0x00000000
I2C_OADDR2	0x0C	0x00000000
I2C_CLKCTRL	0x10	0x00000000
I2C_TIMEOUT	0x14	0x00000000
I2C_STS	0x18	0x00000000
I2C_CLR	0x1C	0x00000000
I2C_PEC	0x20	0x00000000
I2C_RXDT	0x24	0x00000000
I2C_TXDT	0x28	0x00000000

11.7.1 Control register 1 (I2C_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	res	Kept at its default value.
Bit 23	PECEN	0x0	rw	PEC calculation enable 0: Disabled 1: Enabled
Bit 22	SMBALERT	0x0	rw	SMBus alert enable / pin set To enable SMBus master alert feature: 0: Disabled 1: Enabled To enable SMBus slave alert address: 0: Pin high 1: Pin low, response address 0001100x
Bit 21	DEVADDREN	0x0	rw	SMBus device default address enable 0: Disabled 1: Enabled, response device default address 1100001x
Bit 20	HADDREN	0x0	rw	SMBus host default address enable 0: Disabled 1: Enabled, response host address 0001000x
Bit 19	GCAEN	0x0	rw	General call address enable 0: Disabled 1: Enabled, response address 0000000x
Bit 18	Reserved	0x0	res	Kept at its default value.
Bit 17	STRETCH	0x0	rw	Clock stretching mode 0: Enabled 1: Disabled Note: It is valid in slave mode. Note: This bit can be set only when I^2C is disabled (I2CEN=0).
Bit16	SCTRL	0x0	rw	Slave receiving data control 0: Disabled 1: Enabled
Bit 15	DMAREN	0x0	rw	DMA receive data request enable 0: Disabled 1: Enabled
Bit 14	DMATEN	0x0	rw	DMA transmit data request enable 0: Disabled 1: Enabled
Bit 13:12	Reserved	0x0	res	Kept at its default value.
Bit 11:8	DFLT	0x0	rw	Digital filter value The glitches less than the filter time on the SCL bus will filtered; filter time = DFLT x T_{I2C_CLK}. Note: These bits can be set only when I^2C is disabled (I2CEN=0).

Bit 7	ERRIEN	0x0	rw	Error interrupt enable 0: Disabled 1: Enabled
Bit 6	TDCIEN	0x0	rw	Transfer data complete interrupt enable 0: Disabled 1: Enabled
Bit 5	STOPIEN	0x0	rw	Stop generation complete interrupt enable 0: Disabled 1: Enabled
Bit 4	ACKFAILIEN	0x0	rw	Acknowledge fail interrupt enable 0: Disabled 1: Enabled
Bit 3	ADDRIEN	0x0	rw	Address match interrupt enable 0: Disabled 1: Enabled
Bit 2	RDIEN	0x0	rw	Receive data interrupt enable 0: Disabled 1: Enabled
Bit 1	TDIEN	0x0	rw	Transmit data interrupt enable 0: Disabled 1: Enabled
Bit 0	I2CEN	0x0	rw	I^2C peripheral enable 0: Disabled 1: Enabled

11.7.2 Control register 2 (I2C_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31:27	Reserved	0x00	res	Kept at its default value.
Bit 26	PECTEN	0x0	rw	Request PEC transmission enable 0: Transmission disabled 1: Transmission enabled Note: This bit can be set only when I^2C is enabled (I2CEN=1).
Bit 25	ASTOPEN	0x0	rw	Automatically send stop condition enable 0: Disabled (Software sends STOP condition) 1: Enabled (Automatically send STOP condition)
Bit 24	RLDEN	0x0	rw	Send data reload mode enable 0: Disabled 1: Enabled
Bit 23:16	CNT[7:0]	0x00	rw	Transmit data counter Note: These bits are invalid when SCTRL=0 in slave mode. Note: These bits can be set only when I^2C is disabled (I2CEN=0).
Bit 15	NACKEN	0x0	rw	Not acknowledge enable

				0: Acknowledge enabled 1: Acknowledge disabled Note: This bit can be set only when I²C is enabled (I2CEN=1). Note: This bit is reset when I²C is disabled (I2CEN=0).
Bit 14	GENSTOP	0x0	rw	Generate stop condition 0: No stop generation 1: Stop generation Note: This bit can be set only when I²C is enabled (I2CEN=1). Note: This bit is reset by hardware automatically when a STOP condition is detected.
Bit 13	GENSTART	0x0	rw	Generate start condition 0: No start generation 1: Start generation Note: This bit can be set only when I²C is enabled (I2CEN=1). Note: This bit is reset by hardware automatically when a START condition is detected.
Bit 12	READH10	0x0	rw	10-bit address header read enable 0: 10-bit address header read disabled 1: 10-bit address header read enabled
Bit 11	ADDR10	0x0	rw	Host sends 10-bit address mode enable 0: 7-bit address mode 1: 10-bit address mode
Bit 10	DIR	0x0	rw	Master data transmission direction 0: Transmit 1: Receive
Bit 9:0	SADDR[9:0]	0x000	rw	Slave address sent by the master In 7-bit address mode, BIT0 and BIT[9:8] don't care.

11.7.3 Address register 1 (I2C_OADDR1)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	res	Kept at its default value.
Bit 15	ADDR1EN	0x0	rw	Own address 1 enable 0: Own address 1 disabled 1: Own address 1 enabled
Bit 14:11	Reserved	0x0	res	Kept at its default value.
Bit 10	ADDR1MODE	0x0	rw	Own address 1 mode 0: 7-bit address 1: 10-bit address Note:This bit can be set only when ADDR1EN=1.
Bit 9:0	ADDR1[9:0]	0x000	rw	Own address 1 In 7-bit address mode, BIT0 and BIT[9:8] don't care. Note: These bits can be set only when ADDR1EN=1.

11.7.4 Address register 2 (I2C_OADDR2)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	res	Kept at its default value.
Bit 15	ADDR2EN	0x0	rw	Own address 2 enable 0: Own address 2 disabled 1: Own address 2 enabled
Bit 14:11	Reserved	0x0	res	Kept at its default value.
Bit 10:8	ADDR2MASK[2:0]	0x0	rw	Own address 2-bit mask 000: Match address bit [7:1] 001: Match address bit [7:2] 010: Match address bit [7:3] 011: Match address bit [7:4] 100: Match address bit [7:5] 101: Match address bit [7:6] 110: Match address bit [7] 111: Response all addresses other than those reserved for I^2C Note: These bits can be set only when ADDR2EN=1.
Bit 7:1	ADDR2[7:1]	0x00	rw	Own address 2 7-bit address Note: These bits can be set only when ADDR2EN=1.
Bit 0	Reserved	0x0	res	Kept at its default value.

11.7.5 Timing register (I2C_CLKCTRL)

Bit	Name	Reset value	Type	Description
Bit 31:28	DIVL[3:0]	0x0	rw	Low 4 bits of clock divider value
Bit 27:24	DIVH[7:4]	0x0	rw	High 4 bits of clock divider value DIV = (DIVH << 4) + DIVL
Bit 23:20	SCLD[3:0]	0x0	rw	SCL output delay T_{SCLD} = (SCLD + 1) x (DIV + 1) x T_{I2C_CLK}
Bit 19:16	SDAD[3:0]	0x0	rw	SDA output delay) T_{SDAD} = (SDAD + 1) x (DIV + 1) x T_{I2C_CLK}
Bit 15:8	SCLH[7:0]	0x00	rw	SCL high level) T_{SCLH} = (SCLH + 1) x (DIV + 1) x T_{I2C_CLK}
Bit 7:0	SCLL[7:0]	0x00	rw	SCL low level T_{SCLL} = (SCLL + 1) x (DIV + 1) x T_{I2C_CLK}

Note: The I2C_CLKCTRL register can be configured only when I^2C is disabled (I2CEN=0).

11.7.6 Timeout register (I2C_TIMEOUT)

Bit	Name	Reset value	Type	Description
Bit 31	EXTEN	0x0	rw	Cumulative clock low extend timeout enable 0: Disabled 1: Enabled Corresponds to $T_{LOW:SEXT}$ / $T_{LOW:MEXT}$ in SMBus
Bit 30:28	Reserved	0x0	res	Kept at its default value.
Bit 27:16	EXTTIME[11:0]	0x000	rw	Cumulative clock low extend timeout value Timeout = (EXTTIME + 1) x 2048 x T_{I2C_CLK} Note: These bits can be set only when EXTEN=1.
Bit 15	TOEN	0x0	rw	Detect clock low/high timeout enable 0: Disabled 1: Enabled Corresponds to $T_{TIMEOUT}$ in SMBus
Bit 14:13	Reserved	0x0	res	Kept at its default value.
Bit 12	TOMODE	0x0	rw	Clock timeout detection mode 0: Clock low level detetion 1: Clock high level detection Note: This bit can be set only when TOEN =1.
Bit 11:0	TOTIME[11:0]	0x000	rw	Clock timeout detection time For clock low level detection (TOMODE = 0): Timeout duration = (TOTIME + 1) x 2048 x T_{I2C_CLK} For clock high level detection (TOMODE = 1): Timeout duration = (TOTIME + 1) x 4 x T_{I2C_CLK} Note: These bits can be set only when TOEN =1.

11.7.7 Status register (I2C_STS)

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	res	Kept at its default value.
Bit 23:17	ADDR[6:0]	0x00	r	Slave address matching value In 7-bit address mode: Slave address received In 10-bit address: 10-bit slave address header received
Bit 16	SDIR	0x0	r	Slave data transmit direction 0: Receive data 1: Transmit data
Bit 15	BUSYF	0x0	r	Bus busy flag transmission mode 0: Bus idle 1: Bus busy Once a START condition is detected, this bit is set. Once a STOP condition is detected, this bit is automatically cleared.
Bit 14	Reserved	0x00	res	Kept at its default value.
Bit 13	ALERTF	0x0	r	SMBus alert flag SMBus host: This bit indicates the reception of an alert signal (ALERT pin changes from high to low) 0: No alert signal received 1: Alert signal received
Bit 12	TMOUT	0x0	r	SMBus timeout flag 0: No timeout 1: Timeout
Bit 11	PECERR	0x0	r	PEC receive error flag 0: No PEC error 1: PEC error
Bit 10	OUF	0x0	r	Overflow or underflow flag In transmission mode: 0: No overflow or underflow 1: Underflow In reception mode: 0: No overflow or underflow 1: Overflow
Bit 9	ARLOST	0x0	r	Arbitration lost flag 0: No arbitration lost detected 1: Arbitration lost detected
Bit 8	BUSERR	0x0	r	Bus error flag 0: No bus error occurs 1: Bus error occurs
Bit 7	TCRLD	0x0	r	Transmission is complete, waiting to load data 0: Data transfer is not complete yet 1: Data transfer is complete This bit is set when data transfer is complete

				(CNT = 1) and reload mode is enabled (RLDEN=1). It is automatically cleared when writing a CNT value. This bit is applicable in master mode or when SCTRL=1 in slave mode.
Bit 6	TDC	0x0	r	Data transfer complete flag 0: Data transfer is not completed yet (the shift register still holds data) 1: Data transfer is completed (shift register becomes empty) This bit is set when ASTOPEN = 0, RLDEN = 0 and CNT = 0. It is automatically cleared after a START or a STOP condition is received.
Bit 5	STOPF	0x0	r	Stop condition generation complete flag 0: No Stop condition is generated 1: Stop condition is generated
Bit 4	ACKFAILF	0x0	r	Acknowledge failure flag 0: No acknowledge failure 1: Acknowledge failure
Bit 3	ADDRF	0x0	r	0~7 bit address match flag 0: 0~7 bit address mismatch 1: 0~7 bit address match
Bit 2	RDBF	0x0	r	Receive data buffer full flag 0: Data register has not received data yet 1: Data register has received data
Bit 1	TDIS	0x0	rw1s	Transmit data interrupt status 0: Data has been written to theI2C_TXDT； 1: Data has been sent from the I2C_TXDT to the shift register. I2C_TXDT becomes empty, and thus the to-be transferred data must be written to the I2C_TXDT. When the clock stretching mode is disabled, a TDIS event is generated by writing 1 so that data is written to the I2C_TXDT register in advance.
Bit 0	TDBE	0x1	rw1s	Transmit data buffer empty flag 0: I2C_TXDT holds data 1: I2C_TXDT is empty This bit is only used to indicate the current status of the I2C_TXDT register. The I2C_TXDT register can be refreshed by writing 1 through software.

11.7.8 Status clear register (I2C_CLR)

Bit	Name	Reset value	Type	Description
Bit 31:14	Reserved	0x00000	res	Kept at its default value.
Bit 13	ALERTC	0x0	w	Clear SMBus alert flag SMBus alert flag is cleared by writing 1.
Bit 12	TMOUTC	0x0	w	Clear SMBus timeout flag SMBus timeout flag is cleared by writing 1.
Bit 11	PECERRC	0x0	w	Clear PEC receive error flag PEC receive error flag is cleared by writing 1.
Bit 10	OUFC	0x0	w	Clear overload / underload flag Overload / underload flag is cleared by writing 1.
Bit 9	ARLOSTC	0x0	w	Clear arbitration lost flag Arbitration lost flag is cleared by writing 1.
Bit 8	BUSERRC	0x0	w	Clear bus error flag Bus error flag is cleared by writing 1.
Bit 7:6	Reserved	0x0	res	Kept at its default value.
Bit 5	STOPC	0x0	w	Clear stop condition generation complete flag The stop condition generation complete flag is cleared by writing 1.
Bit 4	ACKFAILC	0x0	w	Clear acknowledge failure flag The acknowledge failure flag is cleared by writing 1.
Bit 3	ADDRC	0x0	w	Clear 0~7 bit address match flag The 0~7 bit address match flag is cleared by writing 1.
Bit 2:0	Reserved	0x0	res	Kept at its default value.

11.7.9 PEC register (I2C_PEC)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	res	Kept at its default value.
Bit 7:0	PECVAL[7:0]	0x00	r	PEC value

Note: The I2C_PEC register is reset when I^2C is disabled (I2CEN=0).

11.7.10 Receive data register (I2C_RXDT)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	res	Kept at its default value.
Bit 7:0	DT[7:0]	0x00	r	Receive data register

Note: The I2C_RXDT register is reset when I^2C is disabled (I2CEN=0).

11.7.11 Transmit data register (I2C_TXDT)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x000000	res	Kept at its default value.
Bit 7:0	DT[7:0]	0x00	rw	Transmit data register

12 Universal synchronous/asynchronous receiver/transmitter (USART)

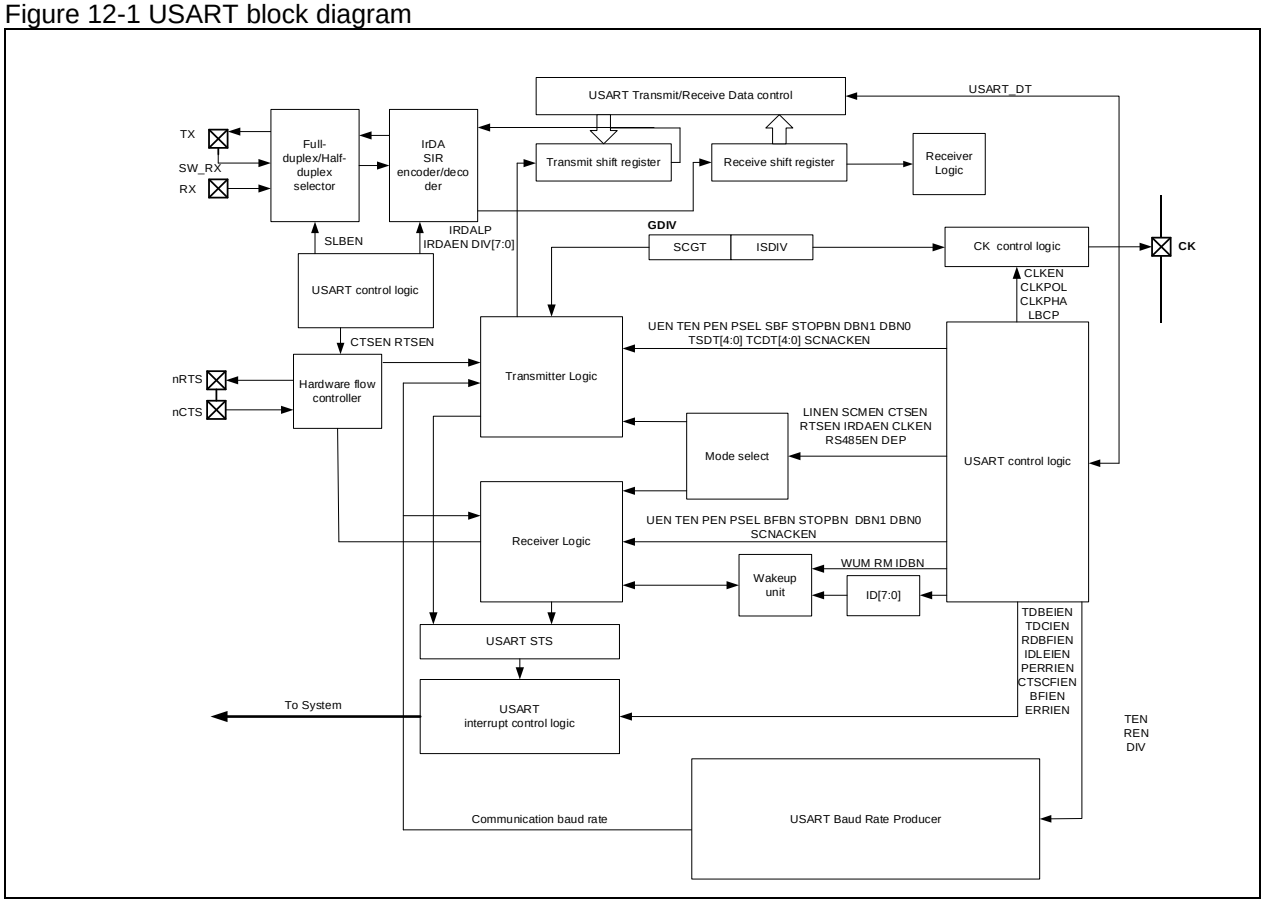
12.1 USART introduction

The universal synchronous/asynchronous receiver/transmitter (USART) serves an interface for communication by means of various configurations and peripherals with different data formats. It supports asynchronous full-duplex and half-duplex as well as synchronous transfer. USART offers a programmable baud rate generator,which enables users to configure the required communication frequency by setting the system frequency and frequency divider.

In addition to standard NRZ asynchronous and synchronous receiver/transmitter communication protocols, USART also supports widely-used serial communication protocols such as LIN (Local Interconnection Network), IrDA (Infrared Data Association) SIRENDEC specification, Asynchronous SmartCard protocol defined in ISO7816-3 standard, CTS/RTS (Clear To Send/Request To Send) hardware flow operation, RS485 and Modbus.

It also allows multi-processor communication, and supports silent mode waken up by idle frames or ID matching to build up a USART network. Meanwhile, high-speed communication is possible by using DMA.

Figure 12-1 USART block diagram



USART main features:

- Programmable full-duplex or half-duplex communication
  - Full-duplex, asynchronous communication
  - Half-duplex, single-wire communication
- Programmable communication modes
  - NRZ standard format (Mark/Space)
  - LIN (Local Interconnection Network)
  - IrDA SIR (SIR Serial Infrared)
  - Asynchronous SmartCard protocol defined in ISO7816-3 standard: support 0.5 or 1.5 stop bits in Smartcard mode
  - RS-232 CTS/RTS (Clear To Send/Request To Send) hardware flow operation
  - Multi-processor communication with silent mode (waken up by configuring ID match and bus idle frame)
  - Synchronous mode
- Programmable baud rate generator
  - Shared by transmission and reception
- Programmable frame format
  - Programmable data word length (7 bits, 8 bits or 9 bits)
  - Programmable stop bits: support 1 or 2 stop bits
  - Programmable parity control: transmitter with parity bit transmission capability, and receiver with received data parity check capability
  - Programmable data transmission order (MSB/LSB)
  - Programmable Tx/Rx pin polarity
  - Programmable DT polarity
- Programmable DMA multi-processor communication
- Programmable separate enable bits for transmitter and receiver
- Programmable output CLK phase, polarity and frequency
- Detection flags
  - Receive buffer full
  - transmit buffer empty
  - Transfer complete flag
- Four error detection flags
  - Overrun error
  - Noise error
  - Framing error
  - Parity error
- Programmable 12 interrupt sources with flags
  - CTS changes
  - LIN break detection
  - Transmit data register empty
  - Transmission complete
  - Receive data register full
  - Idle bus detected
  - Overrun error
  - Framing error

- Noise error
- Parity error
- Receiver timeout detection
- Byte match detection

12.2 Full-duplex/half-duplex selector

The full-duplex and half-duplex selector enables USART to perform data exchanges with peripherals in full-duplex or half-duplex mode, which is achieved by setting the corresponding registers.

In two-wire unidirectional full-duplex mode (by default), TX pin is used for data output, while the RX pin is used for data input. Since the transmitter and receiver are independent of each other, USART is allowed to send/receive data at the same time so as to achieve full-duplex communication.

When the SLBEN is set to 1, the single-wire bidirectional half-duplex mode is selected for communication. In this case, the LINEN, CLKEN, SCMEN and IRDAEN bits must be set to 0. RX pin is inactive, while TX and SW_RX are interconnected inside the USART. For the USART part, TX pin is used for data output and SW_RX for data input. For the peripheral part, bidirectional data transfer is executed through IO mapped by TX pin.

12.3 Mode selector

12.3.1 Introduction

USART mode selector allows USART to work in different operation modes through software configuration so as to enable data exchanges between USART and peripherals with different communication protocols.

USART supports NRZ standard format (Mark/Space), by default. It also supports LIN (Local Interconnection Network), IrDA SIR (Serial Infrared), Asynchronous Smartcard protocol in ISO7816-3 standard, RS-232 CTS/RTS (Clear To Send/Request To Send) hardware flow operation, silent mode and synchronous mode, depending on USART mode selection configuration.

12.3.2 Configuration procedure

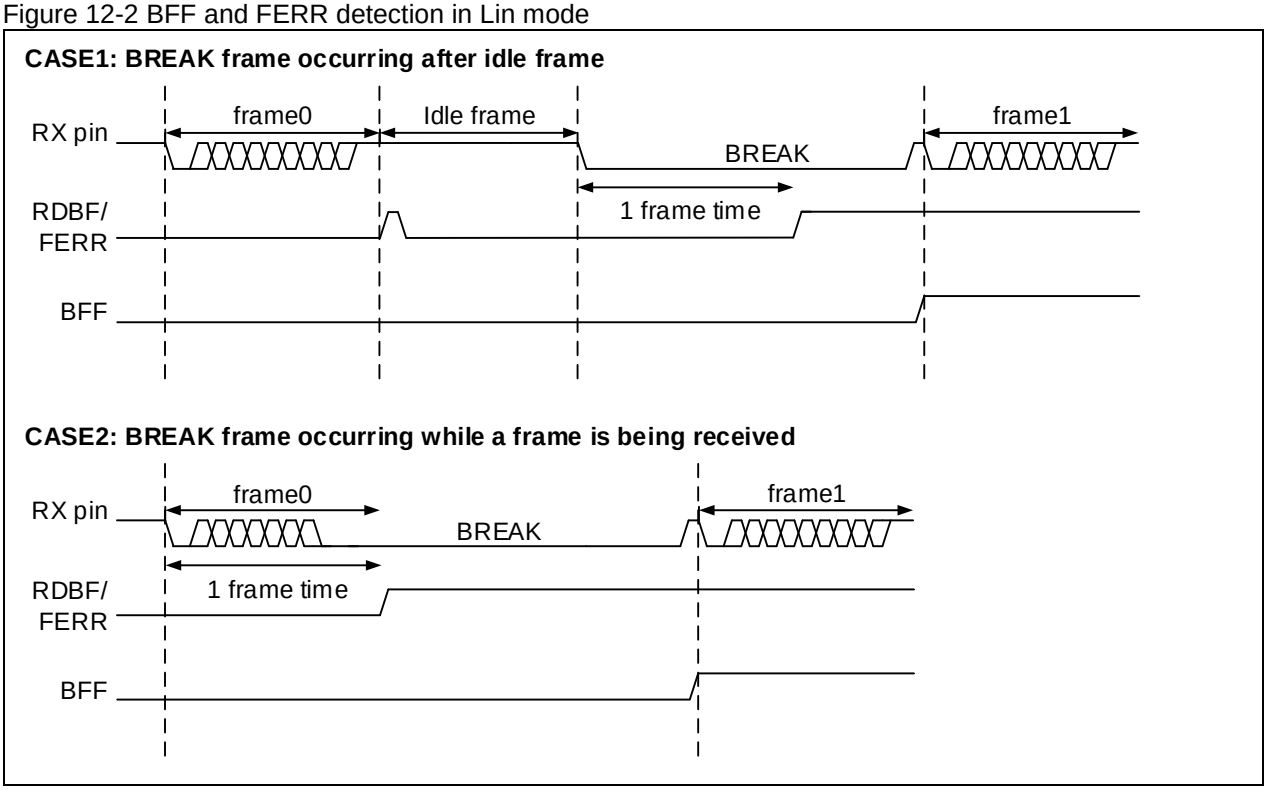
Selection of operation mode is done by following the configuration process listed below. In addition, such configuration method, along with those of receiver and transmitter described in the subsequent sections, are used to make USART initialization configuration.

1. **LIN mode**

   Set LINEN=1, CLKEN=0, STOPBN[1:0]=00, SCMEN=0, SLBEN=0, IRDAEN=0, DBN[1:0]=00.

   LIN master has break generation capability, and can transmit 13-bit low-level LIN synchronous break frame by setting SBF=1. The LIN slave has break detection capability, and can select 11-bit or 10-bit break detection by setting BFBN=1 or BFBN=0.

Figure 12-2 BFF and FERR detection in Lin mode



2. Smartcard mode

Set SCMEN=1, LINEN=0, SLBEN=0, IRDAEN=0, CLKEN=1, DBN[1:0]=01, PEN=1 and STOPBN[1:0]=11.

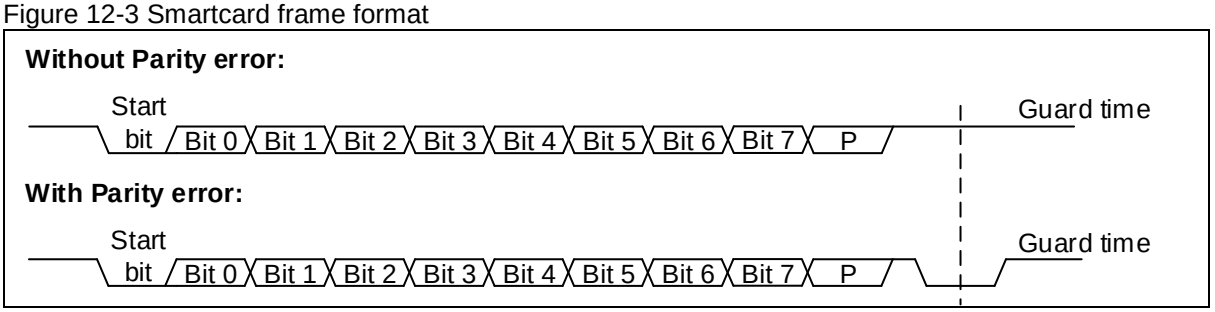
The polarity, phase and pulse number of the clock can be configured by setting the CLKPOL, CLKPHA and LBCP bits (Refer to Synchronous mode for details).

The assertion of the TDC flag can be delayed by setting the SCGT[7:0] bit (guard time bit). The TDF bit can be asserted high after the guard time counter reaches the value programmed in the SCGT[7:0] bit.

The Smartcard is a single-wire half-duplex communication protocol. The SCNACKEN bit is used to select whether to send NACK when a parity error occurs. This is to indicate to the Smartcard that the data has not been correctly received.

Note: It is recommended to use 1.5 stop bits for both transmitting and receiving. The guard time for smartcard must meet 1.5-bit time to receive back-to-back data.

Figure 12-3 Smartcard frame format

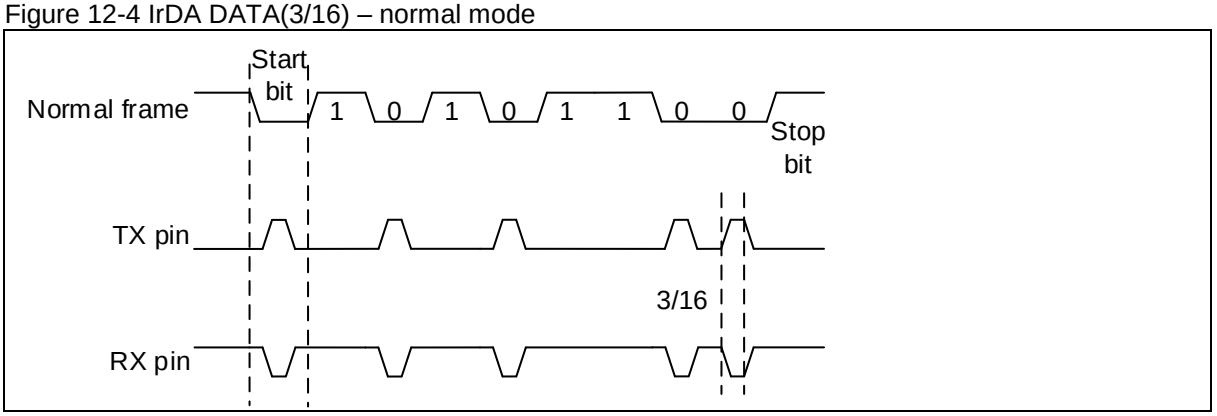


3. Infrared mode

Set IRDAEN=1, CLKEN=0, STOPBN[1:0]=00, SCMEN=0 and SLBEN=0.

The infrared low-power mode can be enabled by setting IRDALP=1. In normal mode, the transmitted pulse width is specified as 3/16 bits. In infrared low-power mode, the pulse width is configurable, and the ISDIV[7:0] bit can be used to achieve the desired low-power frequency.

Figure 12-4 IrDA DATA(3/16) – normal mode



4. Modbus

USART only supports basic hardware required by Modbus/RTU and Modbus/ASCII implementation, which means that the control must be done by software and USART provides EOB (end of block) detection only.

In Modbus/RTU, the EOB detection is implemented by the programmable timeout recognizing the receive line idle time being larger than 2 bytes. Users can configure the RTOV register to set the required timeout value (unit: 1-bit width), and enable timeout detection by setting RTODEN=1. When the receive line idle time detected by USART receiver is equal to the programmed timeout value, the USART will set RTODF. An interrupt is generated when RTODIE=1, and RTODF bit can be cleared by setting RTODCF=1.

In Modbus/ASCII, the EOB detection is implemented by the byte match feature recognizing the special byte sequence (CR/LF). Write LF ASCII code to the ID[7:0] and set CMDIE=1 to enable byte match feature. When the data received by USART matches ID[7:0], the USART will set CMDF. An interrupt is generated when CMDIE=1, and the CMDF bit can be cleared by setting CMDCF=1.

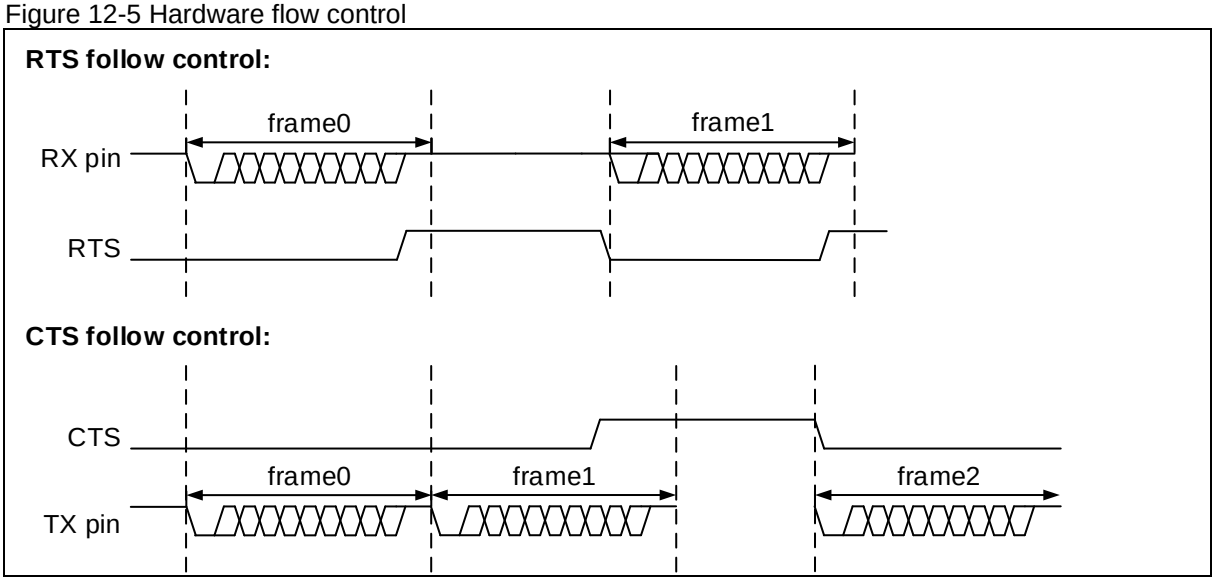
5. Hardware flow control mode

Setting RTSEN=1 and CTSEN=1 will enable RTS and CTS flow control, respectively.

RTS flow control: When the USART receiver is ready to receive new data, the RTS becomes effective (pull down low). When the data is received in the receiver (at the beginning of each stop bit), the RTS bit is set, indicating that the data transmission is to be stopped at the end of the current frame.

CTS flow control: USART transmitter checks CTS input before transmitting the next frame. If CTS is effective (that is, CTS is low), the next data is to be transmitted. If CTS becomes invalid (CTS is high) during transmission, the data transmission will stop after the completion of the current transmission.

Figure 12-5 Hardware flow control



6. RS485 mode

This mode is enabled by setting RS485EN=1. The enable signal is output on the RTS pin. The DEP bit is used to select polarity of the DE signal. The TSDT[4:0] bit is used to define the latency before the transmission of the start bit on the transmitter side, and the TCDT[4:0] bit is used to define the latency before the TC flag is set following the stop bit at the end of the last data.

7. Silent mode

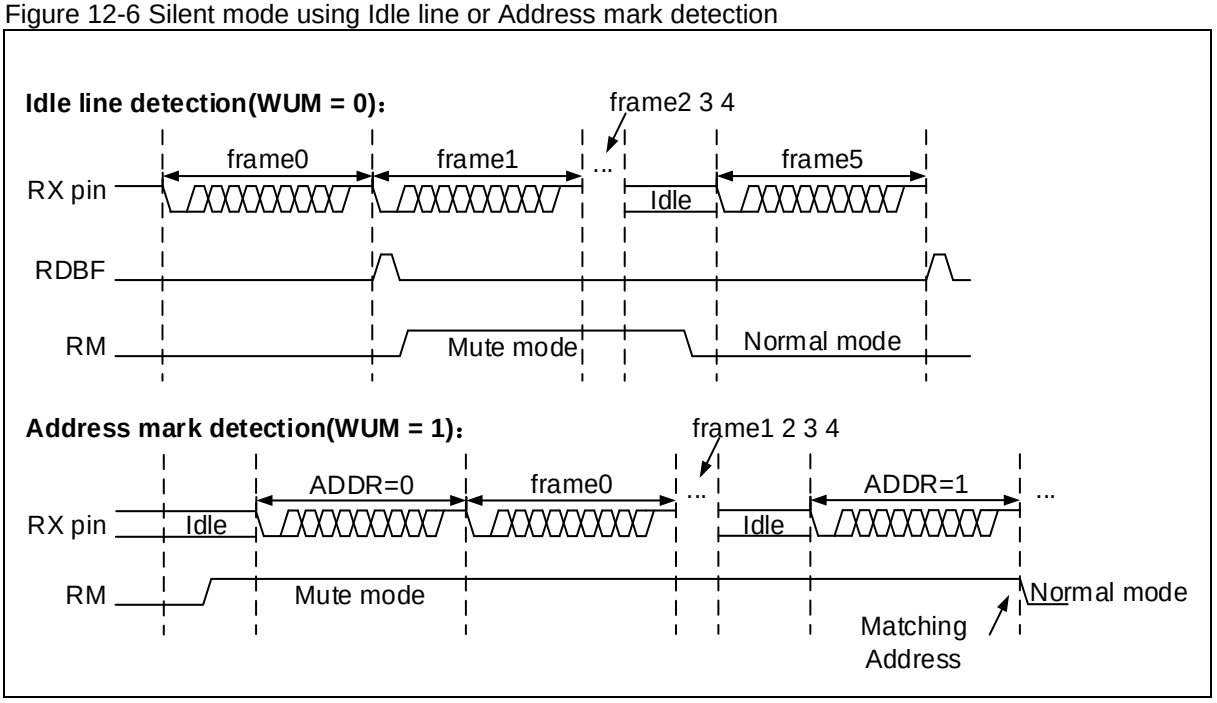
This mode is enabled by setting RM=1. When the WUM bit is set to 1 or 0, it wakes up from silent mode through ID match or idle bus, respectively. The ID[7:0] is configurable. The ID[7:0] or ID[3:0] can be selected by setting the IDBN bit. When ID match is selected, if the MSB of data bit is set, it indicates that the current data stands for ID.

When the parity check is disabled, if DBN[1:0]=10, MSB is USART_DT[6]; if DBN[1:0]=00, MSB is USART_DT[7]; if DBN[1:0]=01, MSB is USART_DT[8].

When the parity check is enabled, if DBN[1:0]=10, MSB is USART_DT[5]; if DBN[1:0]=00, MSB is USART_DT[6]; if DBN[1:0]=01, MSB is USART_DT[7].

When ID[3:0] is selected, the four LSB bits indicate the ID value. When ID[7:0] is selected, all LSB bits indicate the ID value except for the parity check bit and MSB bit.

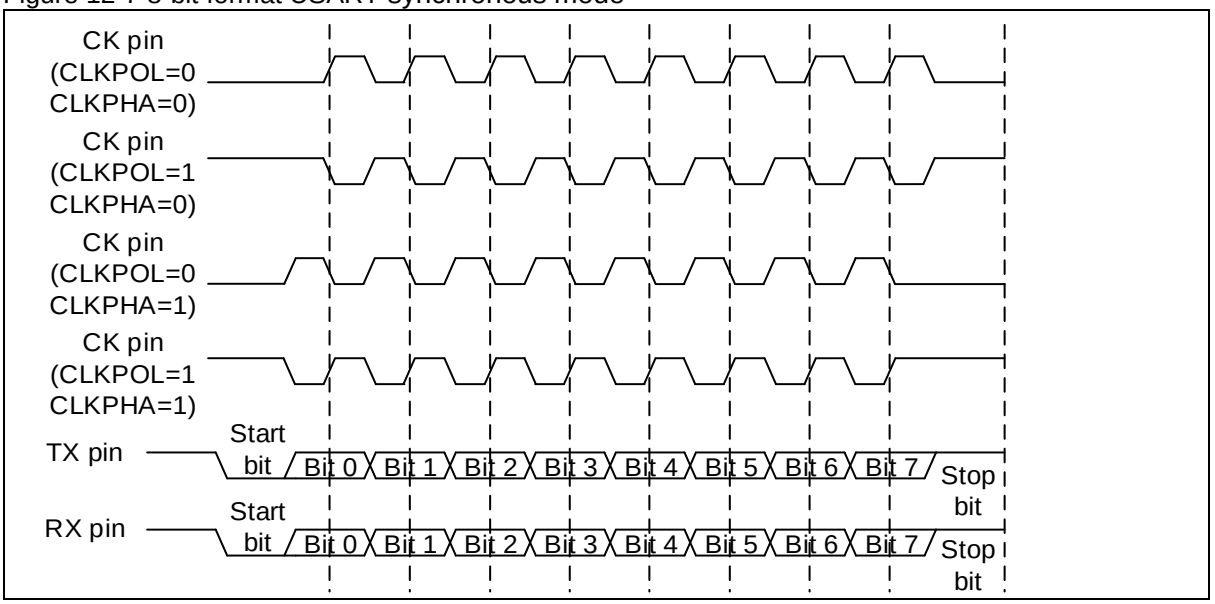
Figure 12-6 Silent mode using Idle line or Address mark detection



8. Synchronous mode

Setting CLKEN=1 enables the synchronous mode and clock pin output. Select CK pin high or low in idle mode by setting the CLKPOL bit (1 or 0), and select to sample data on the second or first edge of the clock by setting the CLKPHA bit (1 or 0). The LBCP bit (1 or 0) is used to select whether to output clock on the last data bit, and the ISDIV[4:0] bit is used to select the required clock output frequency.

Figure 12-7 8-bit format USART synchronous mode



12.4 USART frame format and configuration

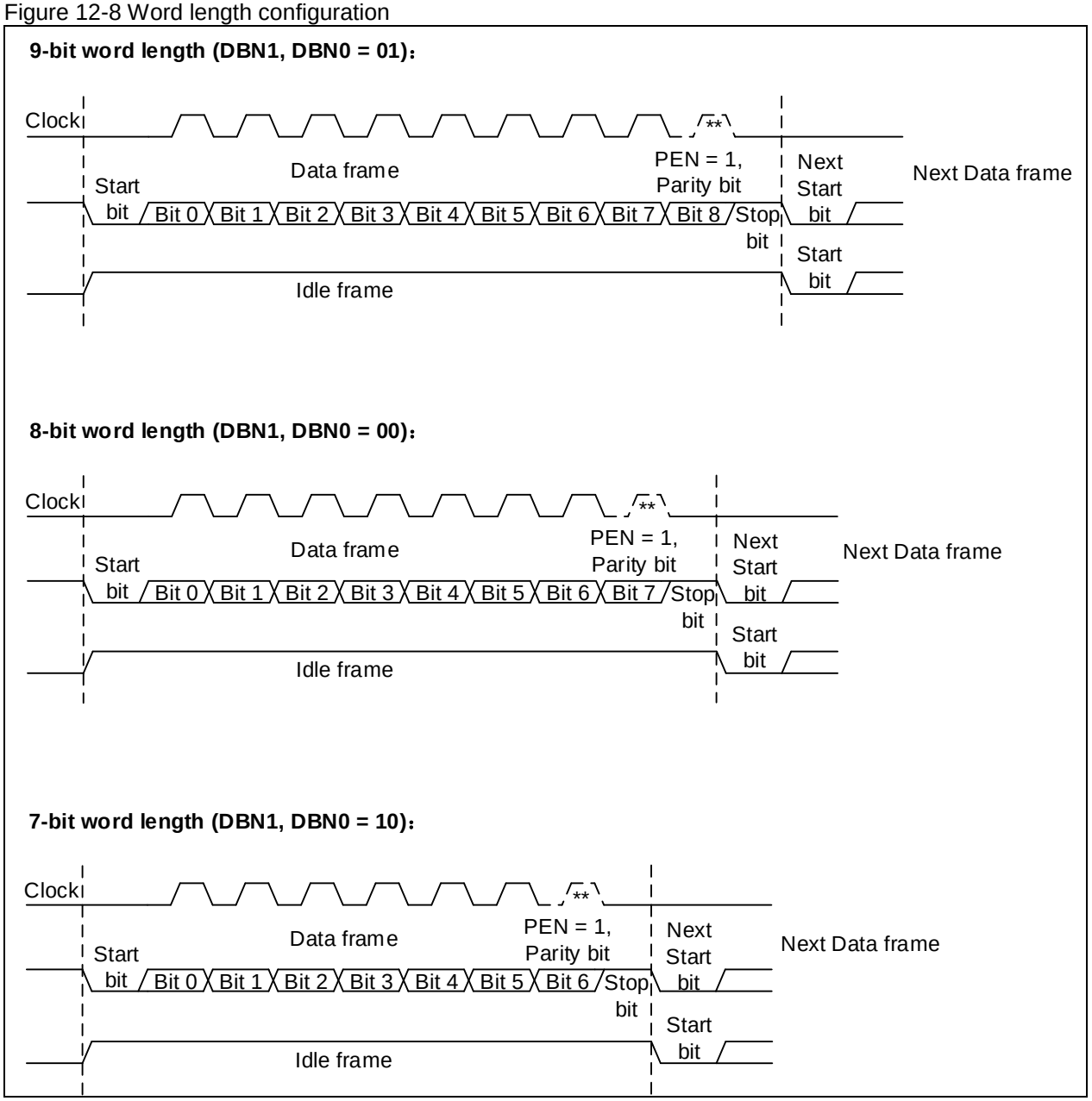
USART data frame consists of start bit, data bit and stop bit, with the last data bit being as a parity bit.

USART idle frame size is equal to that of the data frame under current configuration, but all bits are 1.

USART break frame size is the current data frame size plus its stop bit. All bits before the stop bit are 0. In non-LIN mode, a break frame transmission and detection must be in line with this rule. For instance, if DBN[1:0]=00, the break frame size for transmission and detection should be 10-bit low level plus its stop bit. In LIN mode, refer to Mode selector and configuration process for more details.

The DBN1 and DBN0 bits are used to program 7-bit (DBN[1:0]=10), 8-bit (DBN[1:0]=00) or 9-bit (DBN[1:0]=01) data bits.

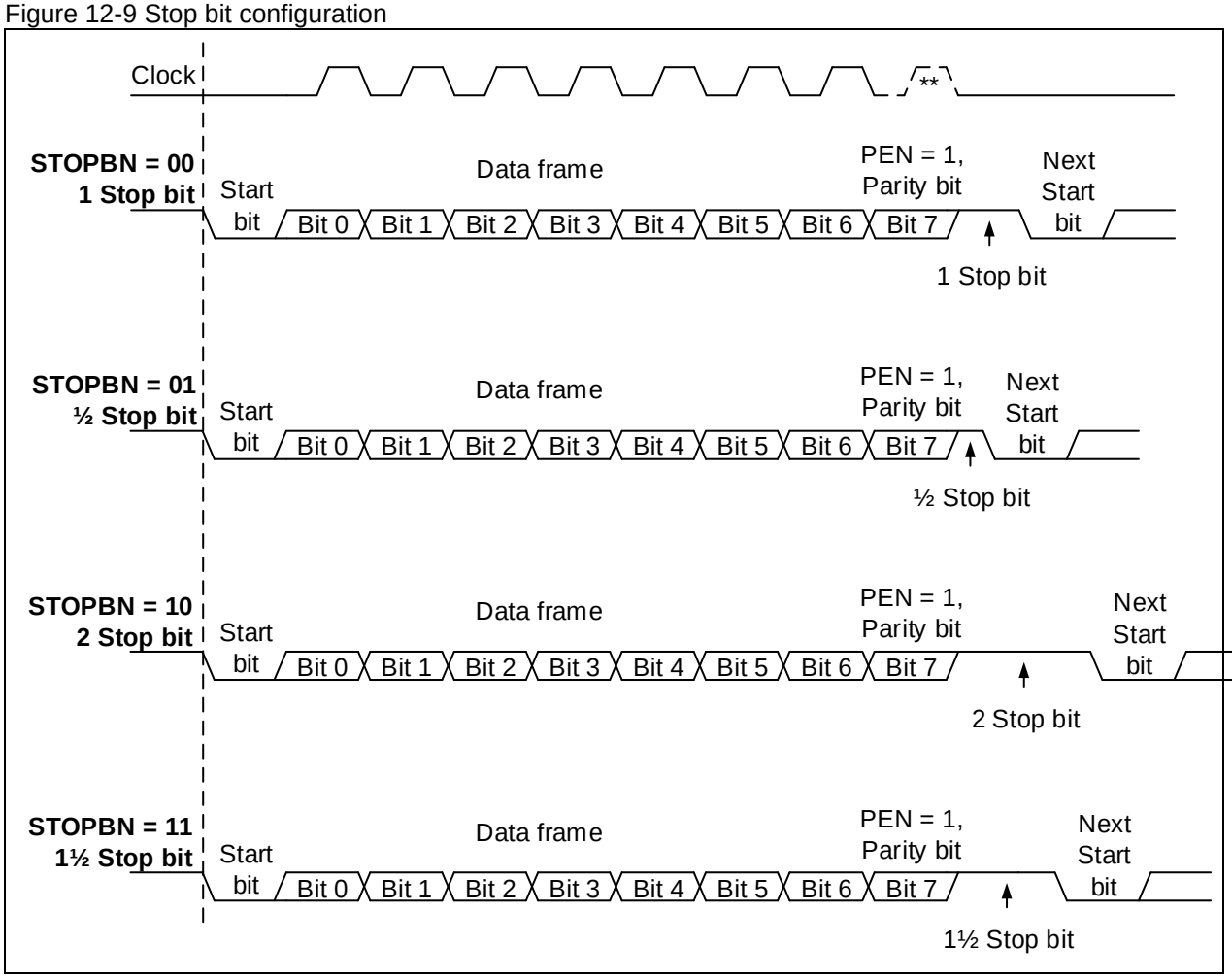
Figure 12-8 Word length configuration



The STOPBN bit is used to program 1-bit (STOPBN=00), 0.5-bit (STOPBN=01), 2-bit (STOPBN=10) and 1.5-bit (STOPBN=11) stop bits.

Setting PEN=1 will enable the parity control. PSEL=1 indicates odd parity, while PSEL=0 for even parity. Once the parity control bit is enabled, the MSB of the data bit will be replaced with parity bit, that is, the significant bits is reduced by one bit.

Figure 12-9 Stop bit configuration



Set the MTF bit to determine whether MSB (MTF=1) first or LSB (MTF=0) first.

Set USART_DT as 1=L,0=H (DTREV=1) or 0=L,1=H (DTREV=0) for transmission and reception by setting the DTREV bit.

Set the TXREV bit to select VDD=0/mark,Gnd=1/idle (TXREV=1) or VDD=1/idle,Gnd=0/mark (TXREV=0) for signal transmission on USART_TX pin.

Set the RXREV bit to select VDD=0/mark,Gnd=1/idle (RXREV=1) or VDD=1/idle,Gnd=0/mark (RXREV=0) for signal transmission on USART_RX pin.

12.5 DMA transfer introduction

Enable transmit data buffer and receive data buffer using DMA to achieve continuous high-speed transmission for USART, which is detailed in subsequent sections. For more information on specific DMA configuration, refer to DMA chapter.

12.5.1 Transmission using DMA

1. Select a DMA channel: Select a DMA channel from DMA channel map table described in DMA chapter.
2. Configure the destination of DMA transfer: Configure the USART_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
3. Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the USART_DT register from the memory address after transmit request is received by DMA.
4. Configure the total number of bytes to be transferred in the DMA control register.
5. Configure the channel priority of DMA transfer in the DMA control register.
6. Configure the DMA interrupt generation after half or full transfer in the DMA control register.
7. Enable DMA transfer channel in the DMA control register.

12.5.2 Reception using DMA

1. Select a DMA transfer channel: Select a DMA channel from DMA channel map table described in DMA chapter.
2. Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the USART_DT register to the programmed destination after reception request is received by DMA.
3. Configure the source of DMA transfer: Configure the USART_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the USART_DT register to the programmed destination after reception request is received by DMA.
4. Configure the total number of bytes to be transferred in the DMA control register.
5. Configure the channel priority of DMA transfer in the DMA control register.
6. Configure DMA interrupt generation after half or full transfer in the DMA control register.
7. Enable DMA transfer channel in the DMA control register.

12.6 Baud rate generation

12.6.1 Introduction

USART baud rate generator uses an internal counter based on PCLK. The DIV (USART_BAUDR[15:0] register) represents the overflow value of the counter. Each time the counter is full, it denotes one-bit data. Thus each data bit width refers to PCLK cycles x DIV.

The receiver and transmitter of USART share the same baud rate generator, and the receiver splits each data bit into 16 equal parts to achieve oversampling, so the data bit width should not be less than 16 PCLK periods, that is, the DIV value must be greater than or equal to 16.

12.6.2 Configuration

User can program the desired baud rate by setting different system clocks and writing different values into the USART_BAUDR register. The calculation format is as follows:

$$\frac{\text{TX}}{\text{RX}}\ baud\ \ rate = \frac{f_{CK}}{\text{DIV}}$$

Where, f_{CK} refers to the system clock of USART (PCLK1 for USART2, 3, 4, 5 and USART7, 8, and PCLK2 for USART1,6)

Note:

1. Write access to the USART_BAUDR register is performed before setting the UEN bit. The baud rate register values should not be altered when UEN=1.

2. When USART receiver or transmitter is disabled, the internal counter will be reset, and baud rate interrupt will occur.

Table 12-1 Error calculation for programmed baud rate

Baud rate		fPCLK=36MHz			fPCLK=72MHz		
No.	Kbps	Actual	Value programmed in the baud rate register	Error %	Actual	Value programmed in the baud rate register	Error %
1	2.4	2.4	15000	0%	2.4	30000	0%
2	9.6	9.6	3750	0%	9.6	7500	0%
3	19.2	19.2	1875	0%	19.2	3750	0%
4	57.6	57.6	625	0%	57.6	1250	0%
5	115.2	115.384	312	0.15%	115.2	625	0%
6	230.4	230.769	156	0.16%	230.769	312	0.16%
7	460.8	461.538	78	0.16%	461.538	156	0.16%
8	921.6	923.076	39	0.16%	923.076	78	0.16%
9	2250	2250	16	0%	2250	32	0%
10	4500	NA	NA	NA	4500	16	0%

If the baud rate is 115.2Kbps, when fPCLK=36MHz, the baud rate register should be set as 312(0x138), and the calculated result is 36000000 / 312 = 115384 = 115.384Kbps.

Error: (actual value – theoretical value) / theoretical value * 100%:(115.384 - 115.2) / 115.2 * 100% = 0.15%.

12.7 Transmitter

12.7.1 Introduction

USART transmitter has its individual TEN. The transmitter and receiver share the same baud rate that is programmable. There is a transmit data buffer (TDR) and a transmit shift register in the USART. The TDBE bit is set whenever the TDR is empty, and an interrupt is generated if the TDBEIEN is set.

The data written by software is stored in the TDR register. When the shift register is empty, the data will be moved from the TDR register to the shift register so that the data in the transmit shift register is output on the TX pin in LSB mode. The output format depends on the programmed frame format.

If synchronous transfer or clock output is selected, the clock pulse is output on the CK pin. If the hardware flow control is selected, the control signal is input on the CTS pin.

Note:

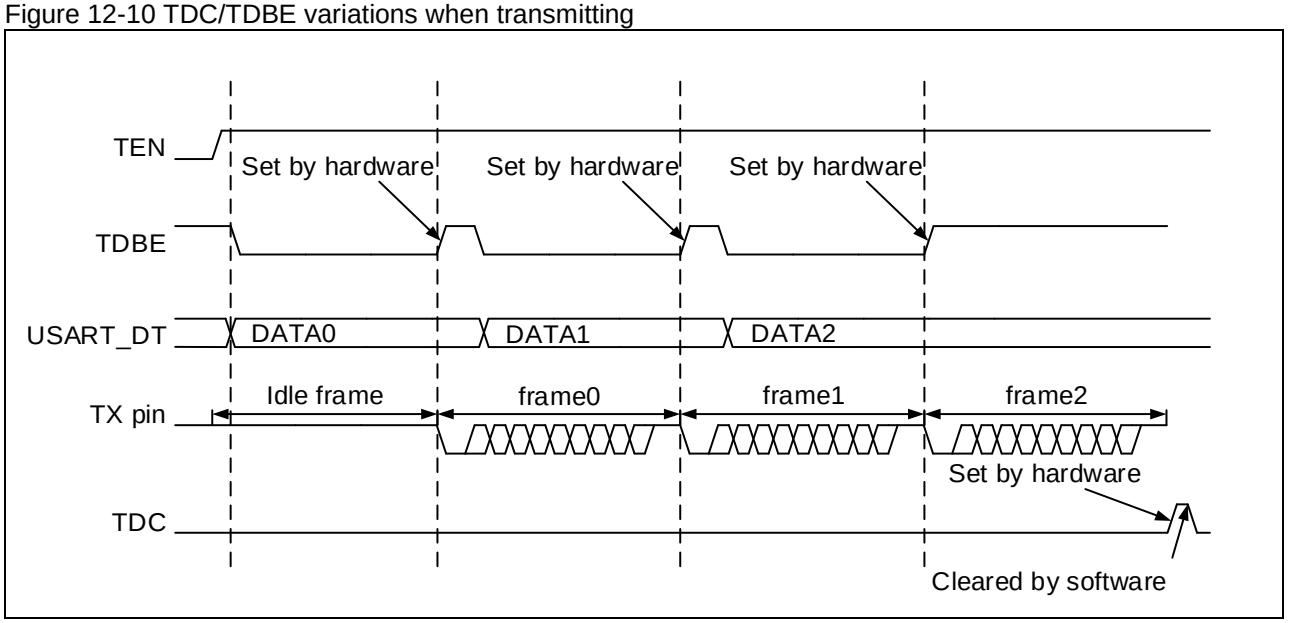
1. The TEN bit cannot be reset during data transfer, or the data on TX pin will be corrupted.
2. After the TEN bit is enabled, the USART will automatically send an idle frame.

12.7.2 Transmitter configuration

1. USART enable: set UEN=1.
2. Full-duplex/half-duplex configuration: Refer to full-duplex/half-duplex selector for more information (Section 12.2).
3. Mode configuration: Refer to mode selector for more information (Section 12.2).
4. Frame format configuration: Refer to frame format for more information (Section 12.4).
5. Interrupt configuration: Refer to interrupt generation for more information (Section 12.10).
6. DMA transmission configuration: If the DMA mode is selected, the DMATEN bit (bit [7] in the USART_CTRL3[7] register) is set, and configure DMA register accordingly.
7. Baud rate configuration: Refer to baud rate generation for details (Section 12.6).
8. Transmitter enable: When the TEN bit is set, the USART transmitter will send an idle frame.
9. Write operation: Wait until the TDBE bit is set, the data to be transferred will be loaded into the USART_DT register (this operation will clear the TDBE bit). Repeat this operation in non-DMA mode.

10. After the last data expected to be transferred is written, wait until the TDC bit is set, indicating the end of transfer. The USART cannot be disabled before the flag is set; otherwise, transfer error will occur.
11. When TDC=1, read access to the USART_STS register and write access to the USART_DT register will clear the TDC bit. This bit can also be cleared by writing “0”, but this is valid only in DMA mode.

Figure 12-10 TDC/TDBE variations when transmitting



12.8 Receiver

12.8.1 Introduction

USART receiver has its individual REN control bit (bit [2] in the USART_CTRL1 register). The transmitter and receiver share the same baud rate that is programmable. There is a receive data buffer (RDR) and a receive shift register in the USART.

The data is input on the RX pin of the USART. When a valid start bit is detected, the receiver ports the data received into the receive shift register in LSB mode. After a full data frame is received, based on the programmed frame format, it will be moved from the receive shift register to the receive data buffer, and the RDBF is set accordingly. An interrupt is generated if the RDBFIEN is set.

If hardware flow control is selected, the control signal is output on the RTS pin.

During data reception, the USART receiver will detect whether there are errors to occur, including framing error, overrun error, parity check error or noise error, depending on software configuration, and whether there are interrupts to generate using the interrupt enable bits.

12.8.2 Receiver configuration

Configuration procedure:

1. USART enable: set UEN=1.
2. Full-duplex/half-duplex configuration: Refer to full-duplex/half-duplex selector for more information (Section 12.2).
3. Mode configuration: Refer to mode selector for more information (Section 12.2).
4. Frame format configuration: Refer to frame format for more information (Section 12.4).
5. Interrupt configuration: Refer to interrupt generation for more information (Section 12.10).
6. Reception using DMA: If the DMA mode is selected, the DMAREN bit is set, and configure DMA register accordingly.
7. Baud rate configuration: Refer to baud rate generation for details (Section 12.6).
8. Receiver enable: set REN=1.

Character reception:

- The RDBF bit is set. It indicates that the content of the shift register is transferred to the RDR (Receiver Data Register). In other words, data is received and can be read (including its associated error flags).
- An interrupt is generated when the RDBFIEN bit is set.
- The error flag is set when a framing error, noise error or overrun error is detected during reception.
- In DMA mode, the RDBF bit is set after every byte is received, and it is cleared when the data register is read by DMA.
- In non-DMA mode, the RDBF bit is cleared when read access to the USART_DT register by software. The RDBF flag can also be cleared by writing “0” to it. The RDBF bit must be cleared before the end of next frame reception to avoid overrun error.

Break frame reception:

- Non-LIN mode: It is handled as a framing error, and the FERR is set. An interrupt is generated if the corresponding interrupt bit is enabled. Refer to framing error described below for details.
- LIN mode: It is handled as a break frame, and the BFF bit is set. An interrupt is generated if the BFIEN is set.

Idle frame reception:

- USART receiver processes this idle frame as a data frame, and the IDLEF bit is set. An interrupt is generated when the IDLEIEN is set.

When a framing error occurs:

- The FERR bit is set.
- The USART receiver moves the invalid data from the receive shift register to the receive data buffer.
- In non-DMA mode, both FERR and RDBF are set at the same time. The latter will generate an interrupt. In DMA mode, an interrupt is generated if the ERRIEN is set.

When an overrun error occurs:

- The ROERR bit is set.
- The data in the receive data buffer is not lost. The previous data is still available when the USART_DT register is read.
- The content in the receive shift register is overwritten. Afterwards, any data received will be lost.
- An interrupt is generated if the RDBFIEN bit is set or both ERRIEN and DMAREN bits are set.
- The ROERR bit is cleared by reading the USART_STS register and then reading the USART_DT register in order.

Note: If the ROERR bit is set, it indicates that at least one piece of data is lost, with two possibilities:
- If RDBF=1, it indicates that the last valid data is still stored in the receive data buffer and can be read.
- If RDBF=0, it indicates that the last valid data in the receive data buffer has already been read.

Note: The REN bit cannot be reset during data reception, or the byte that is currently being received will be lost.

12.8.3 Start bit and noise detection

A start bit detection occurs when the REN bit is set. With the oversampling techniques, the USART receiver samples data on the 3rd, 5th, 7th, 8th, 9th and 10th bits to detect the valid start bit and noise. The table below shows the data sampling over start bit and noise detection.

Table 12-2 Data sampling over start bit and noise detection

Sampled value (3·5·7)	Sampled value (8·9·10)	NERR bit	Start bit validity
000	000	0	Valid
001/010/100	001/010/100	1	Valid
001/010/100	000	1	Valid
000	001/010/100	1	Valid
111/110/101/011	Any value	0	Invalid
Any value	111/110/101/011	0	Invalid

Note: If the sampling values on the 3rd, 5th, 7th, 8th, 9th, and 10th bits do not match the above mentioned requirements, the USART receiver does not think that a correct start bit is received, and thus it will abort the start bit detection and return to idle state waiting for a falling edge.

The USART receiver has the ability to detect noise. In the non-synchronous mode, the USART receiver samples data on the 7th, 8th and 9th bits, with its oversampling techniques, to distinguish valid data input from noise based on different sampling values, and recovers data as well as sets NERR (Noise Error Flag) bit.

Table 12-3 Data sampling over valid data and noise detection

Sampled value	NERR bit	Received bit value	Data validity
000	0	0	Valid
001	1	0	Invalid
010	1	0	Invalid
011	1	1	Invalid
100	1	0	Invalid
101	1	1	Invalid
110	1	1	Invalid
111	0	1	Valid

USART is able to receive data under the maximum allowable deviation condition. Its value depends on the DBN[1:0] of the USART_CTRL1 register and the DIV[3: 0] of the USART_BAUDR register.

Note: The maximum allowable deviations stated in the table below are calculated based on 115.2Kbps. The actual deviations may vary with the settings of buad rate. In other words, the greater the buad rate is, the smaller the maximum allowable deviation; in contrast, when the baud rate gets smaller, the maximum allowable deviation will get bigger.

Table 12-4 Maximum allowable deviation

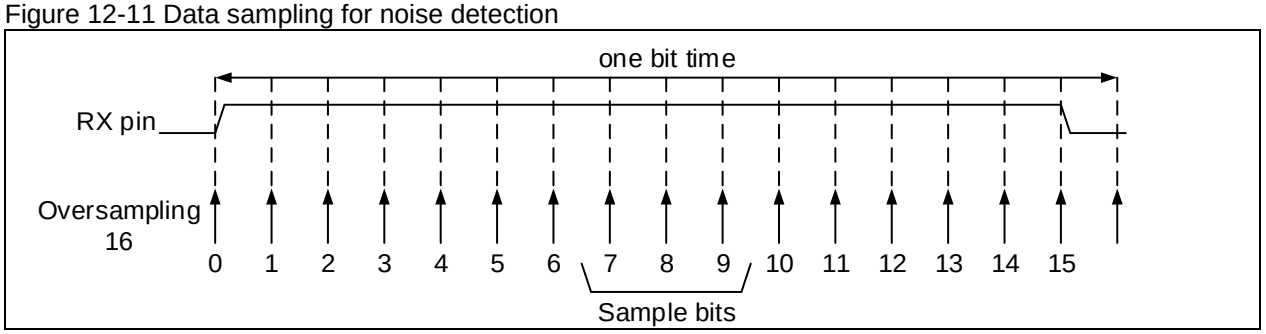
DBN[1:0]	DIV[3:0] = 0	DIV[3:0] != 0
00	3.75%	3.33%
01	3.41%	3.03%
10	4.16%	3.7%

When noise is detected in a data frame:

- The NERR bit is set at the same time as the RDBF bit.
- The invalid data is transferred from the receive shift register to the receive data buffer.
- No interrupt is generated in non-DMA mode. However, since the NERR bit is set at the same time as the RDBF bit, the RDBF bit will generate an interrupt. In DMA mode, an interrupt will be issued if the ERRIEN is set.

- The NERR bit is cleared by reading the USART_STS register and the reading the USART_DT register in order.

Figure 12-11 Data sampling for noise detection

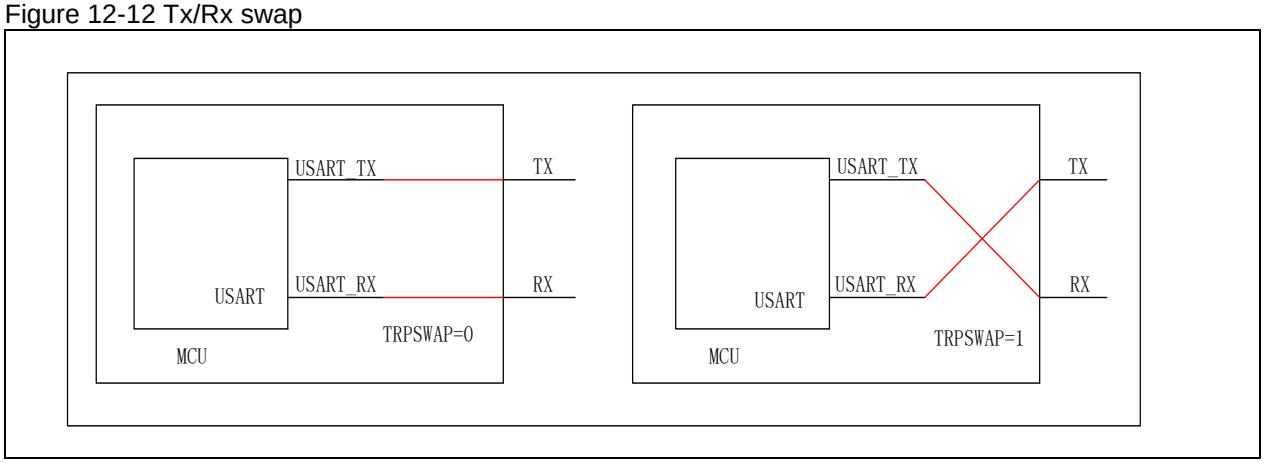


12.9 Tx/Rx swap

When the TRPSWAP (bit [15] in the USART_CTRL2 register) is set, Tx/Rx pins can be swapped. Two common scenes are listed below:

- If Tx/Rx are reversed while the user attempts to connect the device externally to a RS-232 chip, they can be swapped by setting the TRPSWAP bit, without the need of hardware intervention.
- If the user only connects the master Tx to the slave Rx in full-duplex mode, the Tx/Rx can be interchangeable with the TRPSWAP bit after the master and slave are swapped, without the need of hardware intervention.

Figure 12-12 Tx/Rx swap



Note: The SWAP (USART_CTRL2[15]) can be modified only when the USART is disabled (UEN=0).

12.10 Interrupts

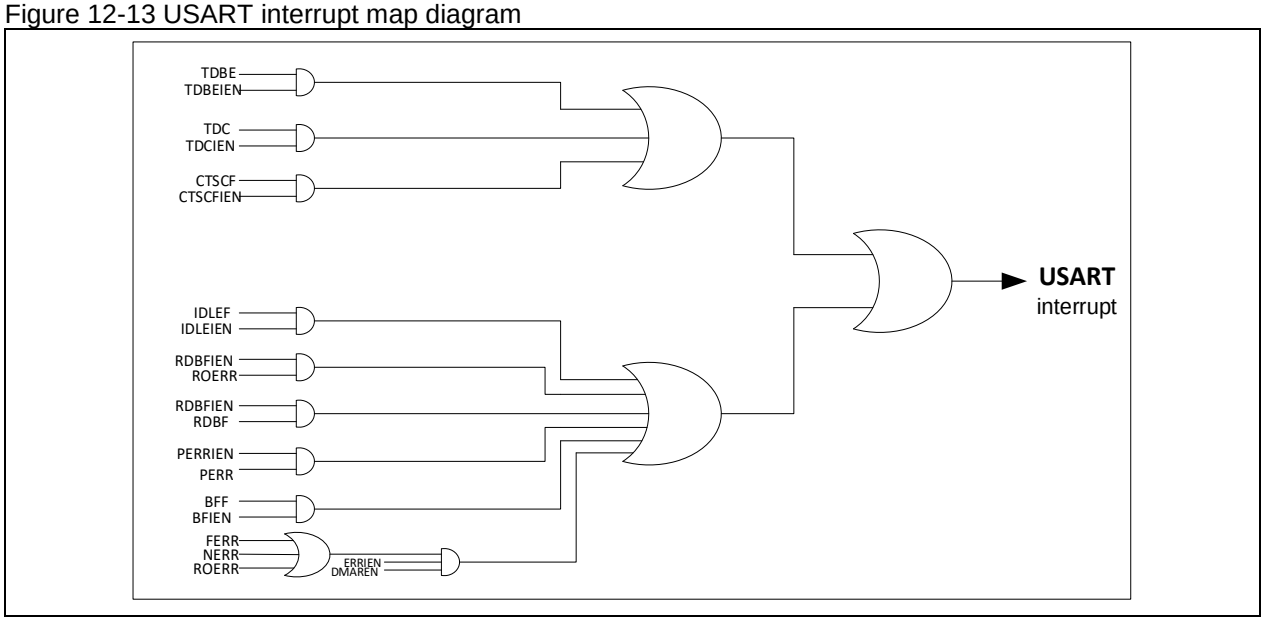
USART interrupt generator serves as a control center of USART interrupts. It is used to monitor the interrupt source inside the USART in real time and the generation of interrupts according to the programmed interrupt control bits. The table below shows the USART interrupt source and interrupt enable control bit. An interrupt will be generated over an event when the corresponding interrupt enable bit is set.

Table 12-5 USART interrupt requests

Interrupt event	Event flag	Enable bit
Transmit data register empty	TDBE	TDBEIEN
CTS flag	CTSCF	CTSCFIEN
Transmit complete	TDC	TDCIEN
Receive data buffer full	RDBF	RDBFIEN
Receive overflow error	ROERR	
Idle flag	IDLEF	IDLEIEN
Parity check	PERR	PERRIEN

	Break frame flag	BFF	BFIEN
	Noise error, overflow error or framing error	NERR, ROERR or FERR	ERRIEN (1)
	Receiver timeout detection	RTODF	RTODIE
	Byte match detection	CMDF	CMDIE

Figure 12-13 USART interrupt map diagram



12.11 I/O pin control

The following five interfaces are used for USART communication.

RX: Serial data input.

TX: Serial data output. In single-wire half-duplex mode and Smartcard mode, the TX pin is used as an I/O for data transmission and reception.

CK: Transmitter clock output. The output CLK phase, polarity and frequency are programmable.

CTS: Transmitter input. Send enable signal in hardware flow control mode.

RTS: Receiver output. Send request signal in hardware flow control mode.

12.12 USART registers

Table 12-6 USART register map and reset value

Register	Offset	Reset value
USART_STS	0x00	0x00C0
USART_DT	0x04	0x0000
USART_BAUDR	0x08	0x0000
USART_CTRL1	0x0C	0x0000
USART_CTRL2	0x10	0x0000
USART_CTRL3	0x14	0x0000
USART_GDIV	0x18	0x0000
USART_RTOV	0x1C	0x0000
USART_IFC	0x20	0x0000

12.12.1 Status register (USART_STS)

Bit	Name	Reset value	Type	Description
Bit 31:18 Bit 16:12 Bit 10	Reserved	0x000000	resd	Forced to 0 by hardware.
Bit 17	CMDF	0	r	Byte match detection flag This bit is set by hardware when the byte defined by ID[7:0] is received. It is cleared by software. 0: No byte received 1: Byte received
Bit 11	RTODF	0	r	Receiver timeout detection flag This bit is set by hardware when the timeout value reaches the programmed value in RTOV register and without any communication. It is cleared by software. 0: No timeout detected 1: Timeout detected
Bit 9	CTSCF	0x0	rw0c	CTS change flag This bit is set by hardware when the CTS status line changes. It is cleared by software. 0: No change on the CTS status line 1: A change occurs on the CTS status line
Bit 8	BFF	0x0	rw0c	Break frame flag This bit is set by hardware when a break frame is detected. It is cleared by software. 0: No break frame is detected 1: A break frame is detected
Bit 7	TDBE	0x1	ro	Transmit data buffer empty This bit is set by hardware when the transmit data buffer is empty. It is cleared by a write operation to the USART_DT register. 0: Transmit data buffer is not empty 1: Transmit data buffer is empty
Bit 6	TDC	0x1	rw0c	Transmit data complete This bit is set by hardware at the end of transmission. It is cleared by software (Option 1: Read access to the USART_STS and then write operation to the USART_DT; Option 2: Write “0” to this bit) 0: Transmission is not completed 1: Transmission is completed
Bit 5	RDBF	0x0	rw0c	Receive data buffer full This bit is set by hardware when the buffer receives data. It is cleared by software (Option 1: Read access to the USART_DT; Option 2: Write “0” to this bit) 0: Data is not received 1: Data is received
Bit 4	IDLEF	0x0	ro	Idle flag This bit is set by hardware when an idle line is detected. It is cleared by software (read access to the USART_STS and then the USART_DT). 0: No idle line is detected 1: Idle line is detected
Bit 3	ROERR	0x0	ro	Receiver overflow error This bit is set by hardware when the data is received while the RDBF is still set. It is cleared by software (read access to the USART_STS and then the USART_DT).

				0: No overflow error 1: Overflow error is detected *Note: When this bit is set, the DT register content will not be lost, but the subsequent data will be overwritten.*
Bit 2	NERR	0x0	ro	Noise error This bit is set by hardware when noise is detect on a received frame. It is cleared by software (read access to the USART_STS and then the USART_DT). 0: No noise is detected 1: Noise is detected
Bit 1	FERR	0x0	ro	Framing error This bit is set by hardware when a stop bit error (low level), excessive noise or break frame is detected. It is cleared by software (read access to the USART_STS and then the USART_DT). 0: No framing error is detected 1: Framing error is detected
Bit 0	PERR	0x0	ro	Parity error This bit is set by hardware when parity error occurs. It is cleared by software (read access to the USART_STS and then the USART_DT). 0: No parity error occurs 1: Parity error occurs

12.12.2 Data register (USART_DT)

Bit	Name	Reset value	Type	Description
Bit 31:9	Reserved	0x000000	resd	Forced to 0 by hardware.
Bit 8:0	DT	0x000	rw	Data value This register provides read and write function. When transmitting with the parity bit enabled, the value written in the MSB bit will be replaced by the parity bit. When receiving with the parity bit enabled, the value in the MSB bit is the received parity bit.

12.12.3 Baud rate register (USART_BAUDR)

Note: If TEN and REN bits are disabled, the baud counter stops counting.

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Forced to 0 by hardware.
Bit 15:0	DIV	0x0000	rw	Divider This field defines the USART divider.

12.12.4 Control register 1 (USART_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31:29	Reserved	0x00000	resd	Forced to 0 by hardware.
Bit 28	DBN1	0x0	rw	Data bit number This bit, along with the DBN0 bit, is used to program the number of data bits. 10: 7 data bits 00: 8 data bits

				01: 9 data bits 11: Write operation forbidden *Note: When parity check is enabled, the valid data bit is reduced by one and the MSB of data bit is replaced by the parity enable bit.*
Bit 27	RTODEN	0	rw	Receiver time out detection enable 0: Disabled 1: Enabled
Bit 26	RTODIE	0	rw	Receiver time out detection interrupt enable 0: Disabled 1: Enabled
Bit 25:21	TSDT	0x00	rw	Transmit start delay time In RS485 mode, the first data (in sequential transmit mode) is transmitted after a period of time of being written so as to ensure that the transfer direction of the external transmitter/receiver switches back to transmit. This time depends on the TSDT value, in unit of 1/16 baud rate.
Bit 20:16	TCDT	0x00	rw	Transmit complete delay time In RS485 mode, a period of time (delay) is needed before the last data transfer is complete even if the last STOP bit has been transferred. This time duration allows the transfer direction of the external receiver/transmitter to switch back to receive. This time depends on the TCDT value, in unit of 1/16 baud rate.
Bit 15	Reserved	0	resd	Kept at its default value.
Bit 14	CMDIE	0	rw	Character match detection interrupt enable 0: Disabled 1: Enabled
Bit 13	UEN	0x0	rw	USART enable 0: Disabled 1: Enabled
Bit 12	DBN0	0x0	rw	Data bit number This bit, along with DBN1, is used to program the number of data bits. 10: 7 data bits 00: 8 data bits 01: 9 data bits 11: Write operation forbidden *Note: When parity check is enabled, the valid data bit is reduced by one and the MSB of data bit is replaced by the parity enable bit.*
Bit 11	WUM	0x0	rw	Wakeup mode This bit determines the way to wake up from silent mode. 0: Waken up by idle line 1: Waken up by ID match
Bit 10	PEN	0x0	rw	Parity enable

				This bit is used to enable hardware parity control (generation of parity bit for transmission; detection of parity bit for reception). When this bit is enabled, the MSB bit of the transmitted data is replaced with the parity bit; check whether the parity bit of the received data is correct. 0: Disabled 1: Enabled
Bit 9	PSEL	0x0	rw	Parity selection This bit selects the odd or even parity after the parity control is enabled. 0: Even parity 1: Odd parity
Bit 8	PERRIEN	0x0	rw	PERR interrupt enable 0: Disabled 1: Enabled
Bit 7	TDBEIEN	0x0	rw	TDBE interrupt enable 0: Disabled 1: Enabled
Bit 6	TDCIEN	0x0	rw	TDC interrupt enable 0: Disabled 1: Enabled
Bit 5	RDBFIEN	0x0	rw	RDBF interrupt enable 0: Disabled 1: Enabled
Bit 4	IDLEIEN	0x0	rw	IDLE interrupt enable 0: Disabled 1: Enabled
Bit 3	TEN	0x0	rw	Transmitter enable This bit enables the transmitter. 0: Disabled 1: Enabled
Bit 2	REN	0x0	rw	Receiver enable This bit enables the receiver. 0: Disabled 1: Enabled
Bit 1	RM	0x0	rw	Receiver mute This bit determines if the receiver is in mute mode or not. It is set or cleared by software. When the idle line is used to wake up from mute mode, this bit is cleared by hardware after wake up. When the address match is used to wake up from mute mode, it is cleared by hardware after wake up. When address mismatches, this bit is set by hardware to enter mute mode again. 0: Receiver is in active mode 1: Receiver is in mute mode

Bit 0	SBF	0x0	rw	Send break frame This bit is used to send a break frame. It can be set or cleared by software. Generally speaking, it is set by software and cleared by hardware at the end of break frame transmission. 0: No break frame is transmitted 1: Break frame is transmitted

12.12.5 Control register 2 (USART_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31:28	IDH	0x0	rw	USART identification This field holds the upper four bits of USART ID. It is configurable.
Bit 27:20	Reserved	0x000	resd	Kept at its default value.
Bit 19	MTF	0	rw	MSB transmit first This bit is used to select MSB transmit first or LSB transmit first. 0: LSB 1: MSB *Note: Parity check is not available when MTF is enabled.*
Bit 18	DTREV	0	rw	DT register polarity reverse 0: 1=H, 0=L 1: 1=L, 0=H
Bit 17	TXREV	0	rw	TX polarity reverse 0: VDD=1/idle,Gnd=0/mark 1: VDD=0/mark,Gnd=1/idle
Bit 16	RXREV	0	rw	RX polarity reverse 0: VDD=1/idle,Gnd=0/mark 1: VDD=0/mark,Gnd=1/idle
Bit 15	TRPSWAP	0x0	rw	Transmit/receive pin swap 0: Transmit/receive pin is not swappable 1: Transmit/receive pin is swappable
Bit 14	LINEN	0x0	rw	LIN mode enable 0: Disabled 1: Enabled
Bit 13:12	STOPBN	0x0	rw	STOP bit number These bits are used to program the number of stop bits. 00: 1 stop bit 01: 0.5 stop bit 10: 2 stop bits 11: 1.5 stop bits
Bit 11	CLKEN	0x0	rw	Clock enable This bit is used to enable the clock pin for synchronous mode or Smartcard mode. 0: Disabled

				1: Enabeld
Bit 10	CLKPOL	0x0	rw	Clock polarity In synchronous mode or Smartcard mode, this bit is used to select the polarity of the clock output on the clock pin in idle state. 0: Clock output low 1: Clock output high
Bit 9	CLKPHA	0x0	rw	Clock phase This bit is used to select the phase of the clock output on the clock pin in synchronous mode or Smartcard mode. 0: Data capture is done on the first clock edge 1: Data capture is done on the second clock edge
Bit 8	LBCP	0x0	rw	Last bit clock pulse This bit is used to select whether the clock pulse of the last data bit transmitted is output on the clock pin in synchronous mode. 0: The clock pulse of the last data bit is not output on the clock pin 1: The clock pulse of the last data bit is output on the clock pin
Bit 7	Reserved	0x0	resd	Kept at its default value.

Bit 6	BFIEN	0x0	rw	Break frame interrupt enable 0: Disabled 1: Enabled
Bit 5	BFBN	0x0	rw	Break frame bit number This bit is used to select 11-bit or 10-bit break frame. 0: 10-bit 1: 11-bit
Bit 4	IDBN	0	rw	Identification bit number This bit is used to select ID bit number. 0: 4-bit 1: Data bit -1 bit *N*ote: When this bit is set, in 7/8/9-bit data mode, the ID bit number is the lower 6/7/8-bit (or the 5/6/7-bit if parity check is enabled), respectively.
Bit 3:0	IDL	0x0	rw	USART identification This field holds the lower four bits of USART ID. It is configurable.

Note: The CLKPOL, CLKPHA and LBCP bits should not be changed while the transmission is enabled.

12.12.6 Control register 3 (USART_CTRL3)

Bit	Name	Reset value	Type	Description
Bit 31:16 Bit 13:11	Reserved	0x000000	resd	Forced to 0 by hardware.
Bit 15	DEP	0	rw	DE polarity selection 0: High level active 1: Low level active
Bit 14	RS485EN	0	rw	RS485 enable This bit is used to enable RS485 mode. In RS485 mode, the USART controls the transfer direction of the external receiver/transmitter through the DE signal. 0: RS485 mode disabled. The control signal DE output is disabled. RTS pin is used in RS232 mode. 1: RS485 mode enabled. The control signal DE outputs on the RTS pin.
Bit 10	CTSCFIEN	0x0	rw	CTSCF interrupt enable 0: Disabled 1: Enabled
Bit 9	CTSEN	0x0	rw	CTS enable 0: Disabled 1: Enabled
Bit 8	RTSEN	0x0	rw	RTS enable 0: Disabled 1: Enabled

Bit 7	DMATEN	0x0	rw	DMA transmit enable 0: Disabled 1: Enabled
Bit 6	DMAREN	0x0	rw	DMA receiver enable 0: Disabled 1: Enabled
Bit 5	SCMEN	0x0	rw	Smartcard mode enable 0: Disabled 1: Enabled
Bit 4	SCNACKEN	0x0	rw	Smartcard NACK enable This bit is used to send NACK when parity error occurs. 0: NACK is disabled when parity error occurs 1: NACK is enabled when parity error occurs
Bit 3	SLBEN	0x0	rw	Single-line bidirectional half-duplex enable 0: Disabled 1: Enabled

Bit 2	IRDALP	0x0	rw	IrDA low-power mode This bit is used to configure IrDA low-power mode. 0: Disabled 1: Enabled
Bit 1	IRDAEN	0x0	rw	IrDA enable 0: Disabled 1: Enabled
Bit 0	ERRIEN	0x0	rw	Error interrupt enable An interrupt is generated when a framing error, overflow error or noise error occurs. 0: Disabled 1: Enabled

12.12.7 Guard time and divider register (USART_GDIV)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Forced to 0 by hardware.
Bit 15:8	SCGT	0x00	rw	Smartcard guard time This field specifies the guard time value. The transmission complete flag is set after this guard time in Smartcard mode.
Bit 7:0	ISDIV	0x00	rw	IrDA/Smartcard division In IrDA mode: Bits [7:0] are valid which is invalid in common mode and must be set to 00000001. In low-power mode, it divides the peripheral clock to serve as the period base of the pulse width. 00000000: Reserved – Do not write. 00000001: Divided by 1 00000010: Divided by 2 …… In Smartcard mode: The lower five bits [4:0] are valid. This division is used to divide the peripheral clock to provide clock for the Smartcard. Configured as follows: 00000: Reserved – Do not write. 00001: Divided by 2 00010: Divided by 4 00011: Divided by 6 ……

12.12.8 Receiver timeout detection register (USART_RTOV)

Bit	Name	Reset value	Type	Description
Bit 31:24	Reserved	0x00	resd	Forced to 0 by hardware.
Bit 23:0	RTOV	0x00	rw	Receiver timeout value The unit is 1-bit width.

12.12.9 Interrupt flag clear register (USART_IFC)

Bit	Name	Reset value	Type	Description
Bit 31:18 Bit 16:12 Bit 10:0	Reserved	0x00	resd	Forced to 0 by hardware.
Bit 17	CMDFC	0	w1	Character match detection flag clear
Bit 11	RTODFC	0	w1	Receiver timeout detection flag clear

13 Serial peripheral interface (SPI)

13.1 SPI introduction

The SPI interface supports either the SPI protocol or the I²S protocol, depending on software configuration. This chapter gives an introduction of the main features and configuration procedure of SPI used as SPI or I²S.

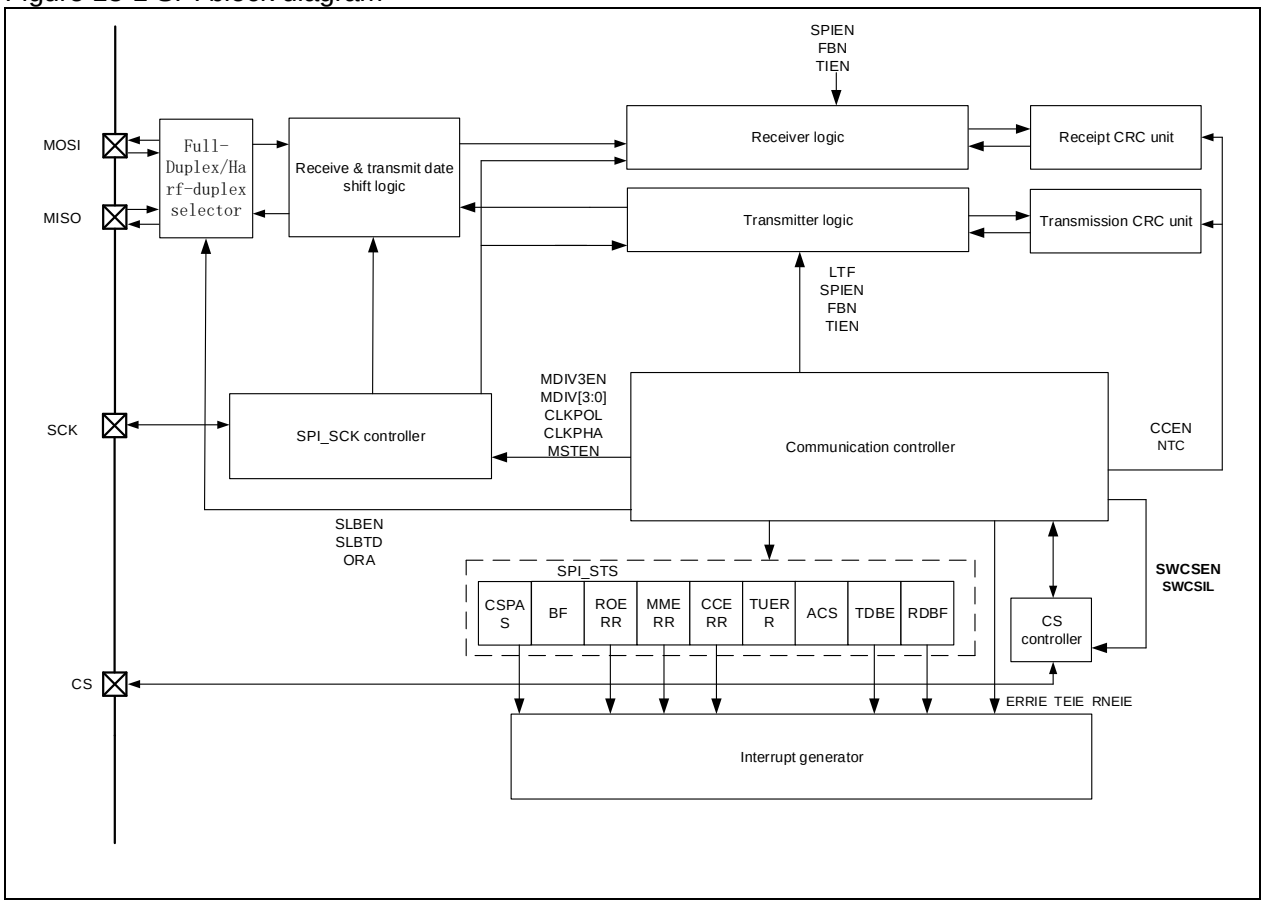
13.2 Functional overview

13.2.1 SPI description

The SPI can be configured as host or slave based on software configuration, supporting full-duplex, reception-only full-duplex and transmission-only/reception-only half-duplex modes, DMA transfer, and automatic CRC calibration and verification of SPI internal hardware. In the meantime, the SPI interface can be compatible with the TI protocol through software configurations.

SPI block diagram:

Figure 13-1 SPI block diagram



Main features as SPI:

- Full-duplex or half-duplex communication
  - Full-duplex synchronous communication (supporting reception-only mode to release IO for transmission purpose)
  - Half-duplex synchronous communication (transfer direction is configurable: receive or transmit)
- Master or slave mode
- CS signal processing mode
  - CS signal processing by hardware
  - CS signal processing by software
- 8-bit or 16-bit frame format

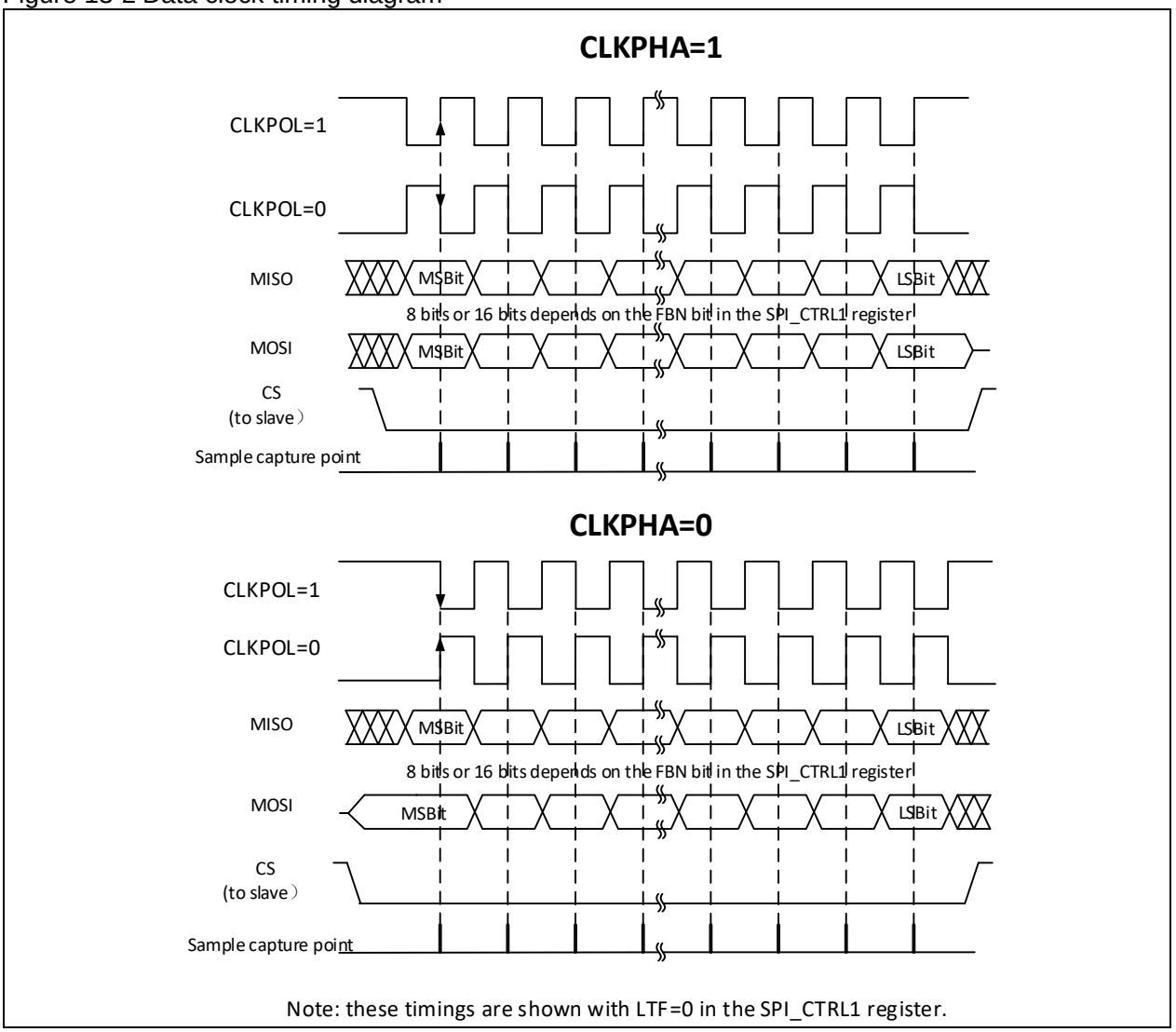
- Communication frequency and prescalers (prescalers up to $f_{PCLK}/2$)
- Programmable clock polarity and phase
- Programmable data transfer order (MSB-first or LSB-first)
- Programmable error interrupt flags (CS pulse error, receiver overflow error, master mode error and CRC error)
- Programmable transmit data buffer empty interrupt and receive data buffer full interrupt
- Support transmission and reception using DMA
- Support hardware CRC transmission and error checking
- Busy status flag
- Compatible with the TI protocol

Clock phase and clock polarity

Four possible timing relationships may be chosen by setting the CLKPOL and CLKPHA bits in the SPI_CTRL1 register. The CLKPOL (clock polarity) bit controls the steady state value of the clock when no data is being transferred. This bits affects both master and slave modes. if CLKPOL is cleared, the SCK pin has a low-level idle state. If CLKPOL is set to 1, the SCK pin has a high-level idle state.

If the CLKPHA (clock phase) bit is set to 1, the second edge on the SCK pin (falling edge if the CLKPOL bit is reset, rising edge if the CLKPOL bit is set) will sample data bits. Data are latched on the occurrence of the second lock transition. If the CLKPHA bit is reset, the first edge on the SCK pin (falling edge if CLKPOL bit is set, rising edge if CLKPOL bit is reset) will sample data bits. Data are latched on the occurrence of the first clock transition.

Figure 13-2 Data clock timing diagram



Note: these timings are shown with LTF=0 in the SPI_CTRL1 register.

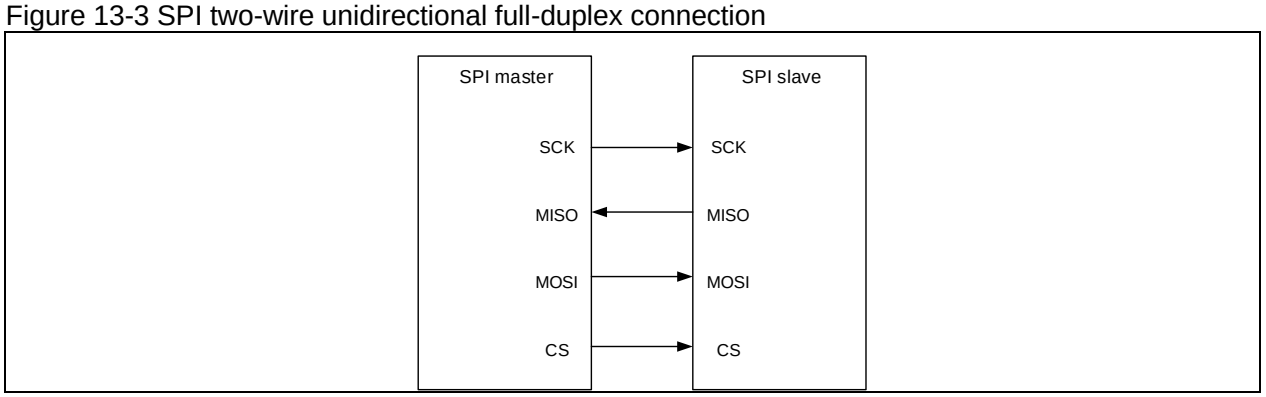
13.2.2 Full-duplex/half-duplex selector

When used as an SPI interface, it supports four synchronous modes: two-wire unidirectional full-duplex, single-wire unidirectional receive only, single-wire bidirectional half-duplex transmit and single-wire bidirectional half-duplex receive.

Figure 13-3 shows the two-wire unidirectional full-duplex mode and SPI IO connection:

The SPI operates in two-wire unidirectional full-duplex mode when the SLBEN bit and ORA bit are both 0. In this case, the SPI supports data transmission and reception at the same time. IO connection is as follows:

Figure 13-3 SPI two-wire unidirectional full-duplex connection



In either master or slave mode, it is required to wait until the RDBF bit and TDBE bit is set, and BF=0 before disabling the SPI or entering power-saving mode (or disabling SPI system clock).

Figure 13-4 shows the single-wire unidirectional receive-only mode and SPI IO connection

The SPI operates in single-wire unidirectional receive-only mode when the SLBEN is 0 and the ORA is set. In this case, the SPI can be used only for data reception (transmission is not supported). The MISO pin transmits data in slave mode and receives data in master mode. The MOSI pin transmits data in master mode and receives data in salve mode.

Figure 13-4 Single-wire unidirectional receive only in SPI master mode

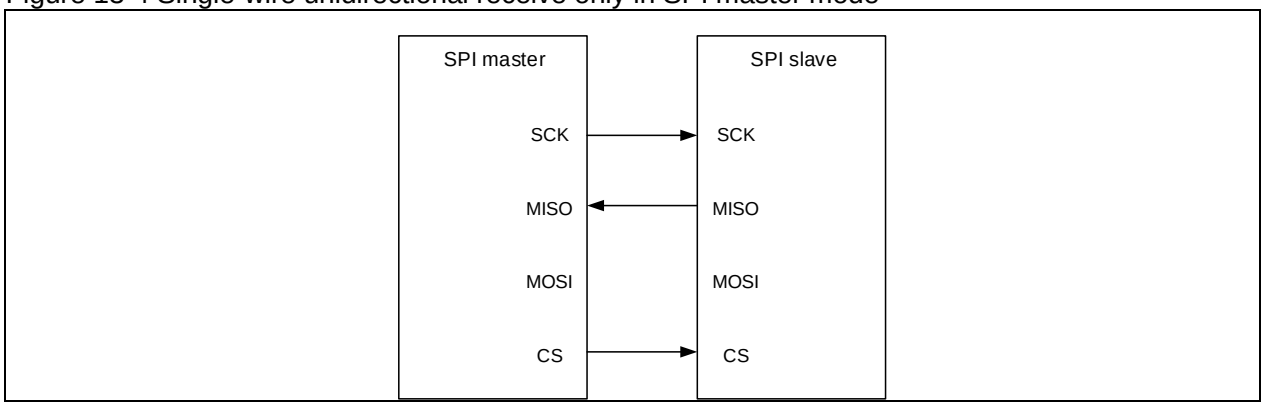
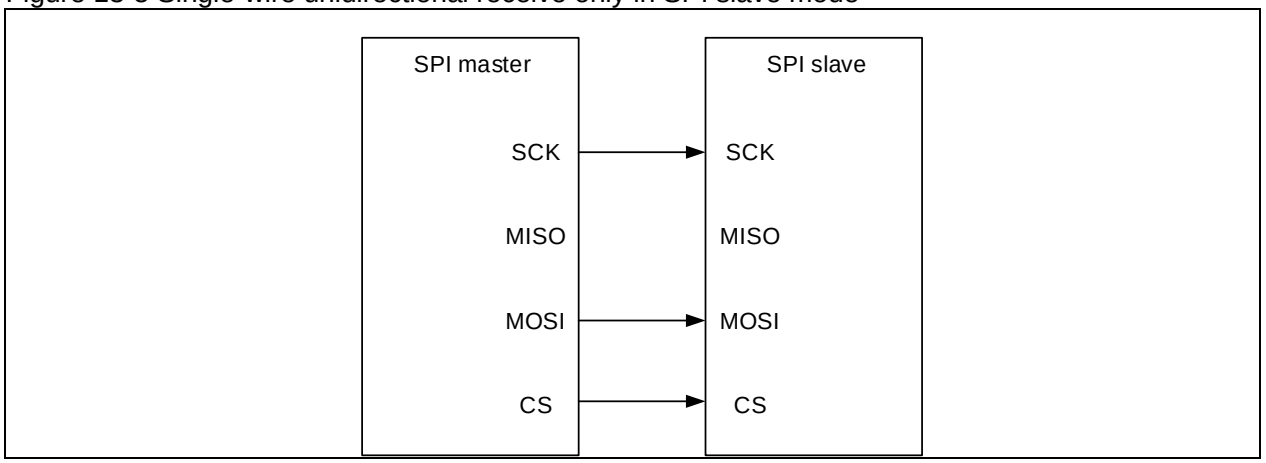


Figure 13-5 Single-wire unidirectional receive only in SPI slave mode



In master mode, it is necessary to wait until the second-to-last RDBF bit is set and then another SPI_CPK period before disabling the SPI. The last RDBF must be set before entering power-saving mode (or disabling SPI system clock).

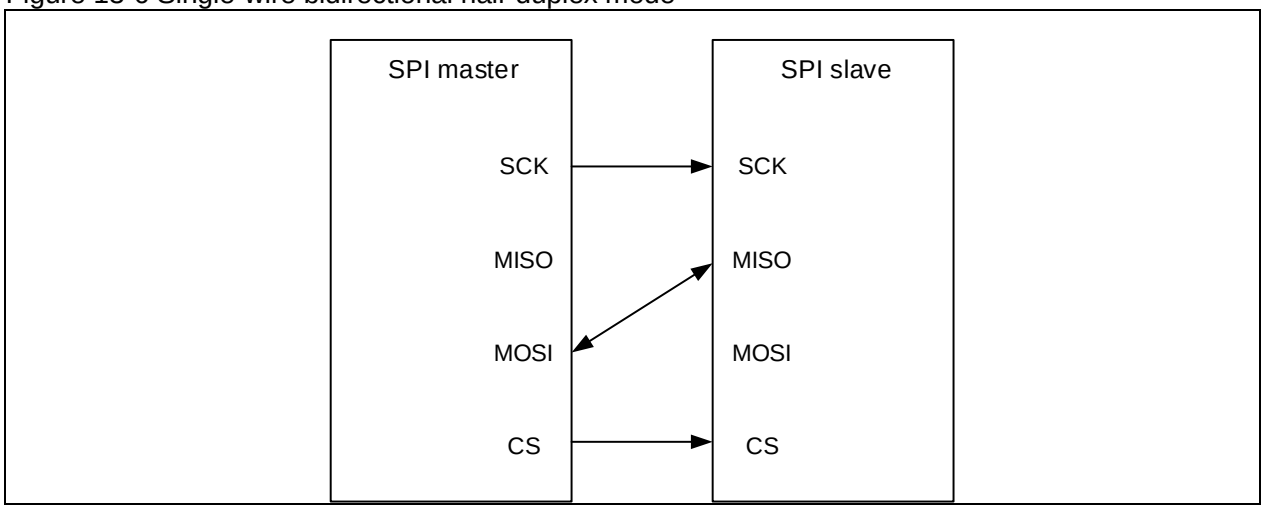
In slave mode, there is no need to check any flag before disabling the SPI. However, it is required to wait until the BF becomes 0 before entering power-saving mode.

Figure 13-6 shows single-wire bidirectional half-duplex mode and SPI IO connection

When the SLBEN is set, the SPI operates in single-wire bidirectional half-duplex mode. In this case, the SPI supports data reception and transmission alternately. In master mode, the MOSI pin transmits or receives data in master mode, while the MISO pin is released. In slave mode, the MISO pin transmits or receives data, but the MOSI pin is released.

The SLBTD bit is used by software to configure transfer direction. When the SLBTD bit is set, the SPI can be used only for data transmission; when the SLBTD bit is 0, the SPI can be used only for data reception.

Figure 13-6 Single-wire bidirectional half-duplex mode



When the SPI is selected for data transmission in single-wire bidirectional half-duplex mode (master or slave), the TDBE bit must be set, and the BF must be 0 before disabling the SPI. The power-saving mode (or disabling SPI system clock) cannot be entered unless the SPI is disabled.

In master mode, when the SPI is selected for data reception in single-wire bidirectional half-duplex mode, it is required to wait until the second-to-last RDBF is set and then another SPI_SCK period before disabling the SPI. And the last RDBF must be set before entering power-saving mode (or disabling SPI system clock).

In slave mode, when the SPI is selected for data reception in single-wire bidirectional half-duplex mode, there is no need to check any flags before disabling the SPI. However, the BT must be 0 before entering power-saving mode (or disabling SPI system clock).

13.2.3 Chip select controller

The Chip select controller (CS) is used to enable hardware or software control for chip select signals through software configuration. This controller is used to select master/slave device in multi-processor mode, and to avoid conflicts on the data lines by enabling the SCK signal output followed by CS signal. The hardware and software configuration procedure is detailed as follows, along with their respective input/output in master and slave mode.

CS hardware configuration procedure:

In master mode with CS being as an output, HWCSOE=1, SWCSEN=0, the CS hardware control is enabled. If the SPI is enabled, low level is output on the CS pin. The CS signal is then released after the SPI is disabled and the transmission is complete.

In master mode with CS being as an input, HWCSOE=0, SWCSEN=0, the CS hardware control is enabled. At this point, the SPI is automatically disabled by hardware and enters slave mode as soon as the CS pin low is detected by master SPI. The mode error flag (MMERR bit) is set at the same time. An interrupt is generated if ERRIE=1. When the MMERR is set, the SPIEN and MSTEN cannot be set by software. The MMERR is cleared by read or write access to the SPI_STS register followed by write operation to the SPI_CTRL1 register.

In slave mode with CS being as an input, HWCSOE=0, SWCSEN=0, the CS hardware control is enabled. The slave selects whether to transmit / receive data based on the level on the CS pin. The slave is selected for data reception and transmission only when the CS pin is low.

CS software configuration procedure:

In master mode with CS being as an input, SWCSEN=1, the CS software control is enabled. When SWCSIL=0, the SPI is automatically disabled by hardware and enters slave mode. The mode error flag (MMERR bit) is set at this time. An interrupt is generated if ERRIE=1. When the MMERR bit is set, the SPIEN and MSTEN bits cannot be set by software. The MMERR bit is cleared by read or write access to the SPI_STS register followed by write operation to the SPI_CTRL1 register.

In slave mode with CS being as an input, SWCSEN=1, the CS software control is enabled. The SPI judges the CS signal with the SWCSIL bit, instead of CS pin. When SWCSIL=0, the slave is selected for data reception and transmission.

13.2.4 SPI_SCK controller

The SPI protocol adopts synchronous transmission. In master mode with the SPI being used as SPI, it is required to generate a communication clock for data reception and transmission on the SPI, and the communication clock should be output to the slave via IO for data reception and transmission. In slave mode, the communication clock is provided by peripherals, and is input to the SPI via IO. In all, the SPI_SCK controller is used for the generation and distribution of SPI_SCK, with the configuration procedure detailed as follows:

SPI_SCK controller configuration procedure:

- Clock polarity and clock phase selection: It is selected by setting the CLKPOL and CLKPHA bit.
- Clock prescaler selection: Select the desired PCLK frequency by setting the CRM bit. Select the desired prescaler by setting the MDIV[3: 0] bit or .
- Master/slave selection: Select SPI as master or slave by setting the MSTEN bit.

Note that the clock output is activated after the SPI is enabled in master reception-only mode, and it remains output until when the SPI is disabled and the reception is complete.

13.2.5 CRC overview

There is an independent transmission and reception CRC calculation unit in the SPI. When used as SPI through software configuration, the SPI enables CRC calculation and CRC check automatically while the user is reading or writing through DMA or CPU. During the transmission, if the received data is not consistent with, detected by hardware, the data in the SPI_RCRC register, and such data is exactly the CRC value, then the CCERR bit will be set. An interrupt is generated if ERRIE=1.

The CRC function and configuration procedure of the SPI are described as follows.

CRC configuration procedure

- CRC calculation polynomial is configured by setting the SPI_CPOLY register.
- CRC enable: The CRC calculation is enabled by setting the CCEN bit. This operation will reset the SPI_RCRC and SPI_TCRC registers.
- Select if or when the NTC bit is set, depending on DMA or CPU data register. See the following descriptions.

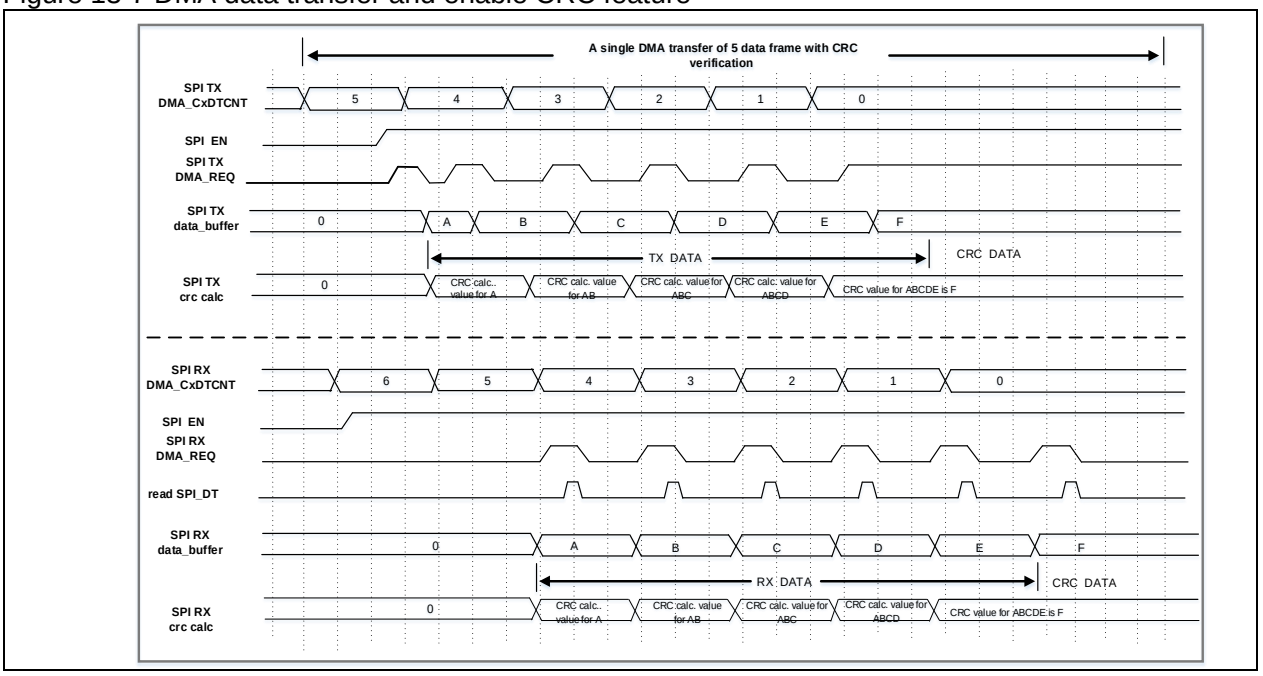
Transmission using DMA

When DMA is used to write the data to be transmitted, if the CCEN bit is enabled, the hardware calculates the CRC value automatically according to the value in the SPI_CPOLY register and each transmitted data, and sends the CRC value at the end of the last data transmission. This result is regarded as the value of the SPI_TCRC register.

Reception using DMA

When DMA is used to read the data to be received, if the CCEN bit is enabled, the hardware calculates the CRC value automatically according to the value in the SPI_CPOLY register and each received data, and waits until the completion of CRC data reception at the end of the last data reception before comparing the received CRC value with the value of the SPI_RCRC register. If check error occurs, the CCERR flag is set. An interrupt is generated if the ERRIE bit is enabled.

Figure 13-7 DMA data transfer and enable CRC feature



Transmission using CPU

Unlike DMA mode, after writing the last data to be transmitted, the CPU mode requires the NTC bit to be set by software before the end of the last data transmission.

Reception using CPU

In two-wire unidirectional full-duplex mode, follow CPU transmission mode to operate the NTC bit, the CRC calculation and check in CPU reception mode will be completed automatically.

In single-wire unidirectional reception-only mode and single-wire bidirectional reception-only mode, it is required to set the NTC bit before the software receives the last data when the second-to-last data is received.

13.2.6 DMA transfers

The SPI supports write and read operations with DMA. Refer to the following configuration procedure.

Special attention: when the CRC calculation and check is enabled, the number of data transferred by DMA is configured as the number of the data to be transferred. The number of data read with DMA is configured as the number of the data to be received. In this case, the hardware will send CRC automatically at the end of full transfer, and the receiver will also perform CRC check. Note that the received CRC data will be moved into the SPI_DT register by hardware, with the RDBF being set, and the DMA read request will be sent if then DAM transfer is enabled. Hence, it is recommended to read the SPI_DT register to get the CRC value at the end of CRC reception in order to avoid the upcoming transfer error.

Transmission with DMA

- Select DMA channel: Select a DMA channel for the current SPI transfer from DMA flexible request map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the SPI_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
- Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the SPI_DT register from the memory address after transmit request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register.
- Enable DMA transfer channel in the DMA control register.

Reception with DMA

- Select DMA transfer channel: Select a DMA channel for the current SPI from DMA flexible request map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the SPI_DT register to the programmed destination after reception request is received by DMA.
- Configure the source of DMA transfer: Configure the SPI_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the SPI_DT register to the programmed destination after reception request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register
- Enable DMA transfer channel in the DMA control register.

13.2.7 TI mode

The SPI interface is compatible with the TI protocol. The TI mode is enabled by setting the TIEN bit.

In this mode, the SPI interface will generates a communication clock SPI_CLK in accordance with the TI protocol. This means that the SPI_CLK polarity and phase are forced to conform to the TI protocol requirements, without the need of the intervention of CLKPOL and CLKPHA bits. Thus the CLKPOL and CLKPHA bits cannot be used to change the polarity and phase of the SPI_CLK either.

In this mode, the SPI interface will generate a CS signal in accordance with the TI protocol, meaning that the CS input and output are forced to conform to the TI protocol requirements, without the need of the intervention of SWCSEN, SWCSIL and HWCSOE bits. Thus, the SWCSEN, SWCSIL and HWCSOE bits cannot be used for CS signal management either.

In slave mode, once the TI mode is enabled, the SPI slave controls the MISO pin only during data transmission, meaning that the MISO pin state remains Hi-Z in idle state.

In slave mode, once the TI mode is enabled, the SPI interface is capable of detecting CS pulse errors during data transmission, and setting the CSPAS bit (It is cleared by reading the SPI_STS) as soon as a CS pulse error is detected. At this point, the detected pulse error will be discarded by the SPI. However, since there is something wrong with the CS signal, the software should disable the SPI slave and re-configure the SPI master before re-enabling the SPI slave for communication.

13.2.8 Transmitter

The SPI transmitter is clocked by SPI_SCK controller. It can output different data frame formats, depending on software configuration. There is a SPI_DT register available in the SPI that is used to be written with the data to be transmitted. When the transmitter is clocked, the contents in the SPI_DT register are copied into the data buffer (Unlike SPI_DT, it is driven by SPI_SCK, and controlled by hardware, instead of software), and sent out in order based on the programmed frame format.

Both DMA and CPU can be used for write operation. For DMA transfer, refer to DMA transfer section for more details. For CPU transfer, attention should be paid to the TDBE bit. The reset value of this bit is 1, indicating that the SPI_DT register is empty. If the TDBEIE bit is set, an interrupt is generated. After the data is written, the TDBE is pulled low until the data is moved to the transmit data buffer before the TDBE is set once again. This means that the user can be allowed to write the data to be transmitted only when the TDBE is set.

After the transmitter is configured and the SPI is enabled, the SPI is ready for data transmission. Before going forward, it is necessary for the users to refer to full-duplex / half-duplex chapter to get detailed configuration information, go to the Chip select controller chapter for specific chip select mode, check the SPI_SCK controller chapter for information on communication clock, and refer to CRC and DMA transfer chapter to configure CRC and DMA (if necessary). The recommended configuration procedure are as follows.

Transmitter configuration procedure:

- Configure full-duplex/half-duplex selector
- Configure chip select controller

- Configure SPI_SCK controller
- Configure CRC (if necessary)
- Configure DMA transfer (if necessary)
- If the DMA transfer mode is not used, the software will check whether to enable transmit data interrupt (TDBEIE =1) through the TDBE bit.
- Configure frame format: select MSB/LSB mode with the LTF bit, and select 8/16-bit data with the FBN bit
- Enable SPI by setting the SPIEN

13.2.9 Receiver

The SPI receiver is clocked by the SPI_SCK controller. It can output different data frame formats through software configuration. There is a receive data buffer register, driven by the SPI_SCK, in the SPI receiver.

At the last CLK of each transfer, the data is moved from the shift register to the receive data buffer register. Then the transmitter sets the receive data complete flag to the SPI logic. When the flag is detected by the SPI logic, the data in the receive data buffer is copied into the SPI_DT register, with the RDBF being set. This means that the data is received, and it is already stored into the SPI_DT. In this case, read access to the SPI_DT register will clear the RDBF bit.

Both DMA and CPU can be used for read operation. For DMA transfer, refer to DMA transfer section for more details. For CPU transfer, attention should be paid to the RDBE bit. The reset value of this bit is 0, indicating that the SPI_DT register is empty. If the data is received and moved into the SPI_DT, the RDBF is set, meaning that there are some data to be read in the SPI_DT register. An interrupt is generated if the RDBFIE bit is set.

When the next received data is ready to be moved to the SPI_DT register, if the previous received data is still not read (RDBF=1), then the data overflow occurs. The previous receive data is not lost, but the next received data will do. At this point, the ROERR is set. An interrupt is generated if the ERRIE is set. Read SPI_DT register and then the SPI_STS register will clear the ROERR bit. The recommended configuration procedure is as follows.

Receiver configuration procedure:

- Configure full-duplex/half-duplex selector
- Configure chip select controller
- Configure SPI_SCK controller
- Configure CRC (if necessary)
- Configure DMA transfer (if necessary)
- If the DMA transfer mode is not used, the software will check whether to enable receive data interrupt (RDBEIE =1) through the RDBE bit.
- Configure frame format: select MSB/LSB mode with the LTF bit, and select 8/16-bit data with the FBN bit
- Enable SPI by setting the SPIEN

13.2.10 Motorola mode

This section describes the SPI communication timings, which includes full-duplex and half-duplex master/slave timings.

Full-duplex communication – master mode

Configured as follows:

MSTEN=1: Master enable

SLBEN=0: Full-duplex mode

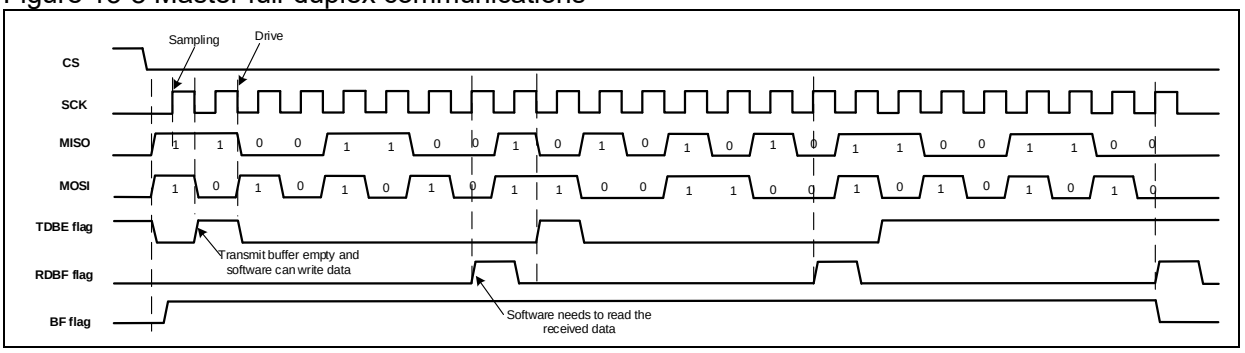
CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa

Slave transmit (MISO): 0xcc, 0xaa, 0xcc

Figure 13-8 Master full-duplex communications



Full-duplex communication – slave mode

Configured as follows:

MSTEN=0: Slave enable

SLBEN=0: Full-duplex mode

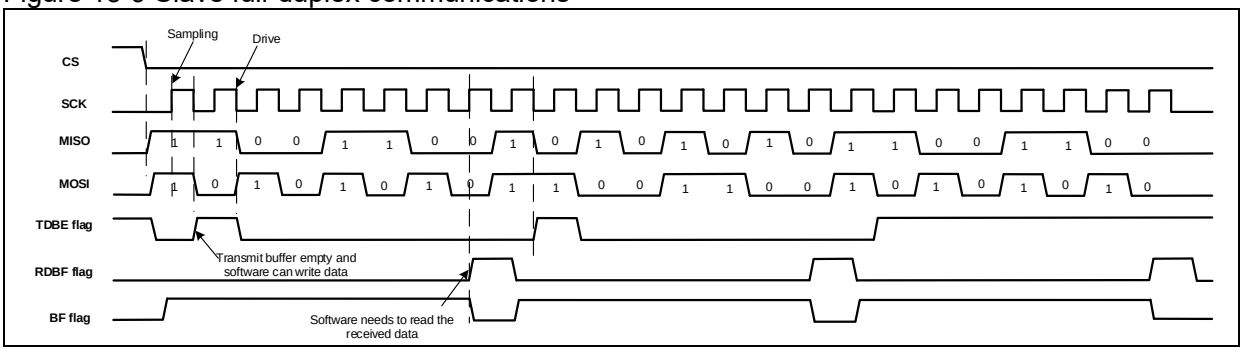
CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa

Slave transmit (MISO): 0xcc, 0xaa, 0xcc

Figure 13-9 Slave full-duplex communications



Half-duplex communication – master transmit

Configured as follows:

MSTEN=1: Master enable

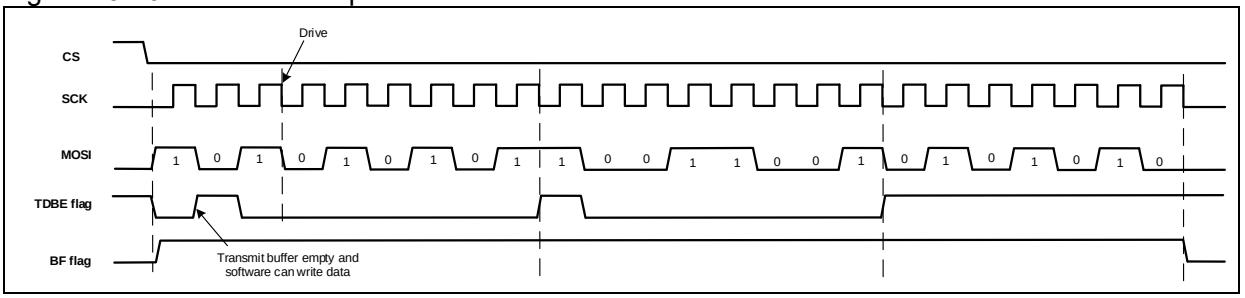
SLBEN=1: Single line bidirectional mode

CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Master transmit (MOSI): 0xaa, 0xcc, 0xaa

Figure 13-10 Master half-duplex transmit



Half-duplex communication – slave receive

Configured as follows:

MSTEN=0: Slave enable

SLBEN=1: Single line bidirectional mode

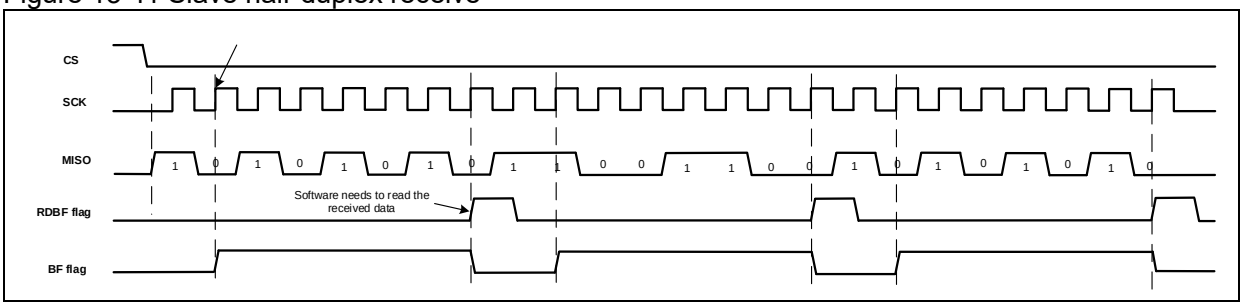
SLBTD=0: Receive mode

CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Slave receive: 0xaa, 0xcc, 0xaa

Figure 13-11 Slave half-duplex receive



Half-duplex communication – slave transmit

Configured as follows:

MSTEN=0: Slave enable

SLBEN=1: Single line bidirectional mode

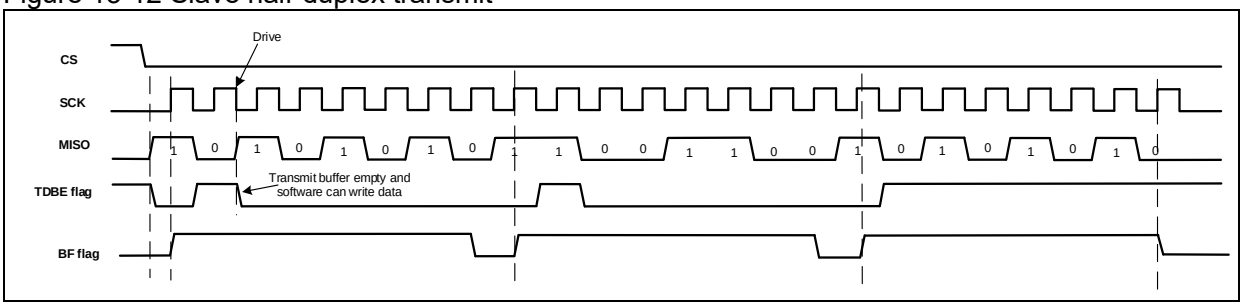
SLBTD=1: Transmit enable

CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Slave transmit: 0xaa, 0xcc, 0xaa

Figure 13-12 Slave half-duplex transmit



Half-duplex communication – master receive

Configured as follows:

MSTEN=1: Master enable

SLBEN=1: Single line bidirectional mode

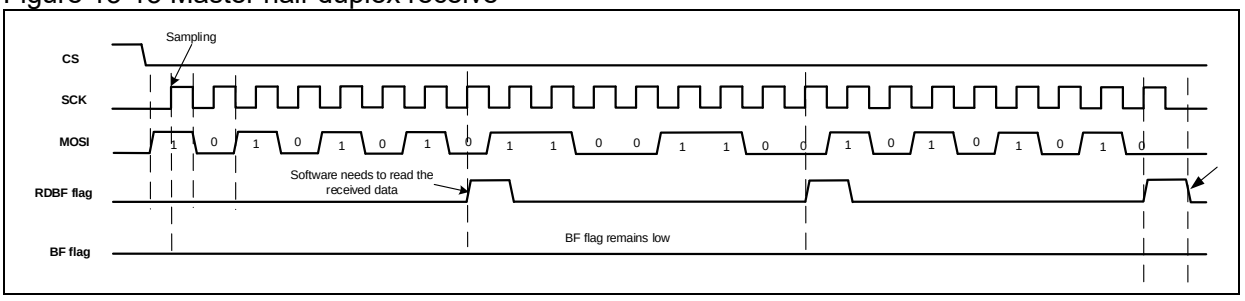
SLBTD=0: Receive enable

CLKPOL=0, CLKPHA=0: SCK idle output low, use the first edge for sampling

FBN=0: 8-bit frame

Master receive: 0xaa, 0xcc, 0xaa

Figure 13-13 Master half-duplex receive

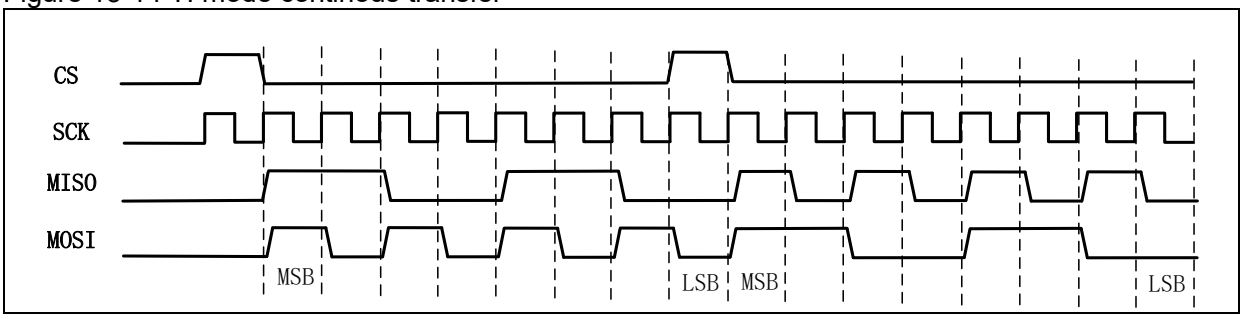


13.2.11 TI mode

The SPI interface supports TI mode. The TIEN bit can be set to enable SPI TI mode.

In TI mode, a bit of difference is present between continuous and discontinuous communication timings. When the to-be transmitted data is written before the rising SCK edge corresponding to the last data of the current transmit frame, it is a continuous communication, without dummy CLK between data, and the host sends a valid CS pulse while transmitting the last data of the current frame.

Figure 13-14 TI mode continous transfer



In TI mode, when the to-be-transmitted data is written between the rising and falling SCK edge corresponding to the last data of the current transmit frame, a dummy CLK exists between data.

Figure 13-15 TI mode continous transfer with dummy CLK

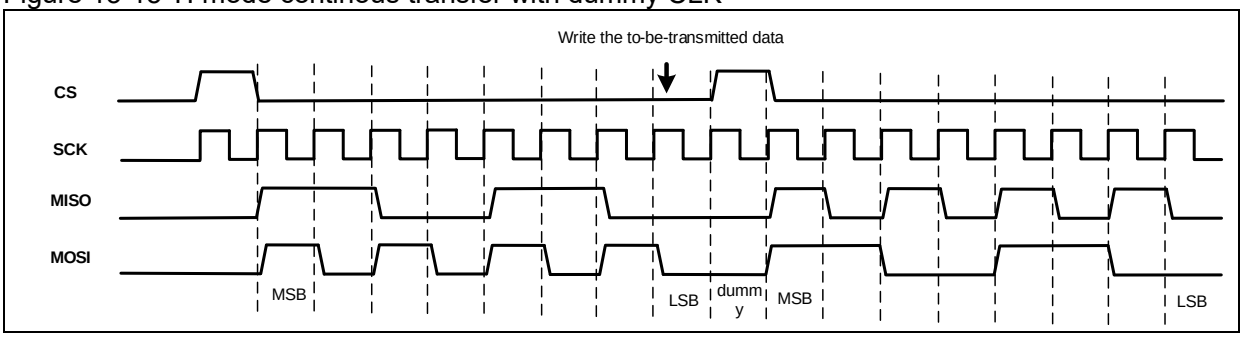
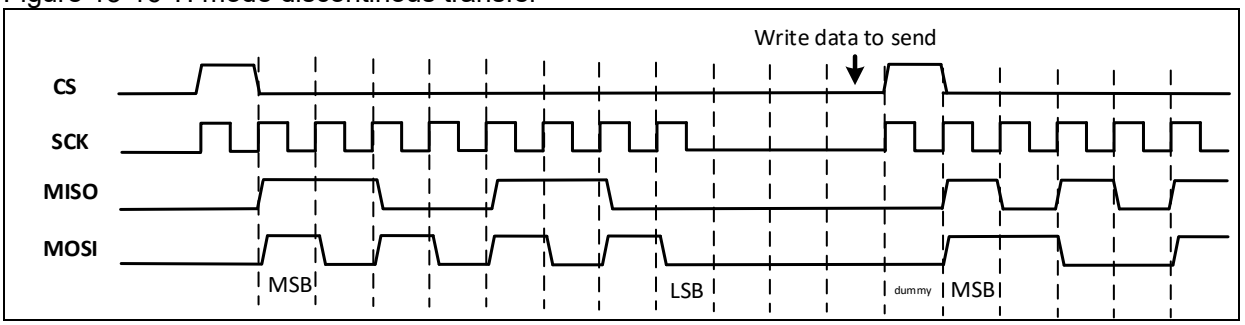
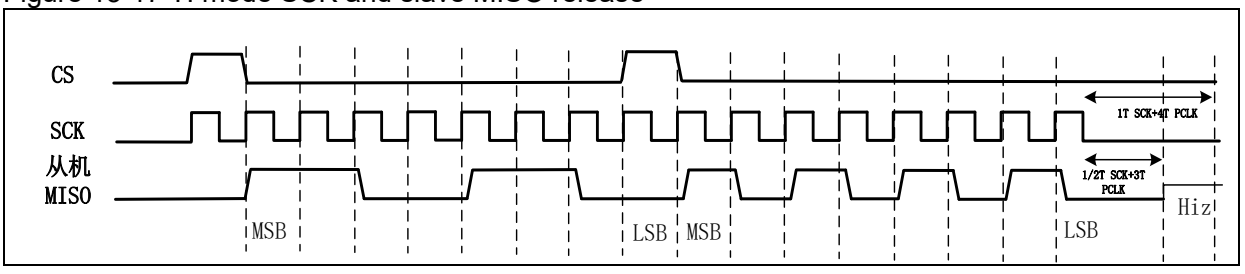


Figure 13-16 TI mode discontinous transfer



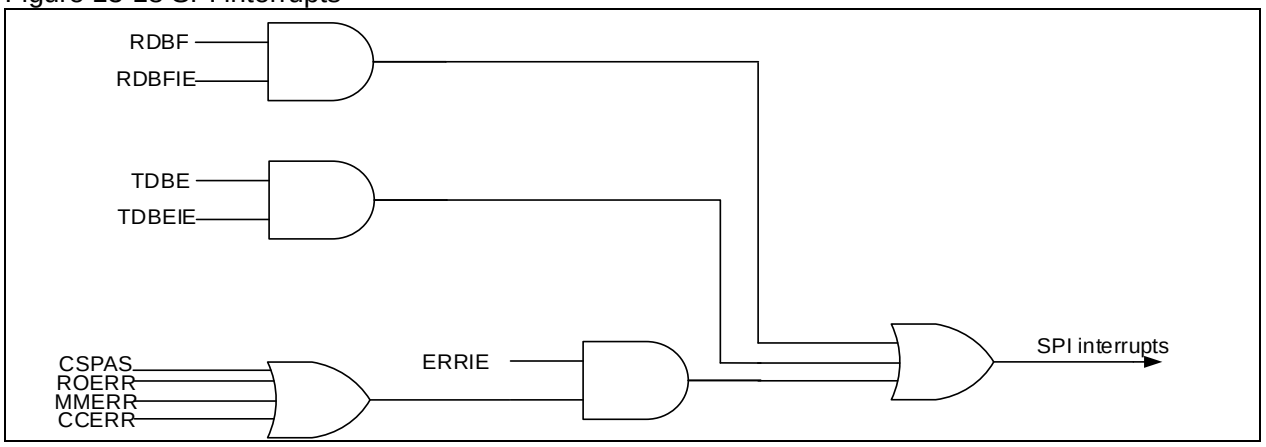
In TI mode, when the to-be-transmitted data is written after the falling SCK edge corresponding to the last data of the current transmit frame, the host always issues a valid SCK clock after 1T SCK + 4T PCLK. If the slave still does not detect a valid CS pulse at the end of the current data reception, it disables MISO output after 1/2T SCK + 3T PCLK to control MISO floating.

Figure 13-17 TI mode SCK and slave MISO release



13.2.12 Interrupts

Figure 13-18 SPI interrupts



13.2.13 IO pin control

Usually, the SPI is connected to external devices through four pins.

MISO: Master In/Slave Out. The pin receives data in master mode, and transmits data in slave mode. When SPI interface is in SPI host mode, this pin is used to iput data from a slave; when SPI in SPI slave mode, this pin is used to output data from the slave.

MOSI: Master Out/Slave In. The pin transmits data in master mode, and receives data in slave mode. When SPI interface is in SPI host mode, this pin is used to output data from a host; when in SPI slave mode, this pin is used to input data from the host.

SCK: SPI communication clock. The pin serves as output in master mode, and input in slave mode. When SPI interface is in SPI host mode, a communication clock is output to a peripheral via this pin; when in SPI slave mode, a communication clock from a host is intput into SPI interface.

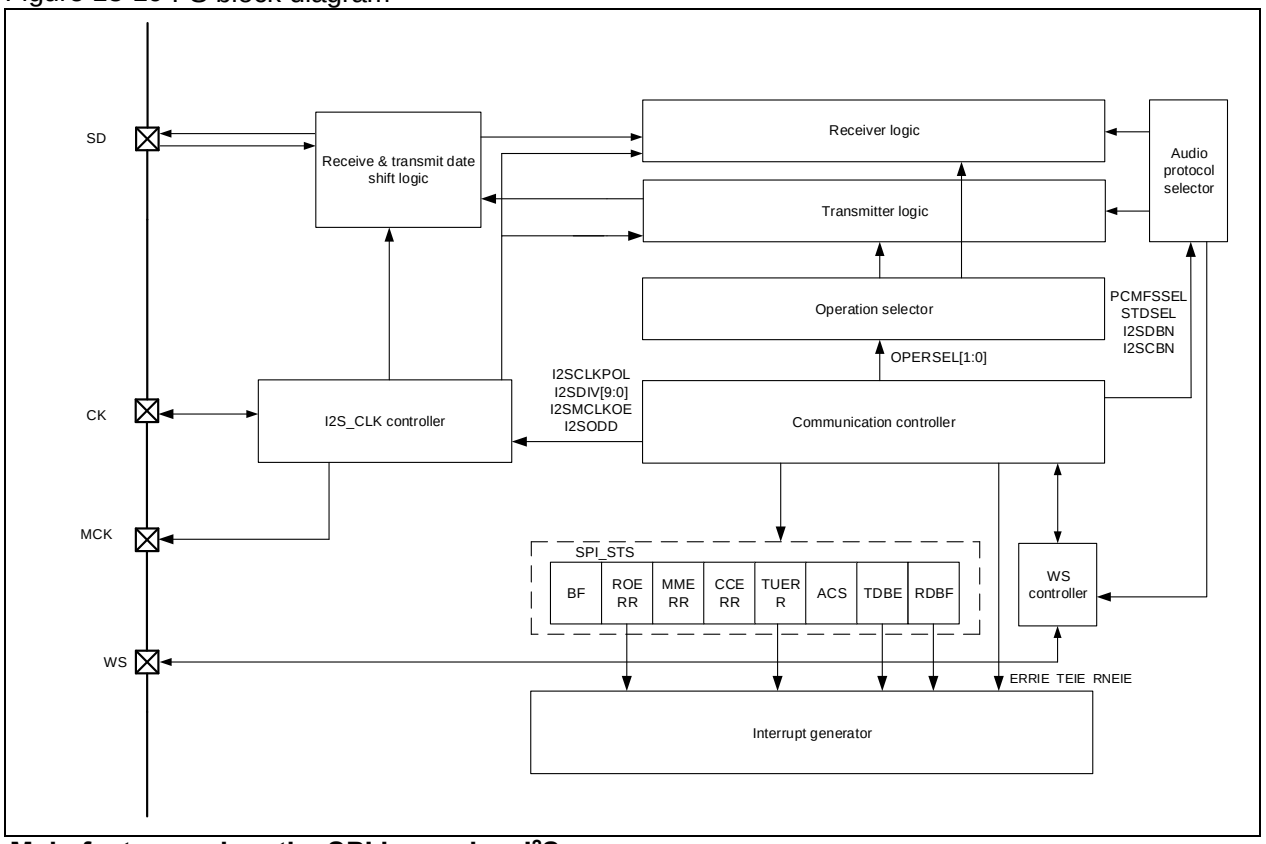
CS: Chip Select. This is an optional pin which selects master/slave mode.

13.3 I²S functional description

13.3.1 I²S introduction

The I²S can be configured by software as master reception/transmission, and slave reception/transmission, supporting four kinds of audio protocols including Philips standard, MSB-aligned standard, LSB-aligned standard and PCM standard, respectively. The DMA transfer is also supported.

Figure 13-19 I²S block diagram



Main features when the SPI is used as I²S:

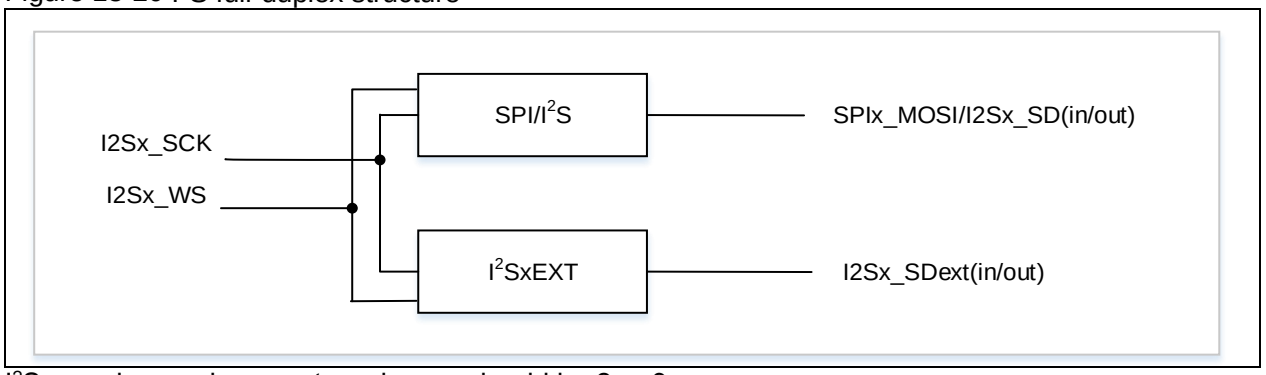
- Programmable operation mode
  - Slave device transmission
  - Slave device reception
  - Master device transmission
  - Master device reception
- Programmable clock polarity
- Programmable clock frequency (8 KHz to 192 KHz)
- Programmable data bits (16 bist, 24 bits, 32 bits)
- Programmable channel bits (16 bits, 32 bits)
- Programmable audio protocol
  - I²S Philips standard
  - MSB-aligned standard (left-aligned)
  - LSB-aligned standard (right-aligned)
  - PCM standard (long or short frame)
- DMA transfer
- Main peripheral clock with a fixed frequency of 256x Fs (audio sampling frequency)

13.3.2 I²S full-duplex

Two extra I²S modules (I²S2EXT and I²S3EXT) are used to support I²S full-duplex mode, combining the I²S2 with the I²S2EXT, and I²S3 with I²S3EXT to achieve full-duplex mode.

Note: I²S2EXT and I²S3EXT are only used for I²S full-duplex mode.

Figure 13-20 I²S full-duplex structure



I²Sx can be used as master, where x should be 2 or 3

- In half-duplex mode, only I²Sx can output SCK and WS
- In full-duplex mode, only I²Sx can output SCK and WS

I²SxEXT is only used for full-duplex mode, and I²SxEXT only for slave mode

Both the I²Sx and I²SxEXT can be configured as transmit or receive mode.

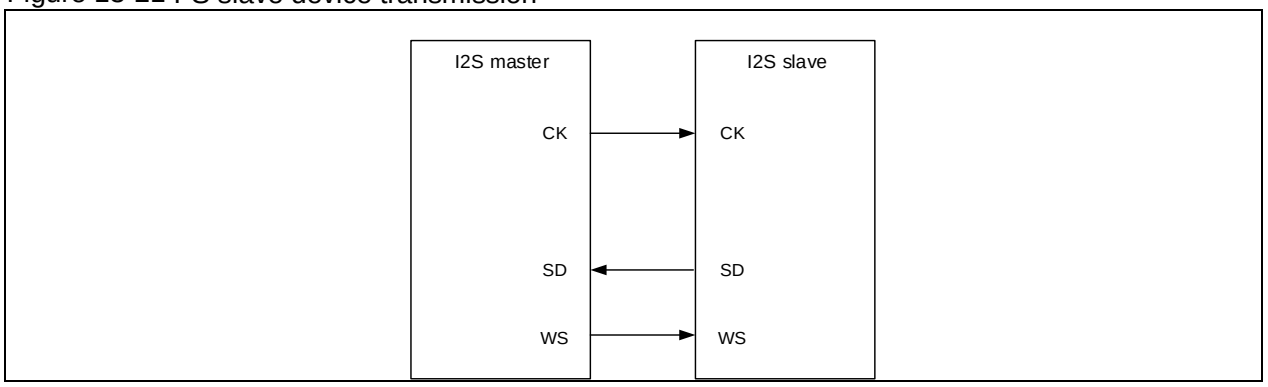
13.3.3 Operating mode selector

The SPI, used as I²S selector, offers multiple operating modes for selection, namely, slave device transmission, slave device reception, master device transmission and master device reception. This is done by software configuration.

Slave device transmission:

Set the I2SMSEL bit, and OPERSEL[1:0] =00, the I²S will work in slave device transmission mode.

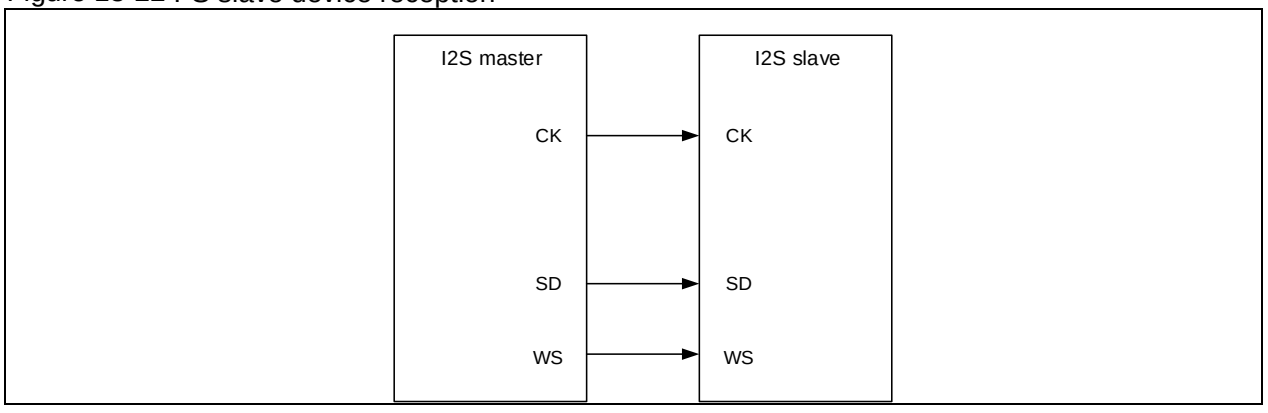
Figure 13-21 I²S slave device transmission



Slave device reception:

Set the I2SMSEL bit, and OPERSEL[1:0]=01, the I²S will work in slave device reception mode.

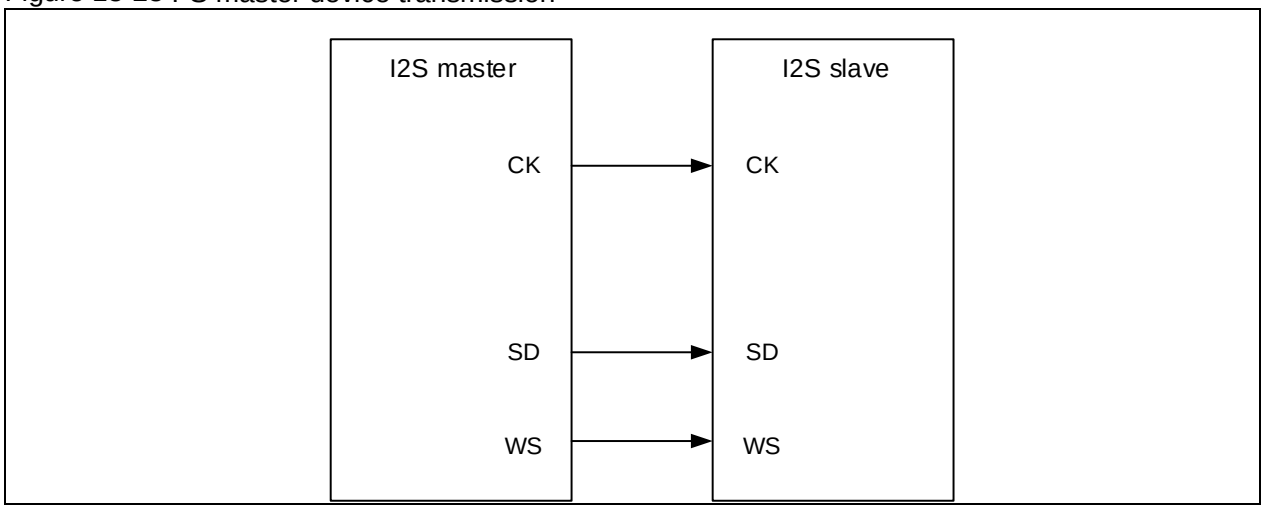
Figure 13-22 I²S slave device reception



Master device transmission:

Set the I2SMSEL bit, and OPERSEL[1:0]=10, the I²S will work in master device transmission mode.

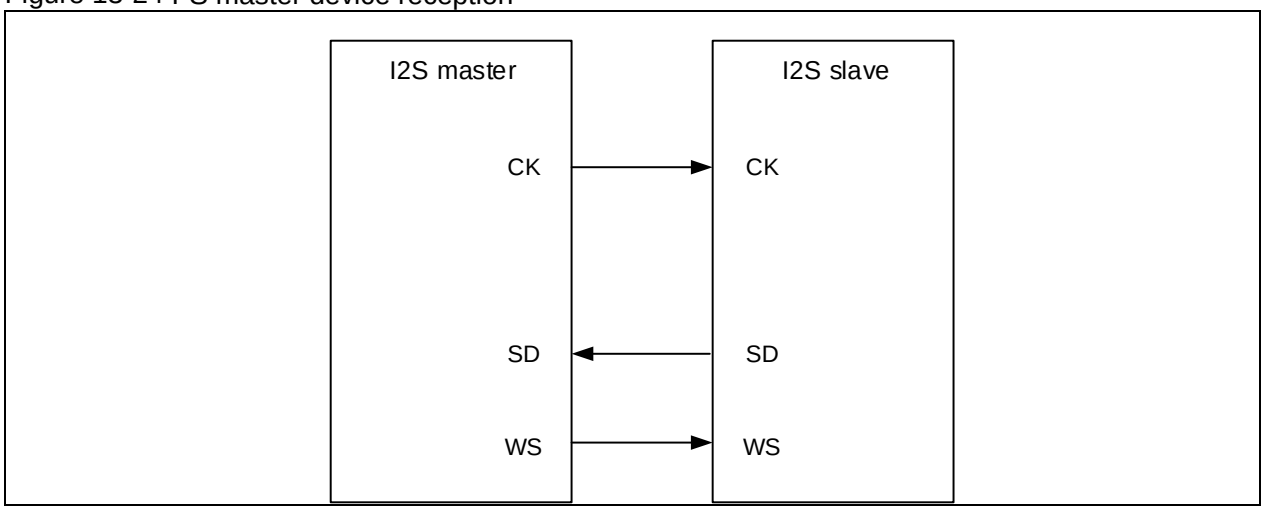
Figure 13-23 I²S master device transmission



Master device reception:

Set the I2SMSEL bit, and OPERSEL[1: 0]=11, the I²S will work in master device reception mode.

Figure 13-24 I²S master device reception



13.3.4 Audio protocol selector

While being used as I²S, the SPI supports multiple audio protocols. The user can control the audio protocol selector through software configuration to select the desired audio protocol, with the data bits and channel bits being controlled by the audio protocol selector. Besides, the user can also select the data bits and channel bits through software configuration. Meanwhile, the audio protocol selector manages the WS controller, output or detect the WS signal that meets the protocol requirements.

- Select audio protocol by setting the STDSEL bit

STDSLE=00: Philips standard
STDSLE=01: MSB-aligned standard (left-aligned)
STDSLE=10: LSB-aligned standard (right-aligned)
STDSLE=11: PCM standard

- Select PCM frame synchronization format: PCMFSSEL=1 for PCM long frame synchronization, PCMFSSEL=0 for short frame synchronization (this step is required when selecting PCM protocol)
- Select data bits by setting the I2SDBN bit

I2SDBN=00: 16 bits
I2SDBN =01: 24 bits
I2SDBN =10: 32 bits

- Select channel bits by setting the I2SCBN bit

I2SCBN =0: 16 bits
I2SCBN =1: 32 bits

Note: Read/Write operation mode depends on the selected audio protocols, data bits and channel bits. The following lists all possible configuration combinations and their respective read and write operation mode.

- Philips standard, PCM standard, MSB-aligned or LSB-aligned standard, 16-bit data and 16-bit channel

The data bit is the same as the channel bit. Each channel requires one read/write operation from/ to the SPI_DT register, and the number of DMA transfer is 1.

- Philips standard, PCM standard or MSB-aligned standard, 16-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires one read/write operation from/to the SPI_DT register, and the number of DMA transfer is 1. The first 16 bits (MSB) are the significant bits, and the 16-bit LSB is forced to 0 by hardware.

- Philips standard, PCM standard or MSB-aligned standard, 24-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires two read/write operations from/to the SPI_DT register, and the number of DMA transfer is 2. The 16-bit MSB transmits and receives the first 16-bit data, the 16-bit LSB transmits and receives the 8-bit MSB data, with 8-bit LSB data being forced to 0 by hardware.

- Philips standard, PCM standard, MSB-aligned or LSB-aligned standard, 32-bit data and 32-bit channel

The data bit is the same as the channel bit. Each channel requires two read/write operations from/to the SPI_DT register, and the number of DMA transfer is 2. These 32-bit data are proceeded in two times, with 16-bit data each time.

- LSB-aligned standard, 16-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires one read/write operation from/to the SPI_DT register, and the number of DMA transfer is 1. The 16 bits (LSB) are the significant bits while the first 16-bit data (MSB) are forced to 0 by hardware.

- LSB-aligned standard, 24-bit data and 32-bit channel

The data bit is different from the channel bit. Each channel requires two read/write operations from/to the SPI_DT register, and the number of DMA transfer is 2. For the first 16-bit data, its 8-bit LSB are the significant bits, with the 8-bit MSB forced to 0 by hardware; the subsequent 16 bits transmit and receive the second 16-bit data.

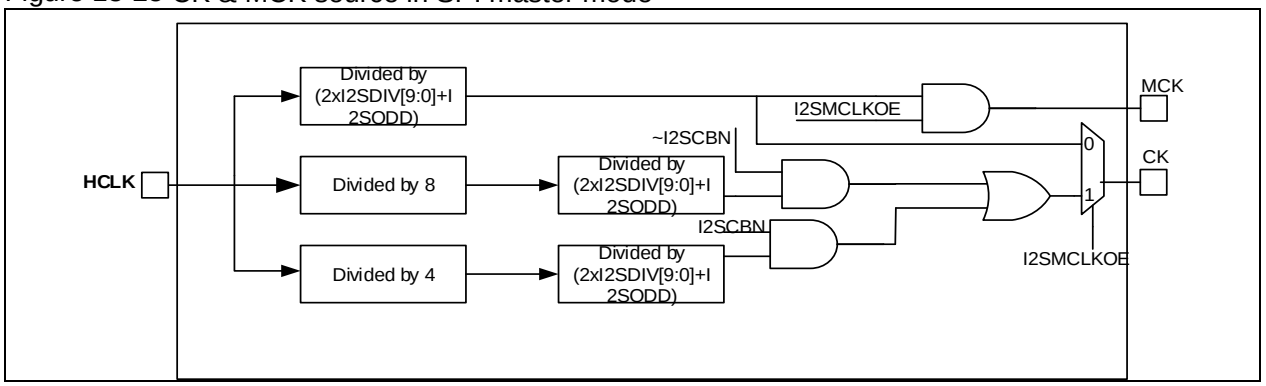
13.3.5 I2S_CLK controller

The audio protocols the SPI supports adopts synchronous transmission. In master mode, it is required to generate a communication clock for data reception and transmission on the SPI, and the communication clock should be output to the slave via IO for data reception and transmission. In slave mode, the communication clock is provided by master, and is input to the SPI via IO. In all, the I2S_SCK controller is used for the generation and distribution of I2S_SCK, with the configuration procedure detailed as follows:

When used as I2S master, the SPI can provide communication clock (CK) and main peripheral clock (MCK) shown in Figure 13-25. The CK and MCK are generated by HCLK divider, with the prescaler of the MCK determined by I2SDIV and I2SODD. The calculation formula is seen in Figure 13-25.

The prescaler of the CK depends on whether to provide the main clock for peripherals. To ensure that the main clock is always 256 times larger than the audio sampling frequency, the channel bits should be taken into account. When the main clock is needed, the CK should be divided by 8 (I2SCBN=0) or 4 (I2SCBN=1), then divided again by the same prescaler as that of the MCK, that is the final communication clock; When the main clock is not needed, the prescaler of the CK is determined by I2SDIV and I2SODD, shown in Figure 13-25.

Figure 13-25 CK & MCK source in SPI master mode



Apart from the above-mentioned configuration, the following table lists the values of I2SDIV and I2SODD corresponding to some specific frequencies, as well as their respective error for the users to configure the I2SDIV and I2SODD.

Table 13-1 Audio frequency precision using system clock

SCLK (MHz)	MCK	Target Fs (Hz)	16bit				32bit			
			I2S DIV	I2S_ODD	RealFs	Error	I2S DIV	I2S_ODD	RealFs	Error
192	No	192000	15	1	193548.4	0.80%	8	0	187500	2.34%
192	No	96000	31	1	95238.1	0.79%	15	1	96774.19	0.80%
192	No	48000	62	1	48000	0%	31	1	47619.05	0.79%
192	No	44100	68	0	44117.65	0.04%	34	0	44117.65	0.04%
192	No	32000	94	0	31914.89	0.26%	47	0	31914.89	0.26%
192	No	22050	136	0	22058.82	0.04%	68	0	22058.82	0.04%
192	No	16000	187	1	16000	0%	94	0	15957.45	0.26%
192	No	11025	272	0	11029.41	0.04%	136	0	11029.42	0.04%
192	No	8000	375	0	8000	0%	187	1	8000	0%
192	Yes	96000	4	0	93750	2.34%	4	0	93750	2.34%
192	Yes	48000	8	0	46875	2.34%	8	0	46875	2.34%
192	Yes	44100	8	1	44117.65	0.04%	8	1	44117.65	0.04%
192	Yes	32000	11	1	32608.7	1.90%	11	1	32608.7	1.90%
192	Yes	22050	17	0	22058.82	0.04%	17	0	22058.82	0.04%
192	Yes	16000	23	1	15957.45	0.26%	23	1	15957.45	0.26%
192	Yes	11025	34	0	11029.41	0.04%	32	0	11029.41	0.04%
192	Yes	8000	47	0	7978.72	0.26%	47	0	7978.72	0.26%

13.3.6 DMA transfers

The SPI supports write and read operations with DMA. Whether used as SPI or I²S, read/write request using DMA comes from the same peripheral. As a result, their configuration procedure are the same, described as follows.

Transmission with DMA

- Select DMA channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the SPI_DT register address as the destination address bit of DMA transfer in the DMA control register. Data will be sent to this address after transmit request is received by DMA.
- Configure the source of DMA transfer: Configure the memory address as the source of DMA transfer in the DMA control register. Data will be loaded into the SPI_DT register from the memory address after transmit request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.

- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register.
- Enable DMA transfer channel in the DMA control register.

Reception with DMA

- Select DMA transfer channel: Select a DMA channel for the current SPI from DMA channel map table described in DMA chapter.
- Configure the destination of DMA transfer: Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be loaded from the SPI_DT register to the programmed destination after reception request is received by DMA.
- Configure the source of DMA transfer: Configure the SPI_DT register address as the source of DMA transfer in the DMA control register. Data will be loaded from the SPI_DT register to the programmed destination after reception request is received by DMA.
- Configure the total number of bytes to be transferred in the DMA control register.
- Configure the channel priority of DMA transfer in the DMA control register.
- Configure DMA interrupt generation after half or full transfer in the DMA control register
- Enable DMA transfer channel in the DMA control register.

13.3.7 Transmitter/Receiver

Whether being used as SPI or I2S, there is no difference for CPU. The SPI (in whatever mode) shares the same base address, the same SPI_DT register, the same transmitter and receiver. The SPI transmitter and receiver is responsible for sending and receiving the desired data frame according to the configuration of the communication controller. Thus their status flags such as TDBE, RDBF and ROERR, and their interrupt enable bits including TDBEIE, RDBFIE and ERRIE are identical.

Special attention must be paid to:

- CRC check is not available on the I2S. Any operation linked to CRC, including CCERR flag and the corresponding interrupts, is not supported.
- I2S protocol needs decode the current channel status. The ACS bit is used to judge whether the current transfer occurs on the left channel (ACS=0) or the right channel (ACS=1).
- TUERR bit indicates whether an underrun occurs. TUERR=1 means an underrun error occurs on the transmitter. An interrupt is generated when the ERRIE is set.
- Read/write operation to the SPI_DT register is different under different audio protocols, data bits and channel bits. Refer to the audio protocol selector section for more information.
- Pay more attention to the I^2S disable operation under different configurations, shown as follows:

— I2SDBN=00, I2SCBN=1, STDSLE=10: wait for the second-to-last RDBF=1 and 17 CK periods before disabling the I^2S.

— I2SDBN=00, I2SCBN=1, STDSLE=00 or STDSLE=01 or STDSLE=11: wait for the last RDBF=1 and one CK period before the I^2S.

— I2SDBN, I2SCBN,STDSLE combination: wait for the second-to-last RDBF=1 and one CK period before disabling the I^2S.

I^2S transmitter configuration procedure:

- Configure operation mode selector
- Configure audio protocol selector
- Configure I2S_CLKK controller
- Configure DMA transfer (if necessary)
- Set the I2SEN bit to enable I^2S

I^2S receiver configuration procedure:

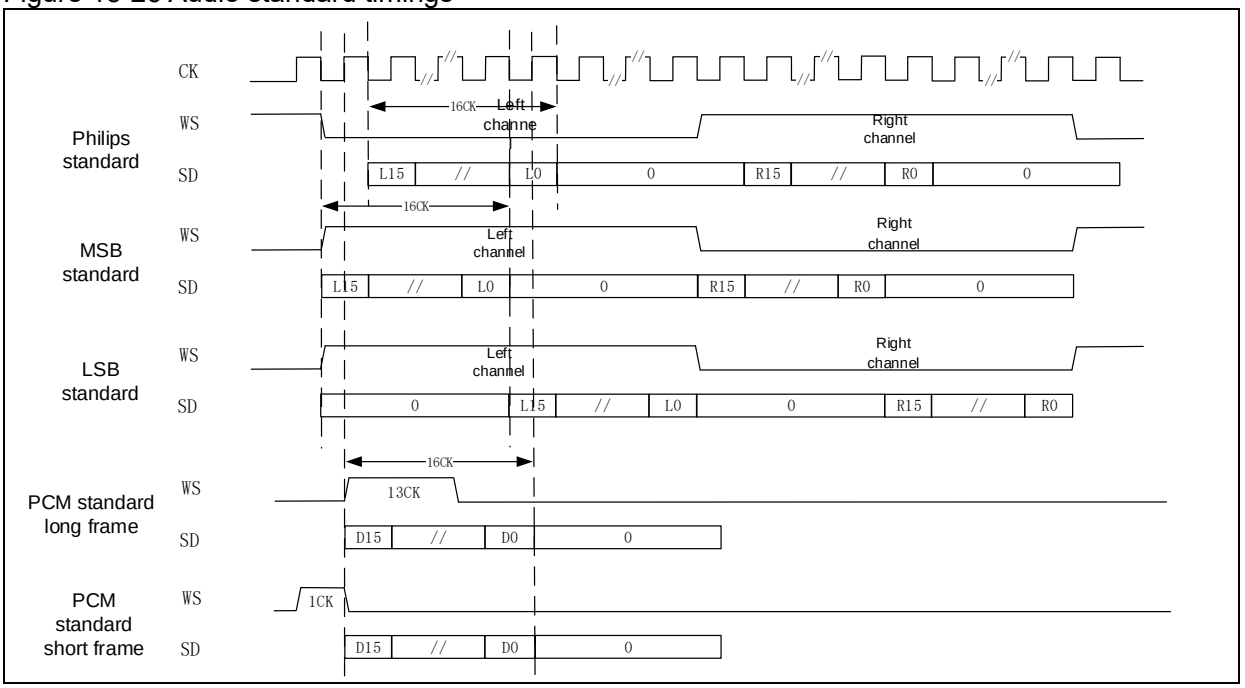
- Configure operation mode selector
- Configure audio protocol selector
- Configure I2S_CLK controller
- Configure DMA transfer (if necessary)

- Set the I2SEN bit to enable I²S

13.3.8 I²S communication timings

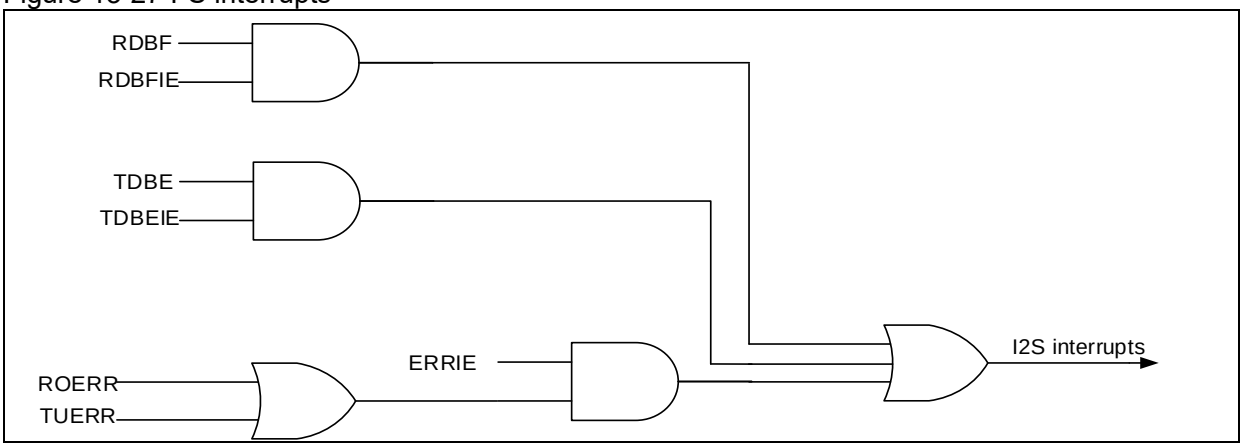
I2S can address four different audio standards: Philips standard, the most significant byte (left-aligned) and the least significant byte (right-aligned) standards, and the PCM standard. Figure 13-26 shows their respective timings.

Figure 13-26 Audio standard timings



13.3.9 Interrupts

Figure 13-27 I²S interrupts



13.3.10 IO pin control

The I²S needs three pins for transfer operation, namely, the SD, WS and CK. The MCK pin is also required if need to provide main clock for peripherals. The I²S shares some pins with the SPI, described as follows:

- SD: Serial data (mapped on the MOSI pin) for bidirectional data transmission and reception.
- WS: Word select (mapped on the CS pin) for data control signal output in master mode, and input in slave mode.
- CK: Communication clock (mapped on the SCK pin) as clock signal output in master mode, and input in slave mode.
- MCK: Master clock (mapped independently) is used to provide main clock for peripherals. The frequency of output clock signal is set to 256x Fs (audio sampling frequency).

13.4 SPI registers

These peripheral registers must be accessed by words (32 bits).

Table 13-2 SPI register map and reset value

Register	Offset	Reset value
SPI_CTRL1	0x00	0x0000
SPI_CTRL2	0x04	0x0000
SPI_STS	0x08	0x0002
SPI_DT	0x0C	0x0000
SPI_CPOLY	0x10	0x0007
SPI_RCRC	0x14	0x0000
SPI_TCRC	0x18	0x0000
SPI_I2SCTRL	0x1C	0x0000
SPI_I2SCLK	0x20	0x0002

13.4.1 SPI control register 1 (SPI_CTRL1) (Not used in I²S mode)

Bit	Name	Reset value	Type	Description
Bit 15	SLBEN	0x0	rw	Single line bidirectional half-duplex enable 0: Disabled 1: Enabled
Bit 14	SLBTD	0x0	rw	Single line bidirectional half-duplex transmission direction This bit and the SLBEN bit together determine the data output direction in “Single line bidirectional half-duplex” mode. 0: Receive-only mode 1: Transmit-only mode
Bit 13	CCEN	0x0	rw	CRC calculation enable 0: Disabled 1: Enabled
Bit 12	NTC	0x0	rw	Transmit CRC next When this bit is set, it indicates that the next data transferred is CRC value. 0: Next transmitted data is the normal value 1: Next transmitted data is CRC value
Bit 11	FBN	0x0	rw	Frame bit num This bit is used to configure the number of data frame bit for transmission/reception. 0: 8-bit data frame 1: 16-bit data frame
Bit 10	ORA	0x0	rw	Receive-only active In two-wire unidirectional mode, when this bit is set, it indicates that Receive-only is active, but the transmit is not allowed. 0: Transmission and reception 1: Receive-only mode
Bit 9	SWCSEN	0x0	rw	Software CS enable When this bit is set, the CS pin level is determined by the SWCSIL bit. The status of I/O level on the CK pin is invalid. 0: Disabled 1: Enabled
Bit 8	SWCSIL	0x0	rw	Software CS internal level This bit is valid only when the SWCSEN is set. It determines the level on the CS pin. In master mode, this bit must be set. 0: Low level 1: High level

Bit 7	LTF	0x0	rw	LSB transmit first This bit is used to select for MST transfer first or LSB transfer first. 0: MSB 1: LSB
Bit 6	SPIEN	0x0	rw	SPI enable 0: Disabled 1: Enabled
Bit 5: 3	MDIV	0x0	rw	Master clock frequency division In master mode, the peripheral clock divided by the prescaler is used as SPI clock. The MDIV[3] bit is in the SPI_CTRL2 register, MDIV[3: 0]: 0000: Divided by 2 0001: Divided by 4 0010: Divided by 8 0011: Divided by 16 0100: Divided by 32 0101: Divided by 64 0110: Divided by 128 0111: Divided by 256 1000: Divided by 512 1001: Divided by 1024
Bit 2	MSTEN	0x0	rw	Master enable 0: Disabled (Slave) 1: Enabled (Master)
Bit 1	CLKPOL	0x0	rw	Clock polarity Indicates the polarity of clock output in idle state. 0: Low level 1: High level
Bit 0	CLKPHA	0x0	rw	Clock phase 0: Data capture starts from the first clock edge 1: Data capture starts from the second clock edge

Note: The SPI_CTRL1 register must be 0 in I^2S mode.

13.4.2 SPI control register 2 (SPI_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 15: 10	Reserved	0x00	resd	Forced 0 by hardware.
Bit 9	MDIV3EN	0x0	rw	Master clock frequency divided by 3 enable 0: Disabled 1: Enabled Note: When this bit is set, the MDIV[3: 0] becomes invalid, and the SPI clock is forced to be PCLK/3. When SPI/3 is used and PCLK/SYSCLK≠1, SPI clock doesn't have 50% duty cycle. For details, reer to the data sheet.
Bit 8	MDIV[3]	0x0	rw	Master clock frequency division Refer to the MDIV[2: 0] of the SPI_CTRL1 register.
Bit 7	TDBEIE	0x0	rw	Transmit data buffer empty interrupt enable 0: Disabled 1: Enabled
Bit 6	RDBFIE	0x0	rw	Receive data buffer full interrupt enable 0: Disabled 1: Enabled
Bit 5	ERRIE	0x0	rw	Error interrupt enable This bit controls interrupt generation when errors occur (CCERR, MMERR, ROERR, TUERR and CSPAS) 0: Disabled 1: Enabled
Bit 4	TIEN	0x0	rw	TI mode enable

				0: TI mode disabled (Motorola mode) 1: TI mode enabled (TI mode) Note: This mode is not used in I2S mode. It must be 0 in I2S mode.
Bit 3	Reserved	0x0	resd	Kept at its default value
Bit 2	HWCSOE	0x0	rw	Hardware CS output enable This bit is valid only in master mode. When this bit is set, the I/O output on the CS pin is low; when this bit is 0, the I/O input on the CS pin must be set high. 0: Disabled 1: Enabled
Bit 1	DMATEN	0x0	rw	DMA transmit enable 0: Disabled 1: Enabled
Bit 0	DMAREN	0x0	rw	DMA receive enable 0: Disabled 1: Enabled

13.4.3 SPI status register (SPI_STS)

Bit	Name	Reset value	Type	Description
Bit 15: 9	Reserved	0x00	resd	Forced 0 by hardware
Bit 8	CSPAS	0x0	ro	CS pulse abnormal setting flag 0: CS pulse flag normal 1: CS pulse flag is set abnormally Note: This bit is used for TI slave mode. It is cleared by reading the STS register.
Bit 7	BF	0x0	ro	Busy flag 0: SPI is not busy. 1: SPI is busy.
Bit 6	ROERR	0x0	ro	Receiver overflow error 0: No overflow error 1: Overflow error occurs.
Bit 5	MMERR	0x0	ro	Master mode error This bit is set by hardware and cleared by software (read/write access to the SPI_STS register, followed by write operation to the SPI_CTRL1 register) 0: No mode error 1: Mode error occurs.
Bit 4	CCERR	0x0	rw0c	CRC error Set by hardware, and cleared by software. 0: No CRC error 1: CRC error occurs.
Bit 3	TUERR	0x0	ro	Transmitter underload error Set by hardware, and cleared by software (read the SPI_STS register). 0: No underload error 1: Underload error occurs. Note: This bit is only used in I^2S mode.
Bit 2	ACS	0x0	ro	Audio channel state This bit indicates the status of the current audio channel. 0: Left channel 1: Right channel Note: This bit is only used in I^2S mode.
Bit 1	TDBE	0x1	ro	Transmit data buffer empty

				0: Transmit data buffer is not empty. 1: Transmit data buffer is empty.
Bit 0	RDBF	0x0	ro	Receive data buffer full 0: Transmit data buffer is not full. 1: Transmit data buffer is full.

13.4.4 SPI data register (SPI_DT)

Bit	Name	Reset value	Type	Description
Bit 15: 0	DT	0x0000	rw	Data value This register controls read and write operations. When the data bit is set as 8 bit, only the 8-bit LSB [7: 0] is valid.

13.4.5 SPICRC register (SPI_CPOLY) (Not used in I²S mode)

Bit	Name	Reset value	Type	Description
Bit 15: 0	CPOLY	0x0007	rw	CRC polynomial This register contains the polynomial used for CRC calculation. Note: This register is valid only in SPI mode.

13.4.6 SPIRxCRC register (SPI_RCRC) (Not used in I²S mode)

Bit	Name	Reset value	Type	Description
Bit 15: 0	RCRC	0x0000	ro	Receive CRC When CRC calculation is enabled, this register contains the CRC value computed based on the received data. This register is reset when the CCEN bit in the SPI_CTRL1 register is cleared. When the data frame format is set to 8-bit data, only the 8-bit LSB ([7: 0]) are calculated based on CRC8 standard; when 16-bit data bit is selected, follow CRC16 standard. Note: This register is only used in SPI mode.

13.4.7 SPITxCRC register (SPI_TCRC)

Bit	Name	Reset value	Type	Description
Bit 15: 0	TCRC	0x0000	ro	Transmit CRC When CRC calculation is enabled, this register contains the CRC value computed based on the transmitted data. This register is reset when the CCEN bit in the SPI_CTRL1 register is cleared. When the data frame format is set to 8-bit data, only the 8-bit LSB ([7: 0]) are calculated based on CRC8 standard; when 16-bit data bit is selected, follow CRC16 standard. Note: This register is only used in SPI mode.

13.4.8 SPI_I2S register (SPI_I2SCTRL)

Bit	Name	Reset value	Type	Description
Bit 15: 12	Reserved	0x0	resd	Forced 0 by hardware.
Bit 11	I2SMSEL	0x0	rw	I²S mode select 0: SPI mode 1: I²S mode
Bit 10	I2SEN	0x0	rw	I²S enable 0: Disabled 1: Enabled
Bit 9: 8	OPERSEL	0x0	rw	I²S operation mode select 00: Slave transmission 01: Slave reception 10: Master transmission

				11: Master reception
Bit 7	PCMFSSEL	0x0	rw	PCM frame synchronization This bit is valid only when the PCM standard is used. 0: Short frame synchronization 1: Long frame synchronization
Bit 6	Reserved	0x0	resd	Kept at its default value
Bit 5: 4	STDSEL	0x0	rw	I^2S standard select 00: Philips standard 01: MSB-aligned standard (left-aligned) 10: LSB-aligned standard (right-aligned) 11: PCM standard
Bit 3	I2SCLKPOL	0x0	rw	I^2S clock polarity This bit indicates the clock polarity on the clock pin in idle state. 0: Low 1: High
Bit 2: 1	I2SDBN	0x0	rw	I^2S data bit num 00: 16-bit data length 01: 24-bit data length 10: 32-bit data length 11: Not allowed.
Bit 0	I2SCBN	0x0	rw	I^2S channel bit num This bit can be configured only when the I^2S is set to 16-bit data; otherwise, it is fixed to 32-bit by hardware. 0: 16-bit wide 1: 32-bit wide

13.4.9 SPI_I2S prescaler register (SPI_I2SCLKP)

Bit	Name	Reset value	Type	Description
Bit 15: 12	Reserved	0x0	resd	Forced 0 by hardware.
Bit 9	I2SMCLKOE	0x0	rw	I^2S Master clock output enable 0: Disabled 1: Enabled
Bit 8	I2SODD	0x0	rw	Odd factor for I^2S division 0: Actual divider factor =I2SDIV*2 1: Actual divider factor =(I2SDIV*2)+1
Bit 11: 10 Bit 7: 0	I2SDIV	0x02	rw	I^2S division It is not allowed to configure I2SDIV[9: 0]=0 or I2SDIV[9: 0]=1

14 Full-duplexed I²S interface (I2SF)

14.1 I2SF interface introduction

I2SF audio interface is an extended version of I²S which supports full duplex mode. Its master clock sources can be from system clock, PLL input clock, HICK output clock or external input clock. A more accurate audio frequency can be achieved by configuring the master clock of I²SF.

14.2 I2SF functional overview

In addition to I²S functionalities described in Chapter 13, there are additional features available to I²SF which will be detailed in the following sections.

14.2.1 I2SF full duplex mode

I²SF full-duplex mode is enabled by setting the I2SFDUPEN bit in the I2SF_I2SCTRL register. The I²SF can operate in master or slave mode through setting the OPERSEL[1] register. When the I²SF is configured in master transmit or slave transmit mode through the OPERSEL[1:0], SD represents input while SDEXT represent input; when the I²SF operates in master-receive/slave receive mode, SD is used as input, and the SDEXT as output.

Figure 14-1 I²SF full-duplex host transmit/slave transmit

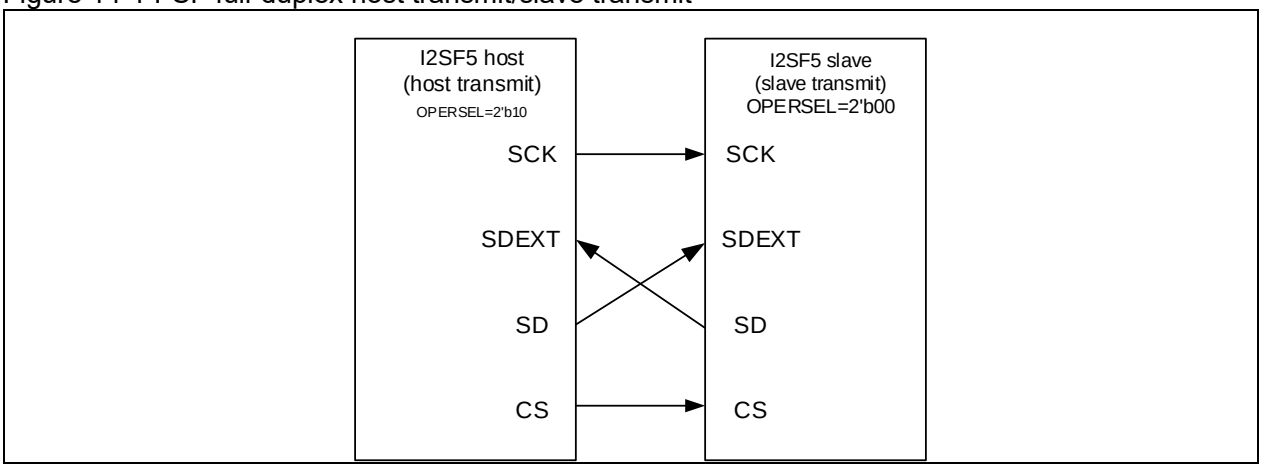


Figure 14-2 I²SF full-duplex host transmit/slave receive

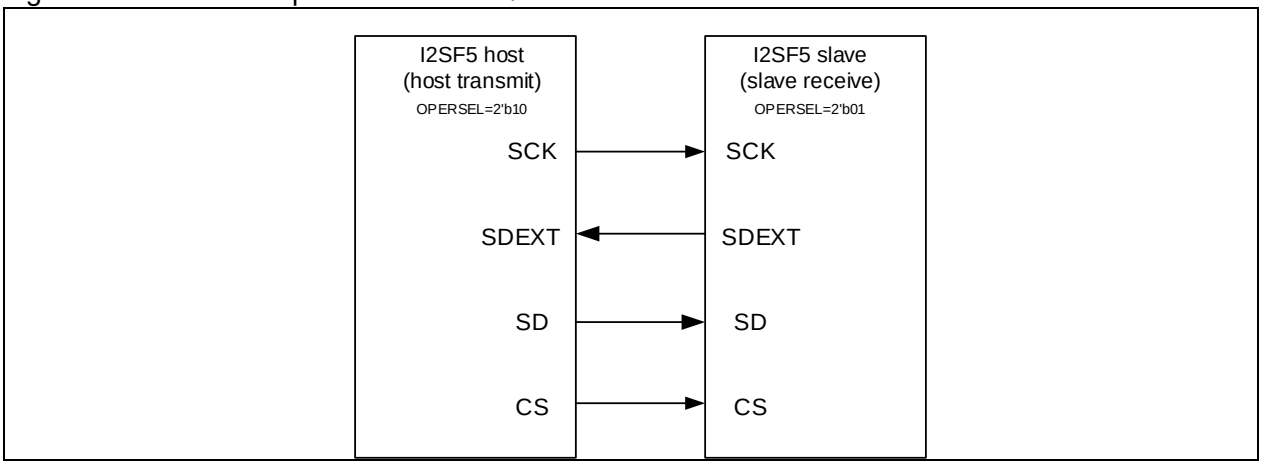


Figure 14-3 I²SF full-duplex host receive/slave transmit

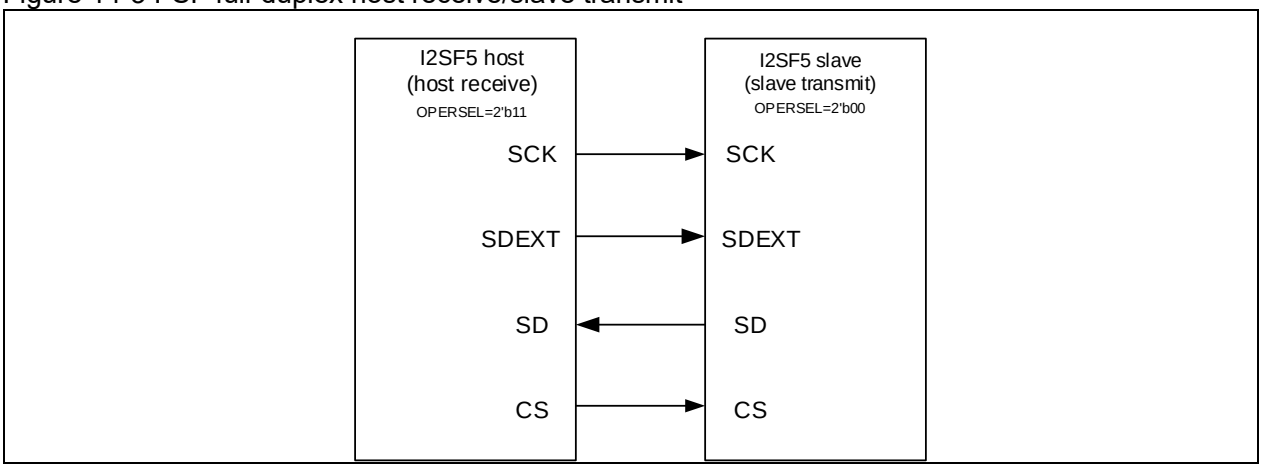
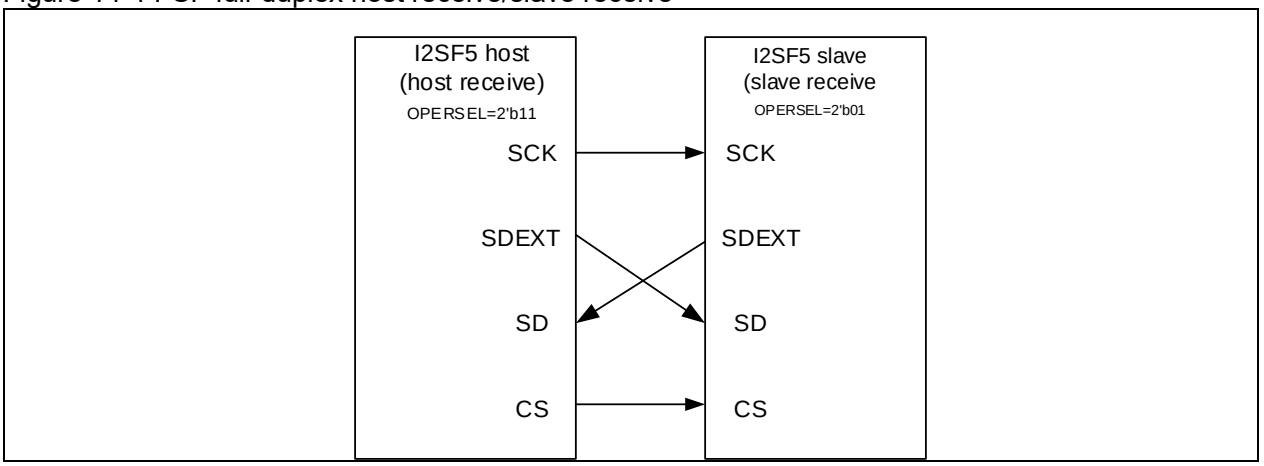


Figure 14-4 I²SF full-duplex host receive/slave receive



Full-duplex transfer in master mode (I2SFDUPEN=1 and OPERSEL[9]=1)

- The transfer sequence begins when a data is written to the I2SF_DT register (transmission buffer).
- The data is parallel loaded into the shift register during the first bit transmission, and then shifted out, serially to the SD/SDEXT pin.
- The data received on SDEXT/SD are serially moved to the shift register, and then serially loaded into the I2SF_DT register.

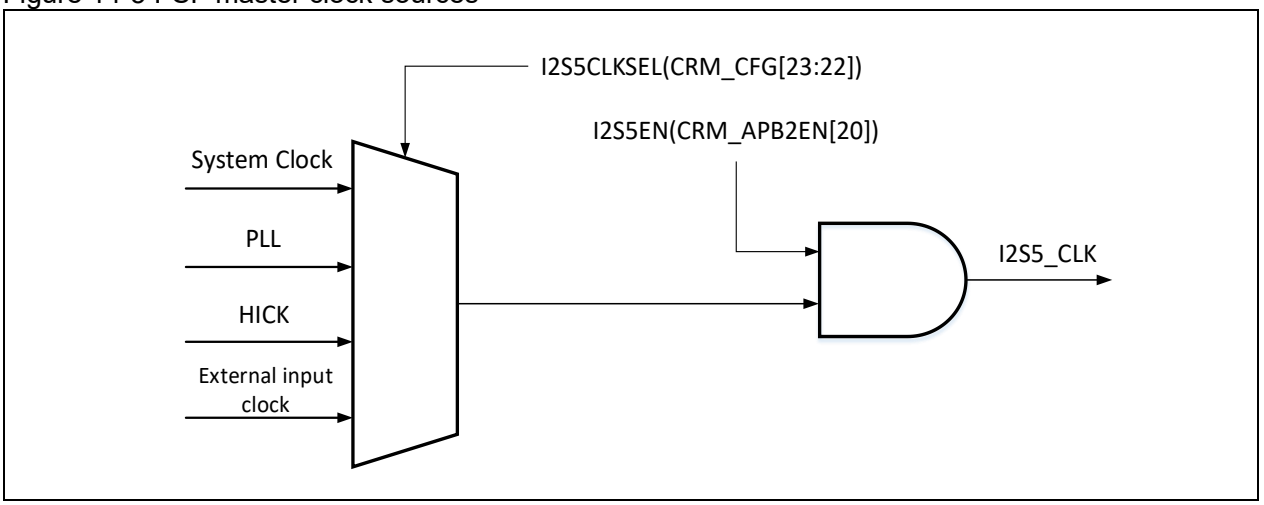
Full-duplex transfer in slave mode (I2SFDUPEN=1 and OPERSEL[9]=0)

- The data transfer begins when the slave receives a clock signal. Then the data on the SDEXT/SD are serially loaded into the shift register. The data in the shift register is sent to the I2SF_DT register each time a 16-bit data transfer is finished.
- At the same time, the data of the I2SF_DT register (transmission buffer) are in parallel loaded into the shift register, and then sent to the SD/SDEXT pins serially. It shout be noted that the to-be-sent data have been written to the I2SF_DT register before the I²SF host starts transferring data.

14.2.2 I2SF master clock sources

There are four clock sources dedicated to I²SF controller, which is configured through the I2SF5CLKSEL[1: 0] register. The external input clock is input via IO. The HICK oscillator clock output can be configured as 48MHz. The PLL is derived from PLLP.

Figure 14-5 I²SF master clock sources



14.2.3 PCM mode

The I²SF has an additional I2SFPCMCKSEL register used to select rising edge or falling edge of the clock for sampling data in PCM long-frame or PCM short-frame format. See the figures below for communication timings.

Figure 14-6 Data sampling on falling edge in PCM mode

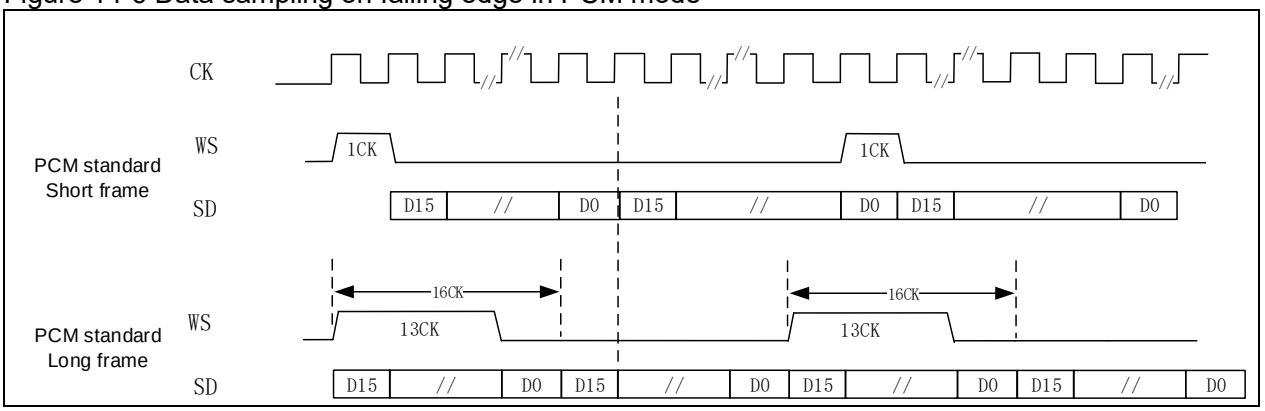
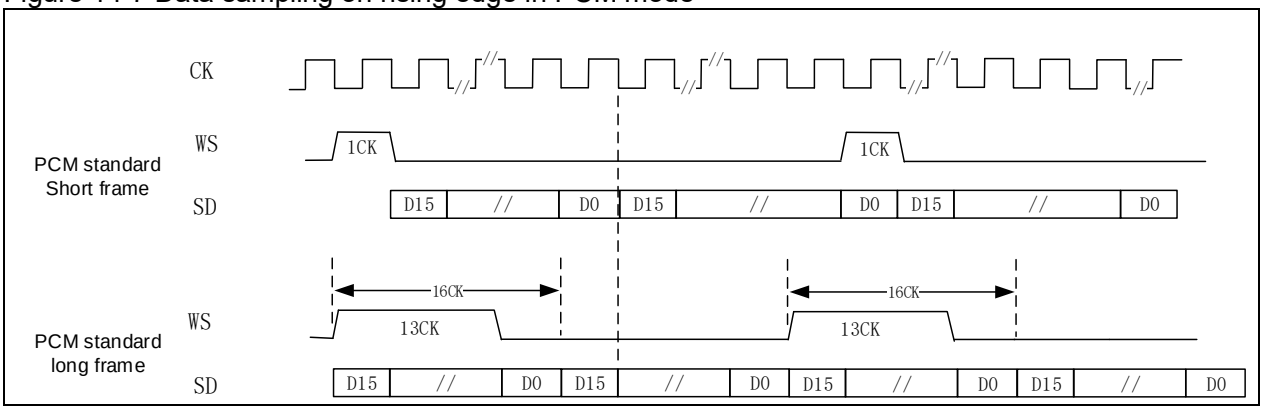
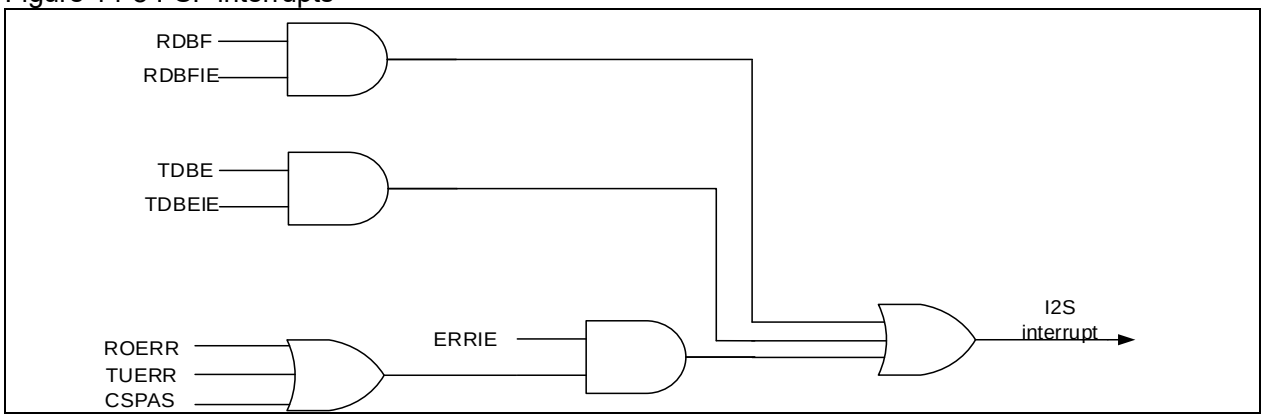


Figure 14-7 Data sampling on rising edge in PCM mode



14.2.4 Interrupts

Figure 14-8 I²SF interrupts



14.2.5 IO pin control

When full-duplex feature is enabled, the I2SF needs three pins to implement I²S communication. They are SD (data pin), WS (synchronization pin) and CK (communication clock pin). If there is a need to provide main clock for peripherals, a MCLK is required. When the full-duplex mode is enabled, it requires another pin SDEXT to implement bidirectional data transfer. In master mode, it is possible to select CLKIN_IN as the external input interface of I²S master clock through the I2SF5CLKSEL[1:0] register.

- SD: Serial data (mapped on the MOSI pin). It serves as a bidirectional transceiver pin when I²SF full-duplex mode is disabled. The direction of data transfer is dependent on registers when the I2SF full-duplex mode is enabled.
- SDEXT: this pin is not available when I2SF full-duplex mode is disabled. The direction of data transfer is dependent on registers when I2SF full-duplex mode is enabled.

  When the OPERSEL[9:8] is configured as host transmit/slave transmit mode, SD is used as output and the SDEXT as input. When the OPERSEL[9:8] is configured as host receive/slave receive mode, SD is used as input, and SDEXT as output.
- WS: Word select (mapped on the CS pin) for data control signal output in master mode, and input in slave mode.
- CK: Communication clock (mapped on the SCK pin) as clock signal output in master mode, and input in slave mode.
- CLKIN_IN: It is the I²S master clock external input interface in master mode, and not available in slave mode.
- MCK: Master clock (mapped independently) is used to provide main clock for peripherals. The frequency of output clock signal is set to 256x Fs (audio sampling frequency).

14.2.6 Special notes on I2SF usage

- The clock of the APB2 should be greater than the one of I²SF.
- In slave mode, when an edge is detected at an unexpected location, the CS pulse error flag is set by hardware, and it is cleared by software reading the register.

14.3 I²SF registers

These peripheral registers must be accessed by words (32 bits).

Table 14-1 I²SF5 register map and reset value

Register	Offset	Reset value
I2SF_CTRL2	0x04	0x0000
I2SF_STS	0x08	0x0002
I2SF_DT	0x0C	0x0000
I2SF_I2SCTRL	0x1C	0x0000
I2SF_I2SCLKP	0x20	0x0002
I2SF_MISC1	0x30	0x0000

14.3.1 I²SF control register 2 (I2SF_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 15: 8	Reserved	0x00	resd	Forced to 0 by hardware.
Bit 7	TDBEIE	0x0	rw	Transmit data buffer empty interrupt enable 0: Disabled 1: Enabled
Bit 6	RDBFIE	0x0	rw	Receive data buffer full interrupt enable 0: Disabled 1: Enabled
Bit 5	ERRIE	0x0	rw	Error interrupt enable This bit controls interrupt generation when errors occur (CCERR, MMERR, ROERR and TUERR) 0: Disabled 1: Enabled
Bit 4: 2	Reserved	0x0	resd	Kept at default value
Bit 1	DMATEN	0x0	rw	DMA transmit enable 0: Disabled 1: Enabled
Bit 0	DMAREN	0x0	rw	DMA receive enable 0: Disabled 1: Enabled

14.3.2 I²SF status register (I2SF_STS)

Bit	Name	Reset value	Type	Description
Bit 15: 9	Reserved	0x00	resd	Forced to 0 by hardware
Bit 8	CSPAS	0x0	ro	CS pulse abnormal setting flag 0: CS pulse flag normal 1: CS pulse flag is set abnormally Note: This bit is used for TI slave mode. It is cleared by reading the STS register.
Bit 7	BF	0x0	ro	Busy flag 0: SPI is not busy. 1: SPI is busy.
Bit 6	ROERR	0x0	ro	Receiver overflow error 0: No overflow error 1: Overflow error occurs.
Bit 5: 4	Reserved	0x0	resd	Forced to 0 by hardware
Bit 3	TUERR	0x0	ro	Transmitter underload error Set by hardware, and cleared by software (read the

				SPI_STS register). 0: No underload error 1: Underload error occurs. Note: This bit is only used in I²S mode.
Bit 2	ACS	0x0	ro	Audio channel state This bit indicates the status of the current audio channel. 0: Left channel 1: Right channel Note: This bit is only used in I²S mode.
Bit 1	TDBE	0x1	ro	Transmit data buffer empty 0: Transmit data buffer is not empty. 1: Transmit data buffer is empty.
Bit 0	RDBF	0x0	ro	Receive data buffer full 0: Transmit data buffer is not full. 1: Transmit data buffer is full.

14.3.3 I²SF data register (I2SF_DT)

Bit	Name	Reset value	Type	Description
Bit 15: 0	DT	0x0000	rw	Data value This register controls read and write operations. When the data bit is set as 8 bit, only the 8-bit LSB [7: 0] is valid.

14.3.4 I2SF configuration register (I2SF_I2SCTRL)

Bit	Name	Reset value	Type	Description
Bit 15: 12	Reserved	0x0	resd	Forced to 0 by hardware.
Bit 13	I2SFDUPEN	0x0	rw	I²S full duplex enable 0: I²S full duplex disabled 1: I²S full duplex enabled
Bit 12:11	Reserved	0x0	resd	Forced to 0 by hardware.
Bit 10	I2SEN	0x0	rw	I²S enable 0: Disabled 1: Enabled
Bit 9: 8	OPERSEL	0x0	rw	I²S operation mode select 00: Slave transmission 01: Slave reception 10: Master transmission 11: Master reception
Bit 7	PCMFSSEL	0x0	rw	PCM frame synchronization This bit is valid only when the PCM standard is used. 0: Short frame synchronization 1: Long frame synchronization
Bit 6	Reserved	0x0	resd	Kept at default value
Bit 5: 4	STDSEL	0x0	rw	I²S standard select 00: Philips standard 01: MSB-aligned standard (left-aligned) 10: LSB-aligned standard (right-aligned) 11: PCM standard
Bit 3	I2SCLKPOL	0x0	rw	I²S clock polarity This bit indicates the clock polarity on the clock pin in idle state. 0: Low 1: High
Bit 2: 1	I2SDBN	0x0	rw	I²S data bit num 00: 16-bit data length 01: 24-bit data length 10: 32-bit data length 11: Not allowed.

Bit 0	I2SCBN	0x0	rw	I²S channel bit num This bit can be configured only when the I²S is set to 16-bit data; otherwise, it is fixed to 32-bit by hardware. 0: 16-bit wide 1: 32-bit wide

14.3.5 I2SF prescaler register (I2SF_I2SCLKP)

Bit	Name	Reset value	Type	Description
Bit 15: 12	Reserved	0x0	resd	Forced 0 by hardware.
Bit 9	I2SMCLKOE	0x0	rw	I²S Master clock output enable 0: Disabled 1: Enabled
Bit 8	I2SODD	0x0	rw	Odd factor for I²S division 0: Actual divider factor =I2SDIV*2 1: Actual divider factor =(I2SDIV*2)+1
Bit 11: 10 Bit 7: 0	I2SDIV	0x02	rw	I²S division It is not allowed to configure I2SDIV[9: 0]=0 or I2SDIV[9: 0]=1

14.3.6 I2SF additional register (I2SF_MISC1)

Bit	Name	Reset value	Type	Description
Bit 15: 1	Reserved	0x0	resd	Forced 0 by hardware.
Bit 0	I2SFPCMCKSEL	0x02	rw	I²S PCM clock edge select 0: Falling edge 1: Rising edge

15 Timers

AT32F455 timers include basic timers, general-purpose timers, and advanced timers.

Please refer to Section 15.1 ~ Section 15.4 for the detailed function modes. All functions of different timers are shown in the following tables.

Table 15-1 TMR functional comparison

Timer type	Timer	Counter bit	Count mode	Repetition	Prescaler	DMA requests	Capture/compare channel	PWM input mode	EXT input	Brake input
Advanced-control timer	TMR1 TMR8	16	Up Down Up/Down	16-bit	1~65535	O	4	O	O	O
General-purpose timer	TMR2 TMR5	16/32	Up Down Up/Down	X	1~65535	O	4	O	TMR2	X
	TMR3 TMR4	16	Up Down Up/Down	X	1~65535	O	4	O	O	X
	TMR9 TMR12	16	Up Down Up/Down	8-bit	1~65535	O	2	O	X	O
	TMR10 TMR11 TMR13 TMR14	16	Up Down Up/Down	8-bit	1~65535	O	1	X	X	O
Basic timer	TMR6 TMR7	16	Up	X	1~65535	O	X	X	X	X

Timer type	Timer	Counter bit	Count mode	PWM output	Single pulse output	Complementary output	Dead-time	Encoder interface connection	Interfacing with hall sensors	Linkage peripheral
Advanced-control timer	TMR1 TMR8	16	Up Down Up/Down	O	O	O	O	O	O	Timer synchronization ADC/DAC
General-purpose timer	TMR2 TMR5	16/32	Up Down Up/Down	O	O	X	X	O	O	Timer synchronization ADC/DAC
	TMR3 TMR4	16	Up Down Up/Down	O	O	X	X	O	O	Timer synchronization ADC/DAC
	TMR9 TMR12	16	Up Down Up/Down	O	O	O	O	X	X	Timer synchronization
	TMR10 TMR11 TMR13 TMR14	16	Up Down Up/Down	O	O	O	O	X	X	Timer
Basic timer	TMR6 TMR7	16	Up	X	X	X	X	X	X	ADC/DAC

15.1 Basic timer (TMR6 and TMR7)

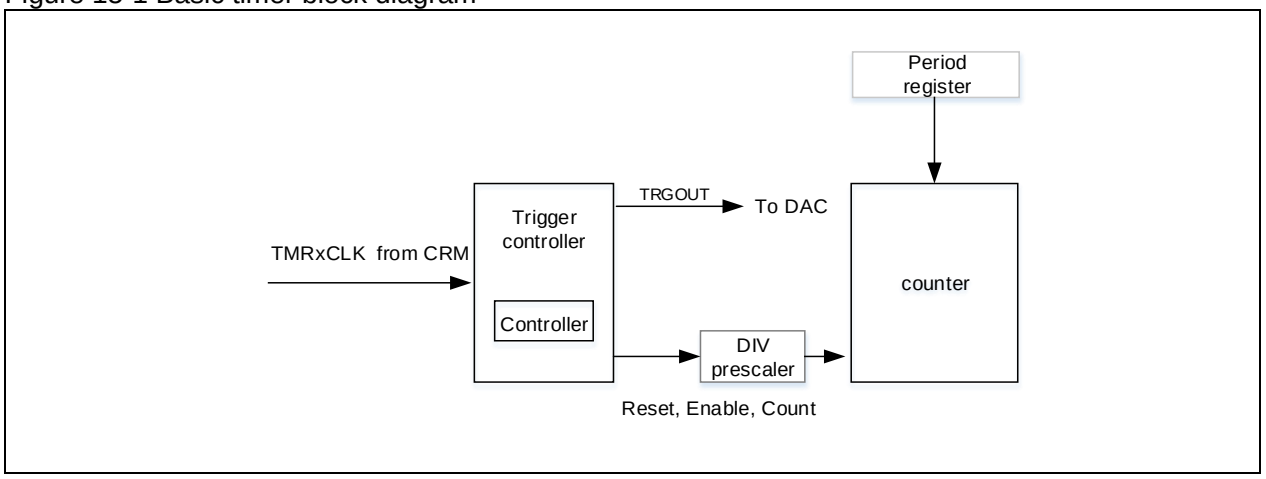
15.1.1 TMR6 and TMR7 introduction

Each of the basic timers (TMR6 and TMR7) includes a 16-bit upcounter and the corresponding control logic, without being connected to external I/Os, they can be used for basic timing and providing clocks for the digital-to-analog converter (DAC) and as an externl trigger for ADC.

15.1.2 TMR6 and TMR7 main features

- 16-bit upcounter, auto reload
- 16-bit prescaler used to divide TMR_CLK frequency by any factor between 1 and 65535
- Synchronization circuit to trigger DAC
- As external source to trigger ADC

Figure 15-1 Basic timer block diagram

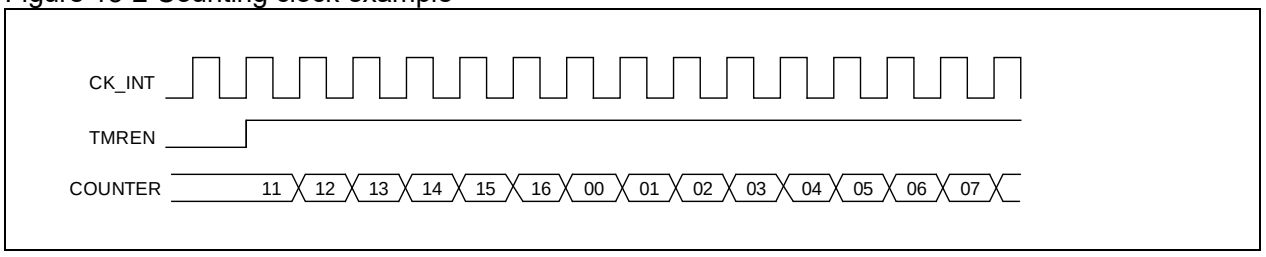


15.1.3 TMR6 and TMR7 functional overview

15.1.3.1 Counting clock

The counter clock of TMR6 and TMR7 is provided by the internal clock source (CK_INT) divided by prescaler. When TMR's APB clock prescaler factor is 1, the CK_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

Figure 15-2 Counting clock example



15.1.3.2 Counting mode

The basic timer only supports upcounting mode, and it has an internal 16-bit counter.

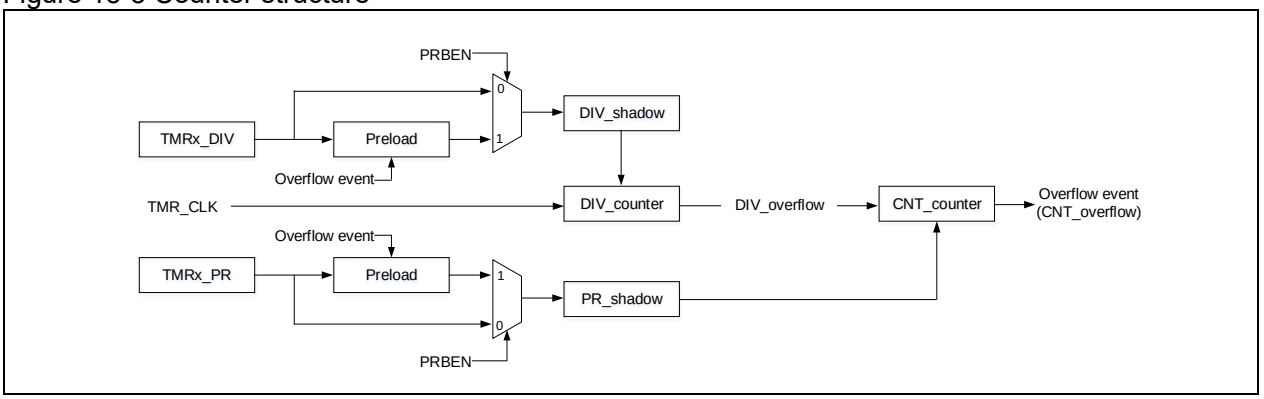
The TMRx_PR register is used to set the counting period. The value in the TMRx_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN=1 in the TMRx_CTRL1 register), the value in the TMRx_PR register is transferred to the shadow register only at an overflow event.

The TMRx_DIV register is used to set the counting frequency. The counter counts once every count clock period (DIV[15:0]+1). Similar to the TMRx_PR register, when the periodic buffer is enabled, the value in the TMRx_DIV register is updated to the shadow register at an overflow event.

Reading the TMRx_CNT register returns to the current counter value, and writing to the TMRx_CNT register updates the current counter value to the value being written.

An overflow event is generated by default. Set OVFEN=1 in the TMRx_CTRL1 to disable generation of update events. The OVFS bit in the TMRx_CTRL1 register is used to select overflow event source. By default, counter overflow/underflow, setting OVFSWTR bit and the reset signal generated by the slave timer controller in reset mode trigger the generation of an overflow event. When the OVFS bit is set, only counter overflow/underflow triggers an overflow event.
Setting the TMREN bit (TMREN=1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR_EN is set 1 clock cycle after the TMREN is set.

Figure 15-3 Counter structure



Upcounting mode

In this mode, the counter counts from 0 to the value programmed in the TMRx_PR register, restarts from 0, and generates a counter overflow event, with the OVFIF bit being set to 1. If the overflow event is disabled, the counter is no longer reloaded with the preload value and period value at a counter overflow event; otherwise, the counter is updated with the preload value and period value at an overflow event.

Figure 15-4 Overflow event when PRBEN=0

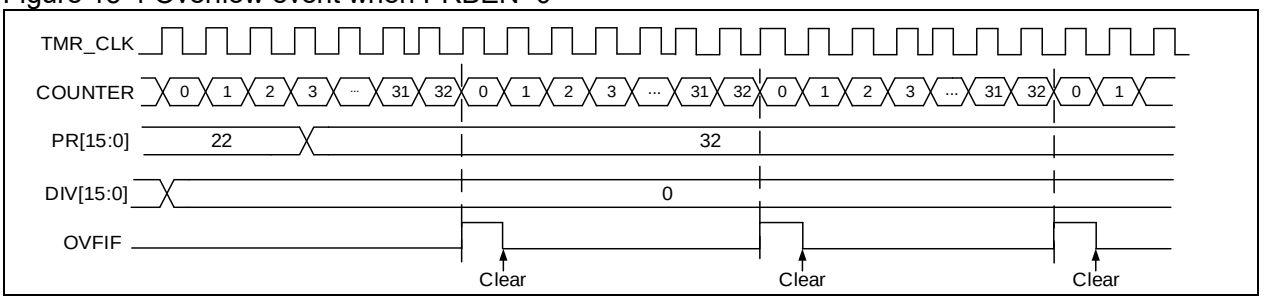


Figure 15-5 Overflow event when PRBEN=1

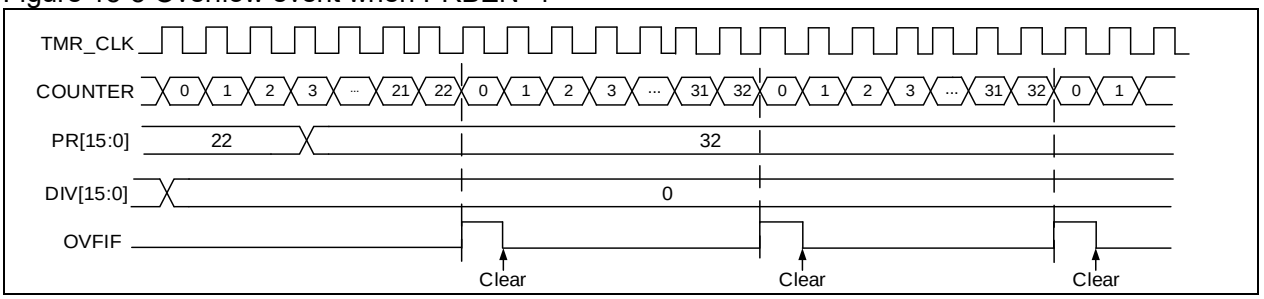
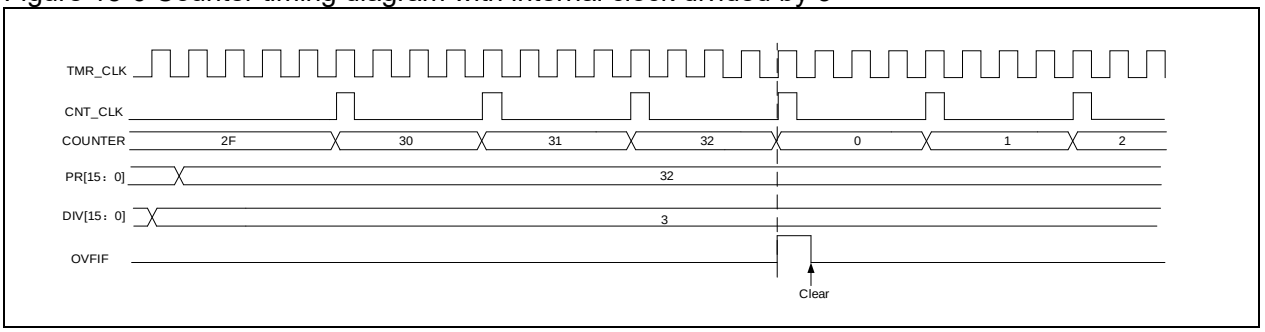


Figure 15-6 Counter timing diagram with internal clock divided by 3



15.1.3.3 Debug mode

When the microcontroller enters debug mode (Cortex ®-M4F core halted), the TMRx counter stops counting when the TMRx_PAUSE bit is set.

15.1.4 TMR6 and TMR7 registers

These peripheral registers must be accessed by words (32 bits).

Table 15-2 TMR6 and TMR7 register map and reset value

Register	Offset	Reset value
TMRx_CTRL1	0x00	0x0000 0000
TMRx_CTRL2	0x04	0x0000 0000
TMRx_IDEN	0x0C	0x0000 0000
TMRx_ISTS	0x10	0x0000 0000
TMRx_SWEVT	0x14	0x0000 0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000 0000
TMRx_PR	0x2C	0x0000 FFFF

15.1.4.1 TMR6 and TMR7 control register 1 (TMRx_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	PRBEN	0x0	rw	Period buffer enable 0: Period buffer is disabled. 1: Period buffer is enabled.
Bit 6: 4	Reserved	0x0	resd	Kept at its default value.
Bit 3	OCMEN	0x0	rw	One cycle mode enable This bit is used to select whether to stop the counter at overflow event. 0: Disabled 1: Enabled
Bit 2	OVFS	0x0	rw	Overflow event source This bit is used to configure overflow event or DMA request sources. 0: Counter overflow, or setting the OVFSWTR bit to generate overflow event 1: Only counter overflow generates an overflow event.
Bit 1	OVFEN	0x0	rw	Overflow event enable This bit is used to enable or disable OEV event generation. 0: OEV event is enabled. An overflow event is generated by any of the following events: - Counter overflow - Setting the OVFSWTR bit 1: OEV event is disabled. If the OVFSWTR bit is set to generate a software reset, the counter and the prescaler are reinitialized. Note: This bit is set and cleared by software.
Bit 0	TMREN	0x0	rw	TMR enable 0: Disabled 1: Enabled

15.1.4.2 TMR6 and TMR7 control register 2 (TMRx_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31: 7	Reserved	0x000 0000	resd	Kept at its default value.
Bit 6: 4	PTOS	0x0	rw	Master TMR output selection This field is used to select the signals in master mode to be sent to slave timers. 000: Reset 001: Enable 010: Update
Bit 3: 0	Reserved	0x0	resd	Kept at its default value.

15.1.4.3 TMR6 and TMR7 DMA/interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31: 9	Reserved	0x00 0000	resd	Kept at its default value.
Bit 8	OVFDEN	0x0	rw	Overflow event DMA request enable 0: Disabled 1: Enabled
Bit 7: 1	Reserved	0x0	resd	Kept at its default value.
Bit 0	OVFIEN	0x0	rw	Overflow interrupt enable 0: Disabled 1: Enabled

15.1.4.4 TMR6 and TMR7 interrupt status register (TMRx_ISTS)

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	OVFIF	0x0	rw0c	Overflow interrupt flag This bit is set by hardware at an overflow event. It is cleared by software. 0: No overflow event occurs. 1: Overflow event occurs, and OVFEN=0, and OVFS=0 in the TMRx_CTRL1 register: − An overflow event occurs when OVFSWTR=1 in the TMRx_SWEVT register

15.1.4.5 TMR6 and TMR7 software event register (TMRx_SWEVT)

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	OVFSWTR	0x0	rw0c	Overflow event triggered by software An overflow event is trigged by software. 0: No effect 1: Generate an overflow event by software write operation.

15.1.4.6 TMR6 and TMR7 counter value (TMRx_CVAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15;	CVAL	0x0000	rw	Counter value

15.1.4.7 TMR6 and TMR7 division register (TMRx_DIV)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	DIV	0x0000	rw	Divider value The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK}$ / (DIV[15: 0]+1). At each overflow event, DIV value is sent to the DIV register.

15.1.4.8 TMR6 and TMR7 period register (TMRx_PR)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	PR	0xFFFF	rw	Period value This indicates the period value of the TMRx counter. The timer stops working when the period value is 0.

15.2 General-purpose timers (TMR2 to TMR5)

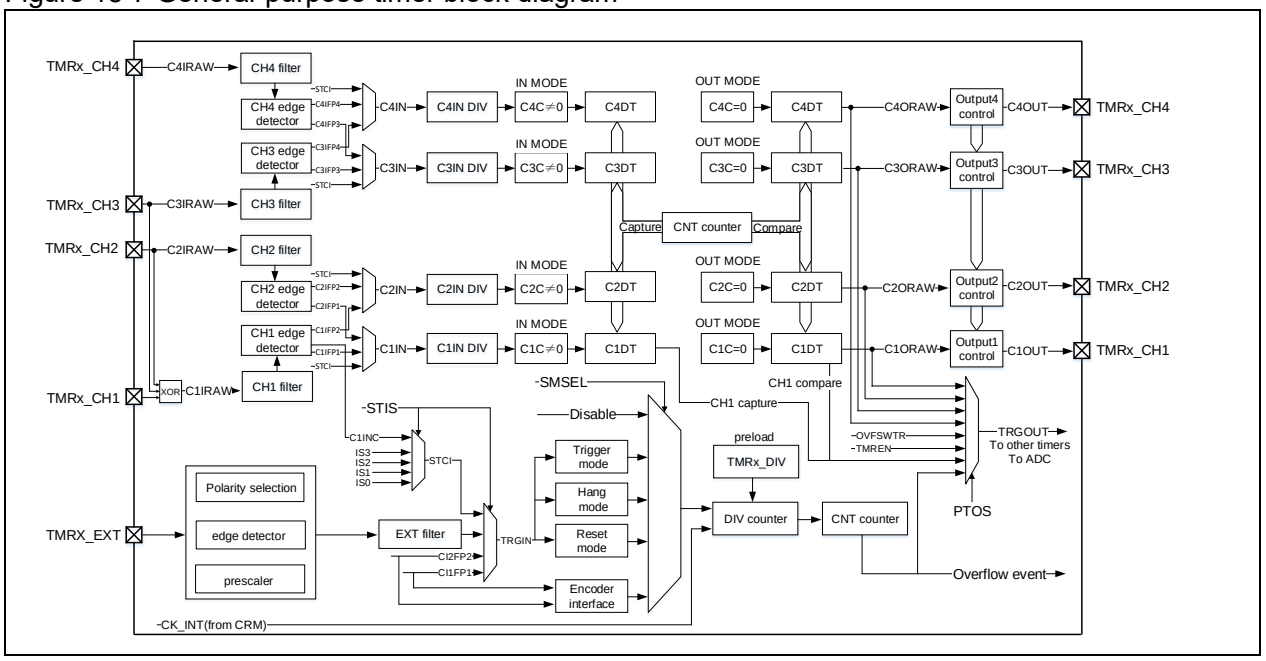
15.2.1 TMR2 to TMR5 introduction

The general-purpose timer (TMR2 to TMR5) consists of a 16-bit counter supporting up, down, up/down (bidirectional) counting modes, four capture/compare registers, and four independent channels to achieve input capture and programmable PWM output.

15.2.2 TMR2 to TMR5 main features

- Source of count clock is selectable : internal clock, external clock and internal trigger
- 16-bit up, down, up/down and encoder mode counter (TMR2/5 can be extended to 32-bit)
- 4 independent channels for input capture, output compare, PWM generation and one-pulse mode output
- Synchronization control between master and slave timers
- Interrupt/DMA generation at overflow event, trigger event and channel event
- Support TMR Brst DMA transfer

Figure 15-7 General-purpose timer block diagram

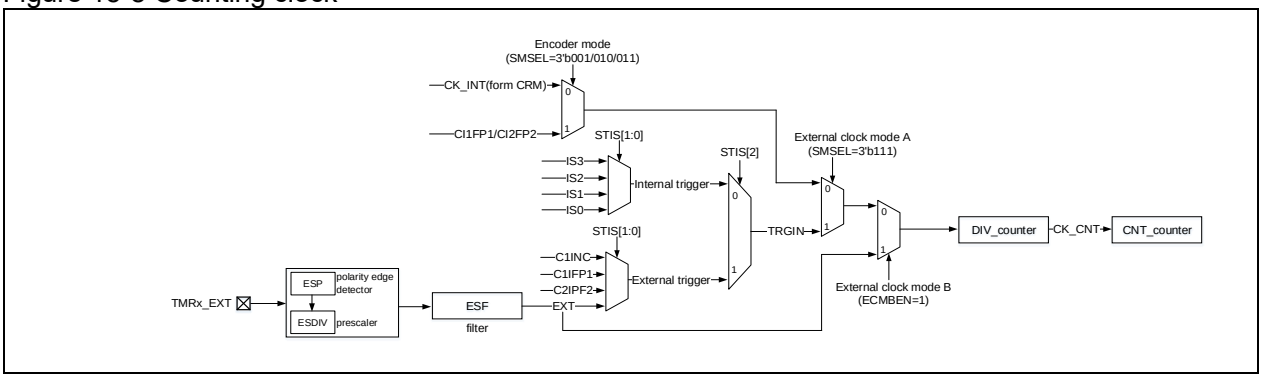


15.2.3 TMR2 to TMR5 functional overview

15.2.3.1 Counting clock

The counting clock of TMR2~TMR5 can be provided by the internal clock (CK_INT), external clock (external clock mode A and B) and internal trigger input (ISx).

Figure 15-8 Counting clock



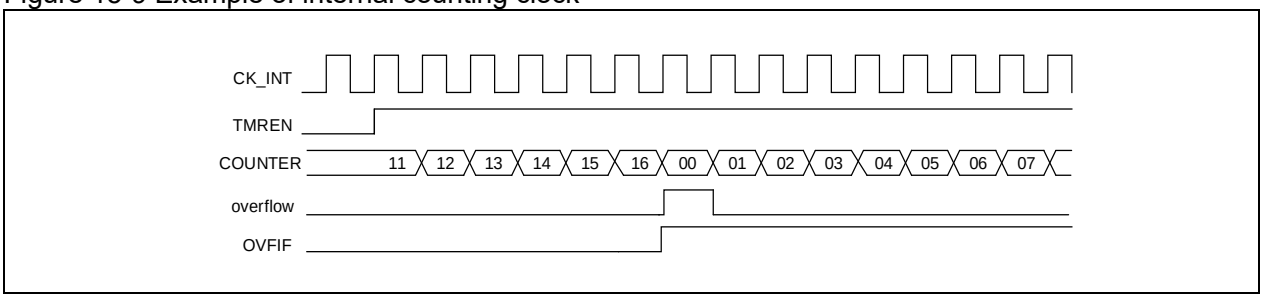
Internal clock (CK_INT)

By default, the CK_INT divided by a prescaler is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

The configuration process is as follows:

- Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 register to select count mode. If one-way count direction is to be used, configure OWCDIR bit in the TMRx_CTRL1 register to select direction;
- Set the TMRx_DIV register to set the counting frequency;
- Set the TMRx_PR register to set the counting period;
- Set the TMREN bit in the TMRx_CTRL1 register to enable the counter.

Figure 15-9 Example of internal counting clock



External clock (TRGIN/EXT)

The counter clock can be provided by two external clock sources, namely, TRGIN and EXT signals.

SMSEL=3'b111 in the TMRx_STCTRL register: external mode A is selected. Select an external clock source TRGIN signal by setting the STIS[2:0] bit in the TMRx_STCTRL register to drive the counter to start counting. The external clock sources include: C1INC (STIS=3'b100, channel 1 rising edge and falling edge), C1IFP1 (STIS=3'b101, channel 1 signal with filtering and polarity selection), C2IFP2 (STIS=3'b110, channel 2 signal with filtering and polarity selection) and EXT (STIS=3'b111, external input signal with polarity selection, frequency division and filtering).

ECMBEN=1 in the TMRx_STCTRL register: External clock mode B is selected. The counter is driven by external input that has gone through polarity selection, frequency division and filtering. The external clock mode B is equivalent to the external clock mode A which selects EXT signal as an external force TRGIN.

To use external clock mode A, follow the steps below:

- Set external source TRGIN parameters.

  If the TMRx_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx_CCTRL register);

If the TMRx_CH2 is used as source of TRGIN, it is necessary to configure channel 2 input filter (C2DF[3:0] in TMRx_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx_CCTR register);

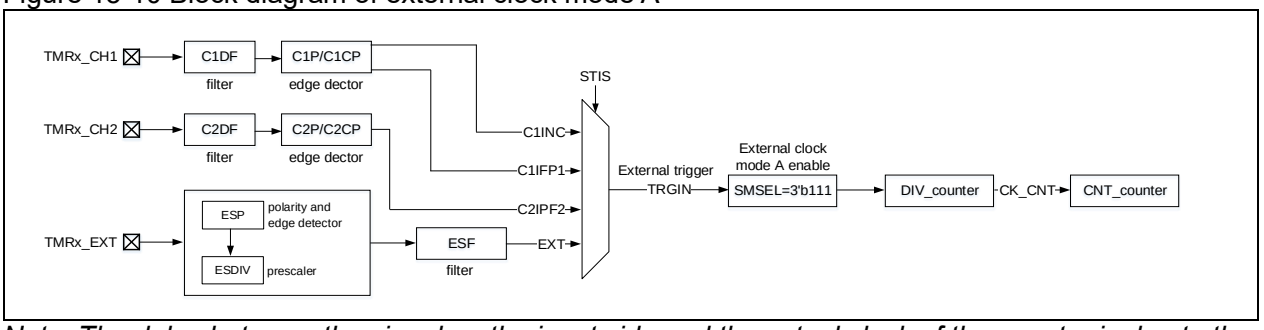
If the TMRx_EXT is used as a source of TRGIN, it is necessary to configure the external signal polarity (ESP in TMRx_STCTRL register), external signal frequency division (ESDIV[1:0] in TMRx_STCTRL) and external signal filter (ESF[3:0] in TMRx_STCTRL register).

- Set TRGIN signal source using the STIS[2:0] bit in TMRx_STCTRL register;
- Enable external clock mode A by setting SMSEL=3'b111 in TMRx_STCTR register;
- Set counting frequency through the DIV[15:0] in TMRx_DIV register;
- Set counting period through the PR[15:0] in TMRx_PR register;
- Enable counter through the TMREN bit in TMRx_CTRL1 register.

To use external clock mode B, follow the steps below:

- Set external signal polarity through the ESP bit in TMRx_STCTRL register;
- Set external signal frequency division through the ESDIV[1:0] bit in TMRx_STCTRL register;
- Set external signal filter through the ESF[3:0] bit in TMRx_STCTRL register;
- Enable external clock mode B through the ECMBEN bit in TMRx_STCTR register;
- Set counting frequency through the DIV[15:0] bit in TMRx_DIV register;
- Set counting period through the PR[15:0] bit in TMRx_PR register;
- Enable counter through the TMREN in TMRx_CTRL1 register.

Figure 15-10 Block diagram of external clock mode A



Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

Figure 15-11 Counting in external clock mode A

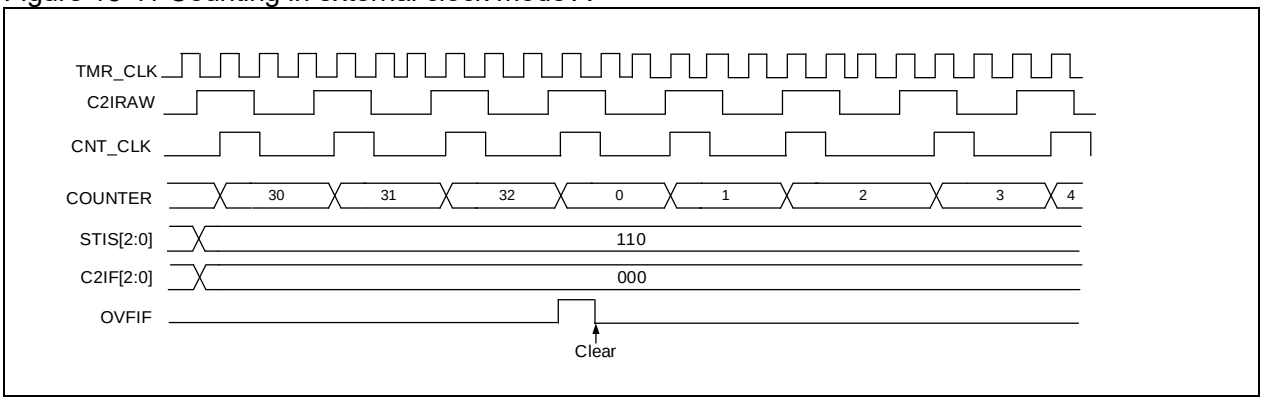
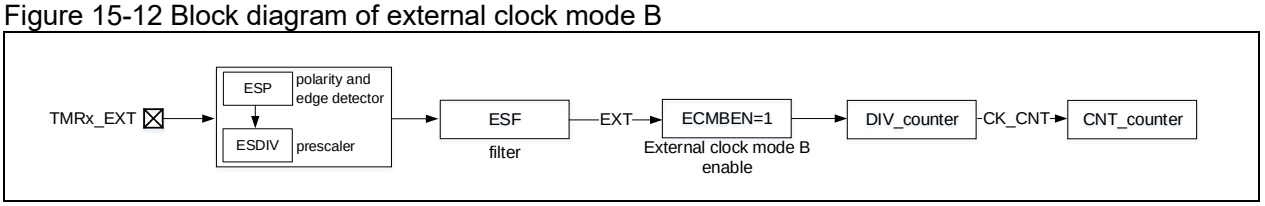
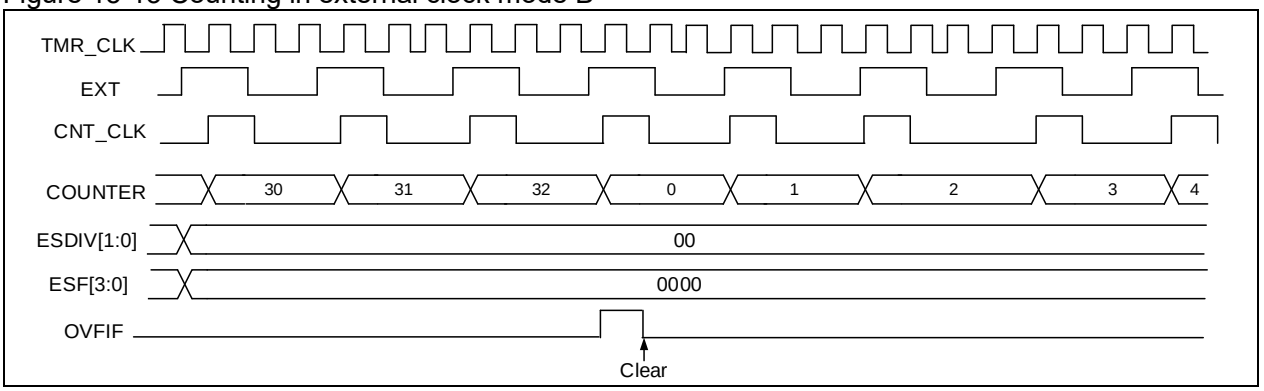


Figure 15-12 Block diagram of external clock mode B



Note: The delay between the EXT signal on the input side and the actual clock of the counter is due to the synchronization circuit.

Figure 15-13 Counting in external clock mode B



Internal trigger input (ISx)

Timer synchronization allows interconnection between several timers. The TMR_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2: 0] bit to select internal trigger signal to enable counting.

Each timer (TMR2 to TMR5) consists of a 16-bit prescaler, which is used to generate the CK_CNT that enables the counter to count. The frequency division relationship between the CK_CNT and TMR_CLK can be adjusted by setting the value of the TMRx_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

The internal trigger input is configured as follows:

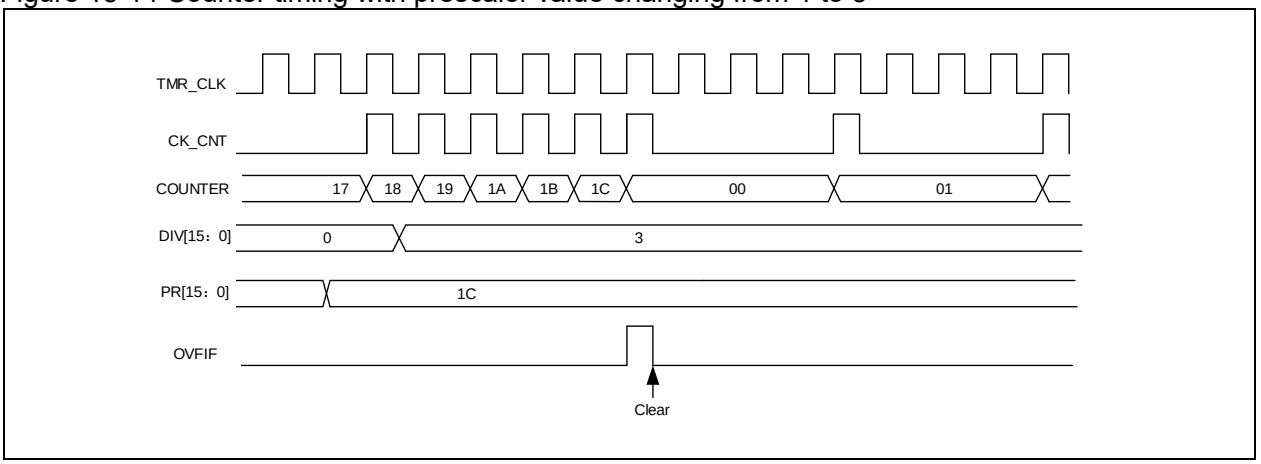
- Set the TMRx_PR register to set the counting period;
- Set the TMRx_DIV register to set the counting frequency;
- Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 register to set the count mode;
- Set the STIS[2:0] bit (range: 3'b000~3'b011) in the TMRx_STCTRL register and select internal trigger;
- Set SMSEL[2:0]=3'b111 in the TMRx_STCTRL register and select external clock mode A;
- Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.

Table 15-3 TMRx internal trigger connection

Slave controller	IS0 (STIS = 000)	IS1 (STIS = 001)	IS2 (STIS = 010)	IS3 (STIS = 011)
TMR2	TMR1	TMR8/USB_SOF/EMAC_PTP	TMR3	TMR4
TMR3	TMR1	TMR2	TMR5	TMR4
TMR4	TMR1	TMR2	TMR3	TMR8
TMR5	TMR2	TMR3	TMR4	TMR8

Note: If there is no corresponding timer in a device, the corresponding trigger signal ISx is not present.

Figure 15-14 Counter timing with prescaler value changing from 1 to 3



15.2.3.2 Counting mode

The timer (TMR2 to TMR5) supports several counting modes to meet different application scenarios. Each timer has an internal 16-bit upcounter, downcounter, upcounter/downcounter. TMR2/5 can be extended to 32-bit by setting the PMEN bit to 1 in the TMRx_CTRL1 register.

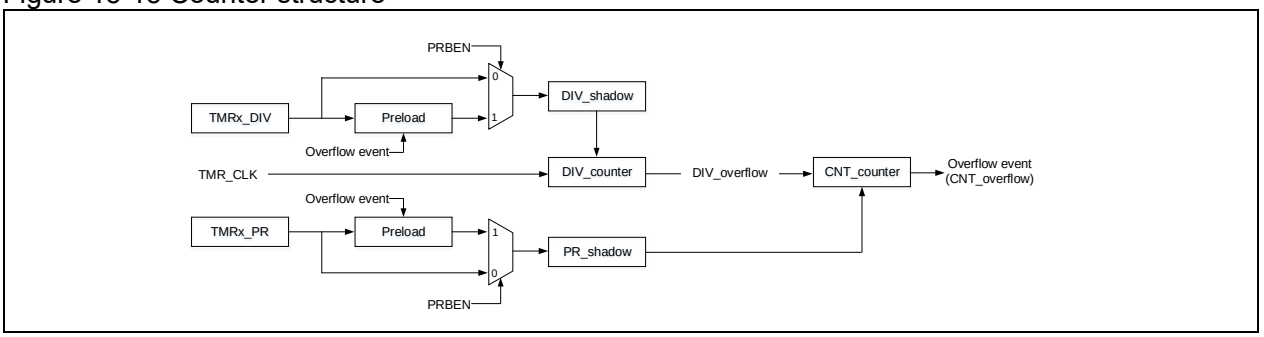
The TMRx_PR register is used to set the counting period. The value in the TMRx_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN=1 in the TMRx_CTRL1 register.), the value in the TMRx_PR register is transferred to the shadow register only at an overflow event. The OVFEN and OVFS bits are used to configure the overflow event.

The TMRx_DIV register is used to configure the counting frequency. The counter counts once every count clock period (DIV[15:0]+1). The value in the TMRx_DIV register is updated to the shadow register at an overflow event.

An overflow event is generated by default. Set OVFEN=1 in the TMRx_CTRL1 to disable generation of overflow events. The OVFS bit in the TMRx_CTRL1 register is used to select overflow event source. By default, counter overflow/underflow, setting OVFSWTR bit and the reset signal generated by the slave timer controller in reset mode trigger the generation of an overflow event. When the OVFS bit is set, only counter overflow/underflow triggers an overflow event.

Setting the TMREN bit (TMREN=1) enables the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR_EN is set 1 clock cycle after the TMREN is set.

Figure 15-15 Counter structure



Upcounting mode

Set TWCMSEL[1:0]=2'b00 and OWCDIR=1'b0 in the TMRx_CTRL1 register to enable upcounting mode. In this mode, the counter counts from 0 to the value programmed in the TMRx_PR register, then restarts from 0, and generates a counter overflow event, with the OVFIF bit being set to 1. If the overflow event is disabled, the counter is no longer reloaded with the preload value and period value at a counter overflow event; otherwise, the counter is updated with the preload value and period value on an overflow event.

Figure 15-16 Overflow event when PRBEN=0

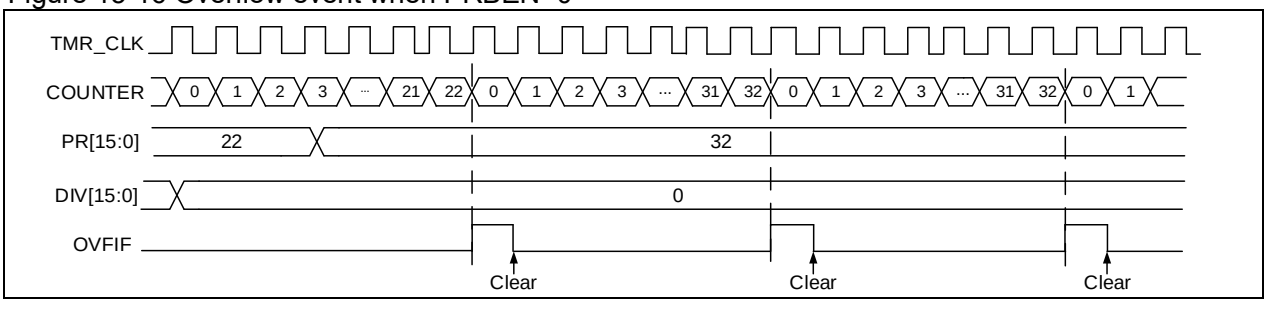
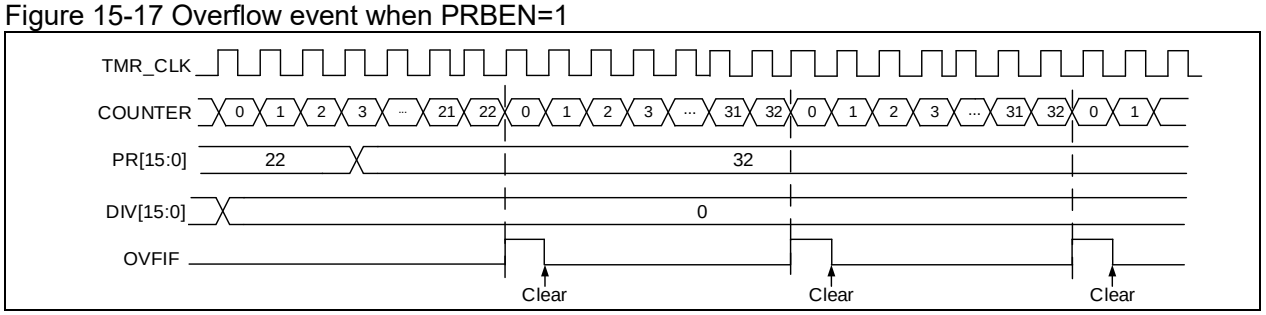


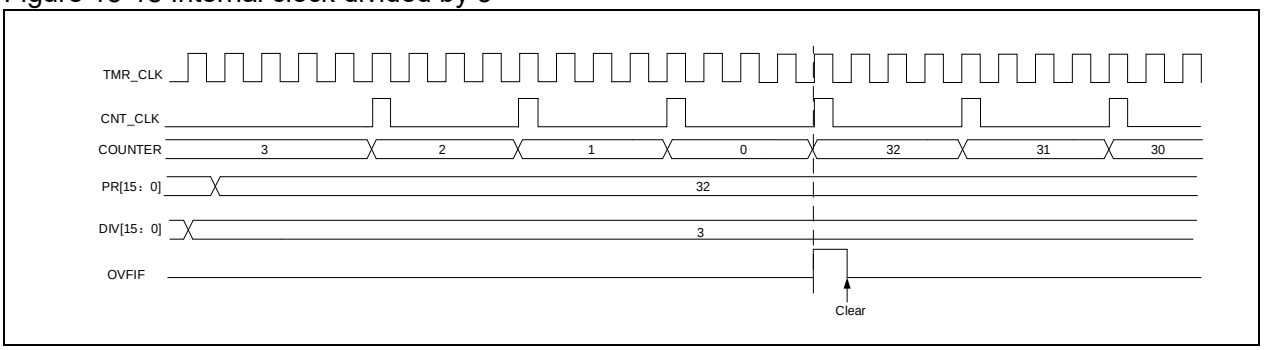
Figure 15-17 Overflow event when PRBEN=1



Downcounting mode

TWCMSEL[1:0]=2'b00 and OWCDIR=1'b1 in the TMRx_CTRL1register to enable downcounting mode. In this mode, the counter counts from the value programmed in the TMRx_PR register down to 0, and restarts from the value programmed, and generates a counter underflow event.

Figure 15-18 Internal clock divided by 3



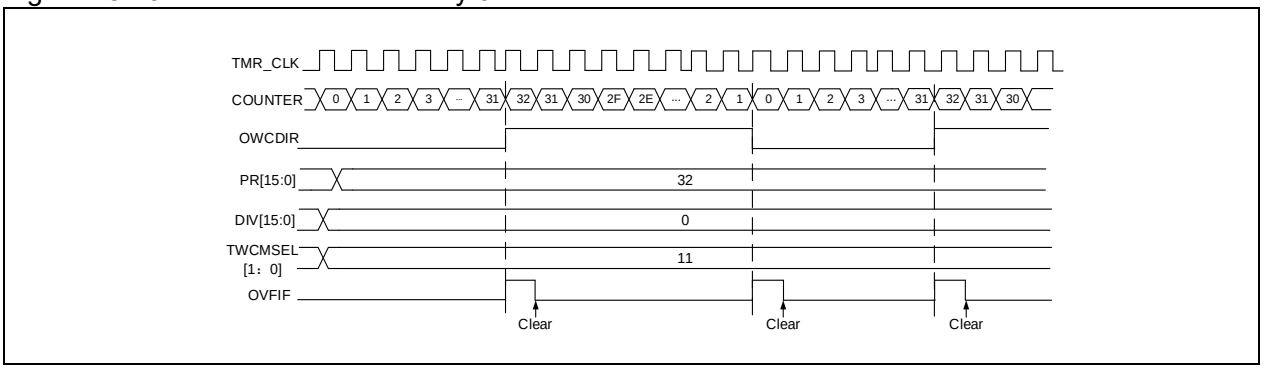
Up/down counting mode

Set TWCMSEL[1:0]≠2'b00 in the TMRx_CTRL1 register to enable up/down counting mode. In this mode, the counter counts up/down alternatively. When the counter counts from the value programmed in the TMRx_PR register down to 1, an underflow event is generated, and then restarts counting from 0; When the counter counts from 0 to the value of the TMRx_PR register -1, an overflow event is generated, and then restarts counting from the value of the TMRx_PR register. The OWCDIR bit indicates the current counting direction.

The TWCMSEL[1:0] bit in the TMRx_CTRL1 register is also used to select the CxIF flag setting method in up/down counting mode. In up/down counting mode 1 (TWCMSEL[1:0]=2'b01), CxIF flag can only be set when the counter counts down; in up/down counting mode 2 (TWCMSEL[1:0]=2'b10), CxIF flag can only be set when the counter counts up; in up/down counting mode 3 (TWCMSEL[1:0]=2'b11), CxIF flag can be set when the counter counts up/down.

Note: The OWCDIR is read-only in up/down counting mode.

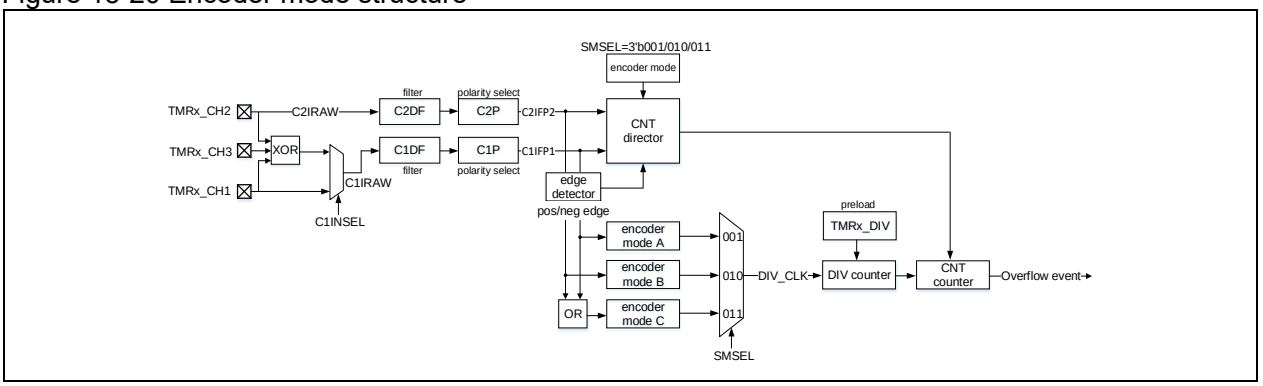
Figure 15-19 Internal clock divided by 0



Encoder interface mode

To enable the encoder interface mode, write SMSEL[2: 0]= 3'b001/3'b010/3'b011. In this mode, the two inputs (C1IN/C2IN) are required. Depending on the level on one input, the counter counts up or down on the edge of the other input. The OWCDIR bit indicates the direction of the counter.

Figure 15-20 Encoder mode structure



Encoder mode A: SMSEL=3'b001. The counter counts on the selected C1IFP1 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C1IFP1 and the level of C2IFP2.

Encoder mode B: SMSEL=3'b010. The counter counts on the selected C2IFP2 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C2IFP2 and the level of C1IFP1.

Encoder mode C: SMSEL=3'b011. The counter counts on both C1IFP1 and C2IFP2 edges (rising and falling edges). The counting direction is dependent on the C1IFP1 edge direction and C2IFP2 level, and C2IFP2 edge direction and C1IFP1 level.

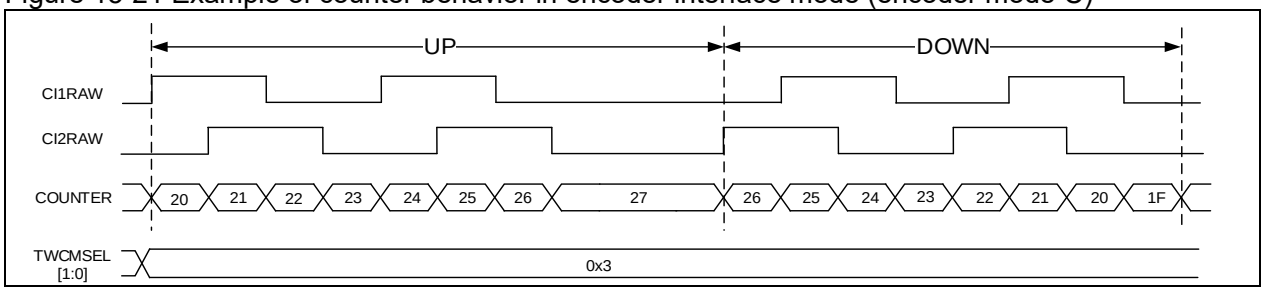
To use encoder mode, follow the procedures below:

- Set channel 1 input signal filtering through the C1DF[3:0] bit in the TMRx_CM1 register;
  Set channel 1 input signal active level through the C1P bit in the TMRx_CCTRL register.
- Set channel 2 input signal filtering through the C2DF[3:0] bit in the TMRx_CM1 register;
  Set channel 2 input signal active level through the C2P bit in the TMRx_CCTRL register.
- Set channel 1 as input mode through the C1C[1:0] bit in the TMRx_CM1 register;
  Set channel 2 as input mode through the C2C[1:0] bit in the TMRx_CM1 register.
- Select encoder mode A (SMSEL=3'b001), encoder mode B (SMSEL=3'b010), or encoder mode C (SMSEL=3'b011) by setting the SMSEL[2:0] bit in the TMRx_STCTRL register.
- Set counting cycles through the PR[15:0] bit in the TMRx_PR register.
- Set counting frequency through the DIV[15:0] bit in the TMRx_DIV register.
- Configure the corresponding IOs of TMRx_CH1 and TMRx_CH2 as multiplexed mode.
- Enable counter through the TMREN bit in the TMRx_CTRL1 register.

Table 15-4 Counting direction versus encoder signals

Active edge	Level on opposite signal (C1IFP1 to C2IFP2, C2IFP2 to C1IFP1)	C1IFP1 signal		C2IFP2 signal	
		Rising	Falling	Rising	Falling
C1IFP1	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
C2IFP2	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
C1IFP1 and C2IFP2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

Figure 15-21 Example of counter behavior in encoder interface mode (encoder mode C)



15.2.3.3 TMR input function

Each of TMR2~TMR5 timers has four independent channels, with each channel being configured as input or output.

As input, each channel input signal is handled as follows:

- TMRx_CHx outputs the pre-processed CxIRAW. The C1INSEL bit is used to select the source of C1IRAW from TMRx_CH1 or the XOR-ed TMRx_CH1, TMRx_CH2 and TMRx_CH3. The sources of C2IRAW, C3IRAW and C4IRAW are TMRx_CH2, TMRx_CH3 and TMRx_CH4, respectively.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit in the TMRx_CM1/CM2 register to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits in the TMRx_CCTRL register. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bit in the TMRx_CM1/CM2 register. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx (x≠y) is the CyIFPy signal that is from Y channel. The STCI comes from slave timer controller, and its source is selected by STIS bit.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit in the TMRx_CM1/CM2 register. It can be used for filtering, selection, division and input capture of input signals.

Figure 15-22 Input/output channel 1 main circuit

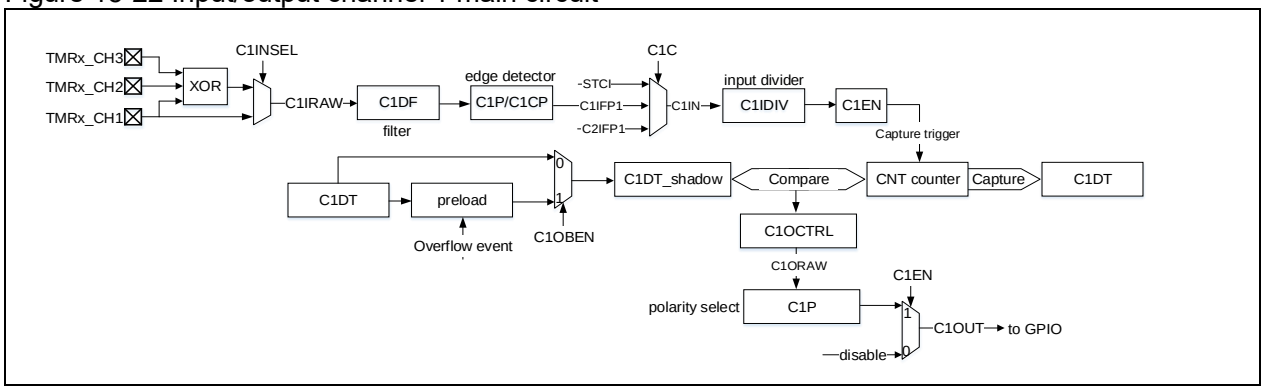
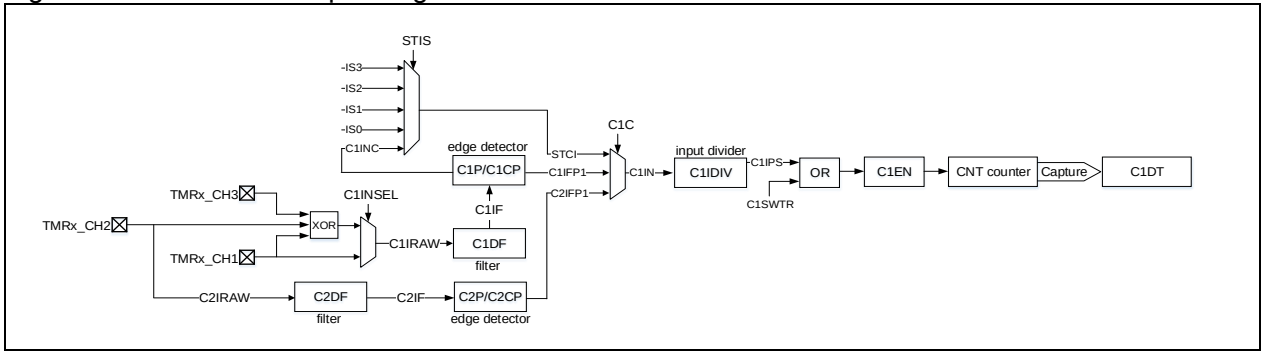


Figure 15-23 Channel 1 input stage



Input mode

In input mode, the TMRx_CxDT registers latch the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF in the TMRx_ISTS register) is set to 1. An interrupt or a DMA request will be generated if the CxIEN in the TMRx_IDEN register or CxDEN bit in the TMRx_IDEN register is enabled. If the selected trigger signal is detected when the CxIF is set, a capture overflow event occurs. The TMRx_CxDT register overwrites the recorded value with the current counter value, and the CxRF is set to 1 in the TMRx_ISTS register.

To capture the rising edge of C1IN input, following the configuration procedure mentioned below:

- Set C1C=2'b01 in the TMRx_CxDT register to select the C1IN as channel 1 input.
- Set the filter bandwidth of C1IN signal (CxDF[3: 0]).
- Set the active edge on the C1IN channel by writing C1P=1'b0 (rising edge) in the TMRx_CCTR register.
- Program the capture frequency division of C1IN signal (C1DIV[1: 0]).
- Enable channel 1 input capture (C1EN=1).
- If needed, enable interrupt or DMA request by setting the C1IEN or C1DEN bit in the TMRx_IDEN register.

Timer Input XOR function

The 3 timer input pins (TMRx_CH1, TMRx_CH2 and TMRx_CH3) are connected to the channel 1 (selected by setting the C1INSE in the TMRx_CTRL2 register) through an XOR gate.

The XOR gate can be used to connect Hall sensors. For example, connect the three XOR inputs to the three Hall sensors respectively so as to calculate the position and speed of the rotation by analyzing three Hall sensor signals.

Input selection

The TMR2 IS1 (internal trigger input 1) and TMR5 channel 4 are mappable through the TMRx_RMP register. The TMR2 IS1 can be configured as TMR8_TRGOUT, EMAC_PTP, USB_SOF. TMR5 channel 4 input can be configured as GPIO, LICK, LEXT or ERTC.

PWM input

The PWM input mode applies to channel 1 and channel 2. To enable this mode, map the C1IN and C2IN to the same TMRx_CHx, and configure the CxIFPx of channel 1/2 to trigger slave timer controller reset.

The PWM input mode can be used to measure the period and duty cycle of input signal. The period and duty cycle of channel 1 can be measured as follows:

- Set C1C=2'b01 in the TMRx_CM1 register to set C1IN as C1IFP1;
- Set C1P=1'b0 in the TMRx_CCTRL register to set C1IFP1 rising edge active;
- Set C2C=2'b10 in the TMRx_CM1 register to set C2IN as C1IFP2;
- Set C2P=1'b1 in the TMRx_CCTRL register to set C1IFP2 falling edge active;
- Set STIS=3'b101 in the TMRx_STCTRL register to set C1IFP1 as the slave timer trigger signal;
- Set SMSEL=3'b110 In the TMRx_STCTRL registerto set the slave timer in reset mode;
- Set C1EN=1'b1 and C2EN=1'b1 In the TMRx_cCTRL register to enable channel 1 and channel 2 input capture.

In these configurations, the rising edge of channel 1 input signal triggers capture and saves captured values to the C1DT register, and channel 1 input signal rising edge resets the counter. The falling edge of channel 1 input signal triggers capture and saves captured values to the C2DT register. The period and duty of channel 1 input signal can be calculated through C1DT and C2DT respectively.

Figure 15-24 Example of PWM input mode configuration

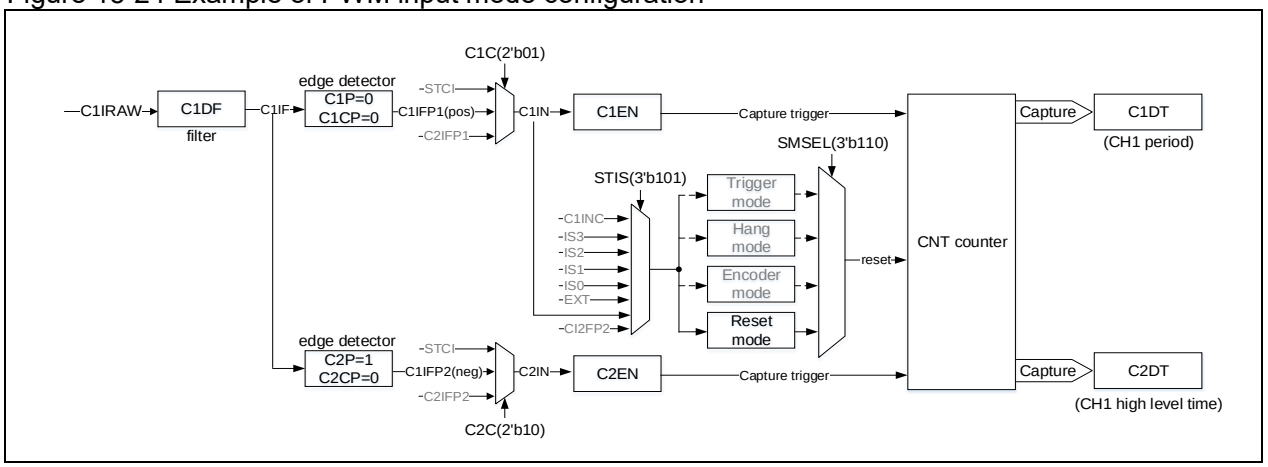
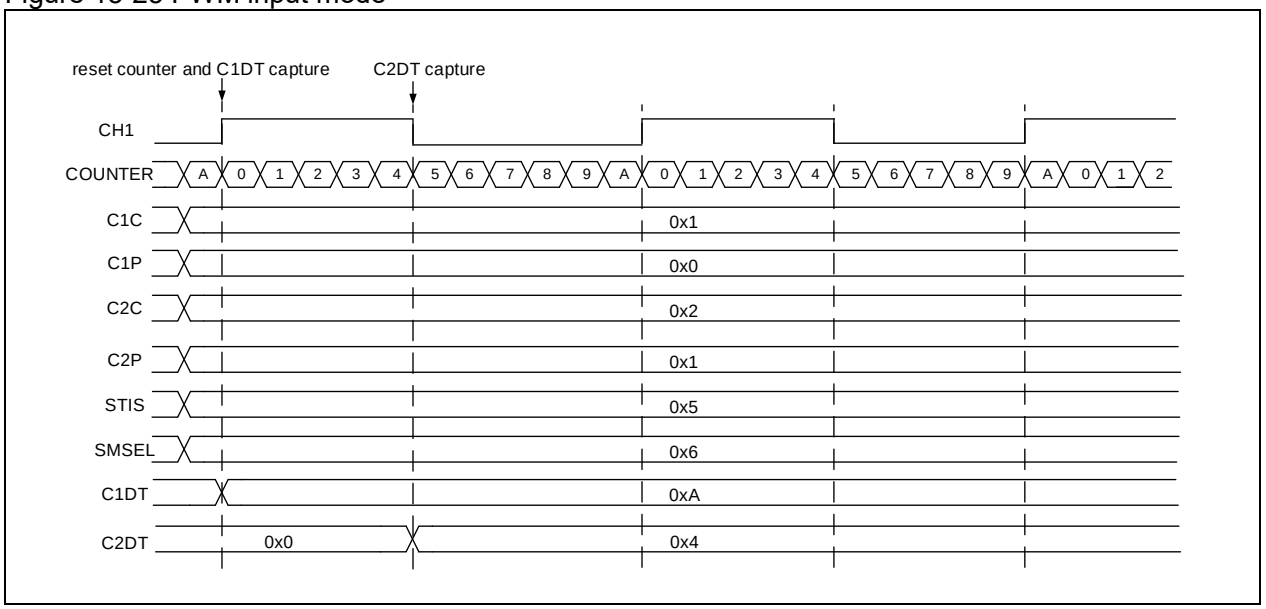


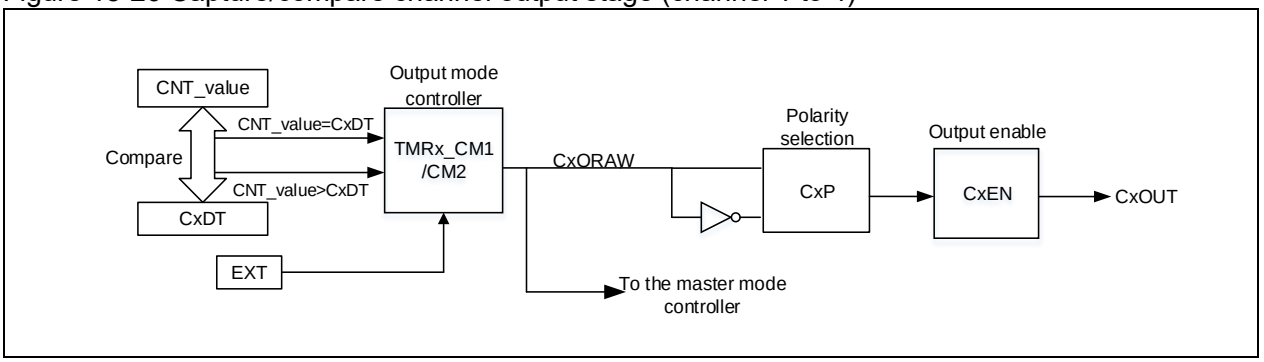
Figure 15-25 PWM input mode



15.2.3.4 TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal.

Figure 15-26 Capture/compare channel output stage (channel 1 to 4)



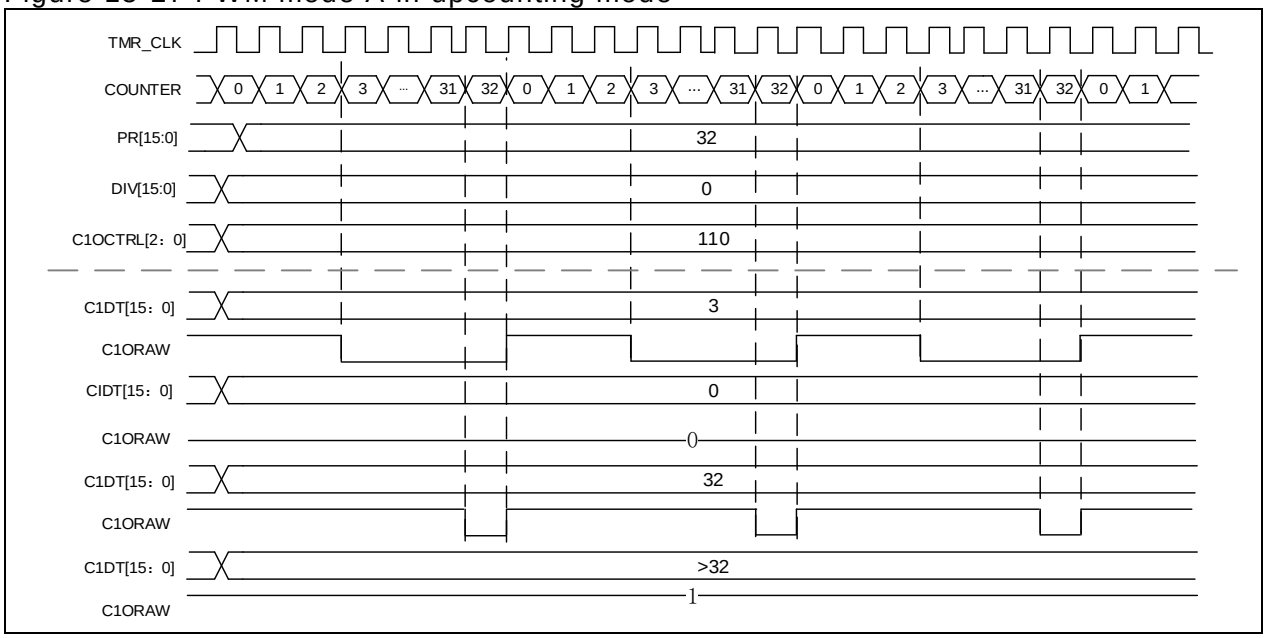
Output mode

Write CxC[1 0]≠2'b00 to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the TMRx_CxDT register, and the intermediate signal CxORAW is generated according to the output mode selected by CxOCTRL[2: 0], which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx_PR register, while the duty cycle by the TMRx_CxDT register.

Output compare modes include:

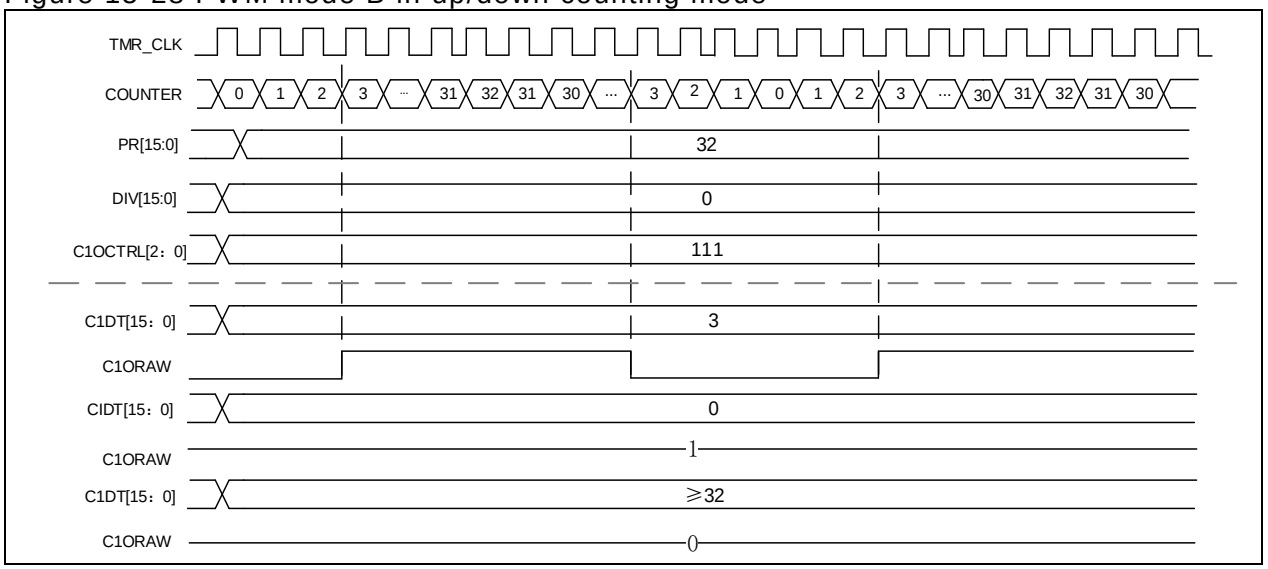
- **PWM mode A**: Set CxOCTRL=3'b110 to enable PWM mode A. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. Figure 15-27 gives an example of upcounting mode with PMW mode A with PR=0x32, and C1DT with different values. To use PWM mode A, the following process is recommended:
  - Set the TMRx_PR register to set PWM period;
  - Set the TMRx_CxDT register to set PWM duty cycle;
  - Set CxOCTRL=3'b110 in TMRx_CM1/CM2 register and set output mode as PWM mode A;
  - Set the TMRx_DIV register to set the counting frequency;
  - Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 to set the count mode;
  - Set the CxP bit in the TMRx_CCTRL register to set the output polarity;
  - Set the CxEN bit in the TMRx_CCTRL register to enable channel output;
  - Set the OEN bit in the TMRx_BRK register to enable TMRx output;
  - Set the corresponding GPIO of TMR output channel as the multiplexed mode;
  - Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.it

Figure 15-27 PWM mode A in upcounting mode



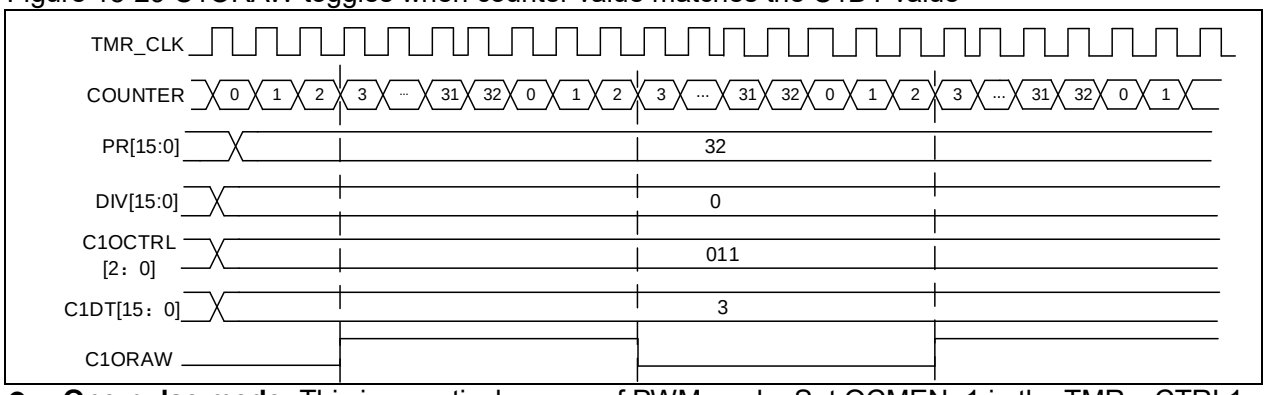
- **PWM mode B**: Set CxOCTRL=3'b111 to enable PWM mode B. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low. Figure 15-28 gives an example of up/down counting mode and PWM mode B, with PR=0x32 and CxDT with different values.

Figure 15-28 PWM mode B in up/down counting mode



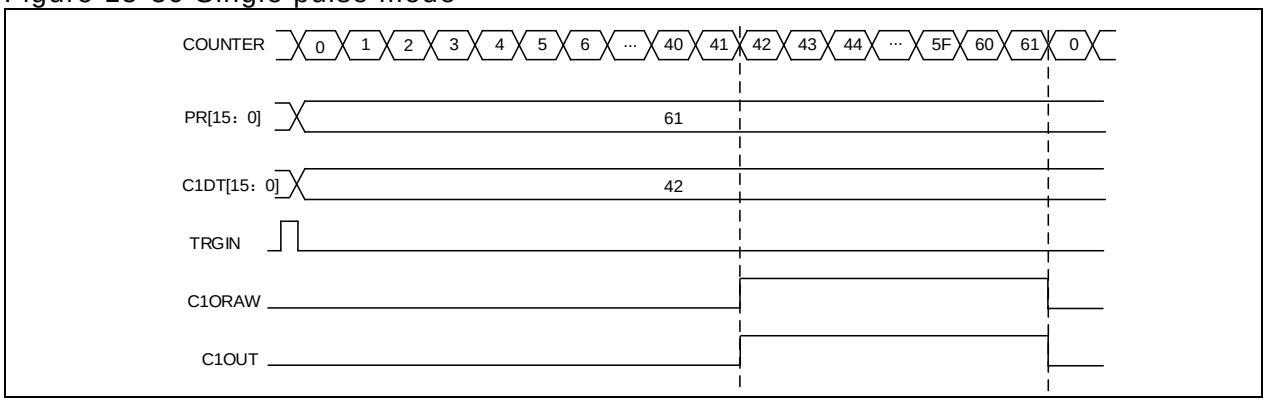
- **Forced output mode:** Set CxOCTRL=3'b100/101 to enable forced output mode. In this case, the CxORAW is forced to be the programmed level, irrespective of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.
- **Output compare mode:** Set CxOCTRL=3'b001/010/011 to enable output compare mode. In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high (CxOCTRL=3'b001), low (CxOCTRL=3'b010) or toggling (CxOCTRL=3'b011). Figure 15-29 gives an example of output compare mode (toggle), with C1DT=0x3 and counter value=0x3.

Figure 15-29 C1ORAW toggles when counter value matches the C1DT value



- **One-pulse mode:** This is a particular case of PWM mode. Set OCMEN=1 in the TMRx_CTRL1 register to enable one-pulse mode. In this mode, the comparison match is performed in the current counting period. The TMREN bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When configured as in upcounting mode, the configuration must follow the rule: CVAL<CxDT≤PR; in downcounting mode, CVAL>CxDT is required. Figure 15-30 shows an example of PWM mode B in upcounting mode and single pulse mode in which the counter only counts for a single cycle and there is a single pulse output.

Figure 15-30 Single pulse mode



- **Fast output mode:** Set CxOIEN=1 in the TMRx_CM1/CM2 register to enable this mode. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx_CxDT register will determine the level of CxORAW in advance.

Master timer event output

When TMR is selected as the master timer, the following signal sources can be selected as TRGOUT signal to output to the slave timer, by setting the PTOS bit in the TMRxCTRL2 register.

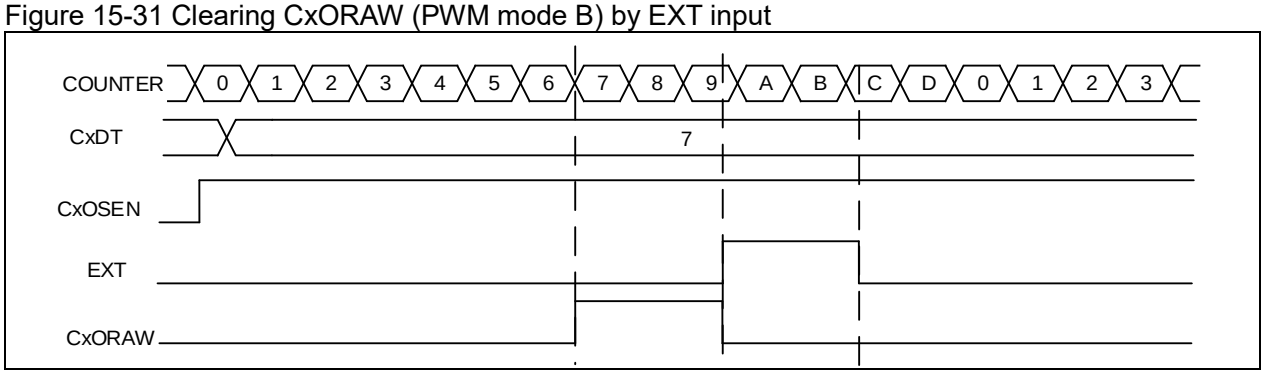
- PTOS=3'b000, TRGOUT outputs software overflow event (OVFSWTR bit in the TMRx_SWEVT register);
- PTOS=3'b001, TRGOUT outputs counter enable signal;
- PTOS=3'b010, TRGOUT outputs counter overflow event;
- PTOS=3'b011, TRGOUT outputs capture and compare event (C1DT);
- PTOS=3'b100, TRGOUT outputs C1ORAW signal;
- PTOS=3'b101, TRGOUT outputs C2ORAW signal;
- PTOS=3'b110, TRGOUT outputs C3ORAW signal;
- PTOS=3'b111, TRGOUT outputs C4ORAW signal.

CxORAW clear

When the CxOSEN bit is set to 1 in the TMRx_CM1/CM2 register , the CxORAW signal for a given channel is cleared by applying a high level to the EXT input. The CxORAW signal remains unchanged until the next overflow event.

This function can only be used in output capture or PWM modes, and does not work in forced mode. Figure 15-31 shows the example of clearing CxORAW signal. When the EXT input is high, the CxORAW signal, which was originally high, is driven low; when the EXT is low, the CxORAW signal outputs the corresponding level according to the comparison result between the counter value and CxDT value.

Figure 15-31 Clearing CxORAW (PWM mode B) by EXT input



15.2.3.5 TMR synchronization

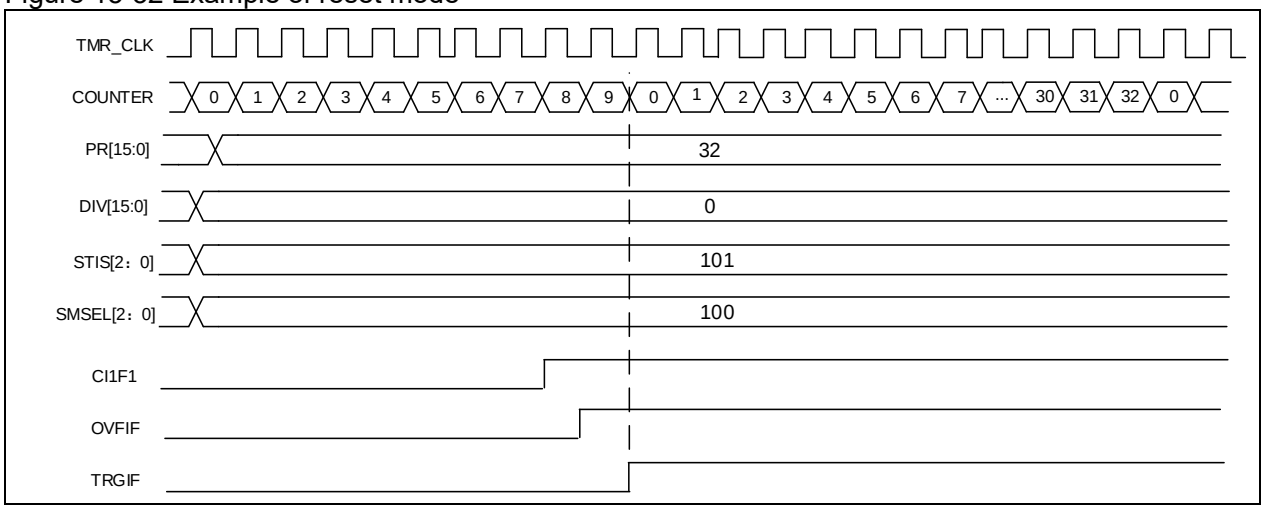
The timers are linked together internally for timer synchronization. Master timer is selected by setting the PTOS[2: 0] bit in the TMRx_CTRL2 register; Slave timer is selected by setting the SMSEL[2: 0] bit in the TMRx_STCTRL register .

Slave mode include:

Slave mode: Reset mode

The counter and its prescaler can be reset by a selected trigger signal. An overflow event is generated when OVFS=0 in the TMRx_CTRL1 register.

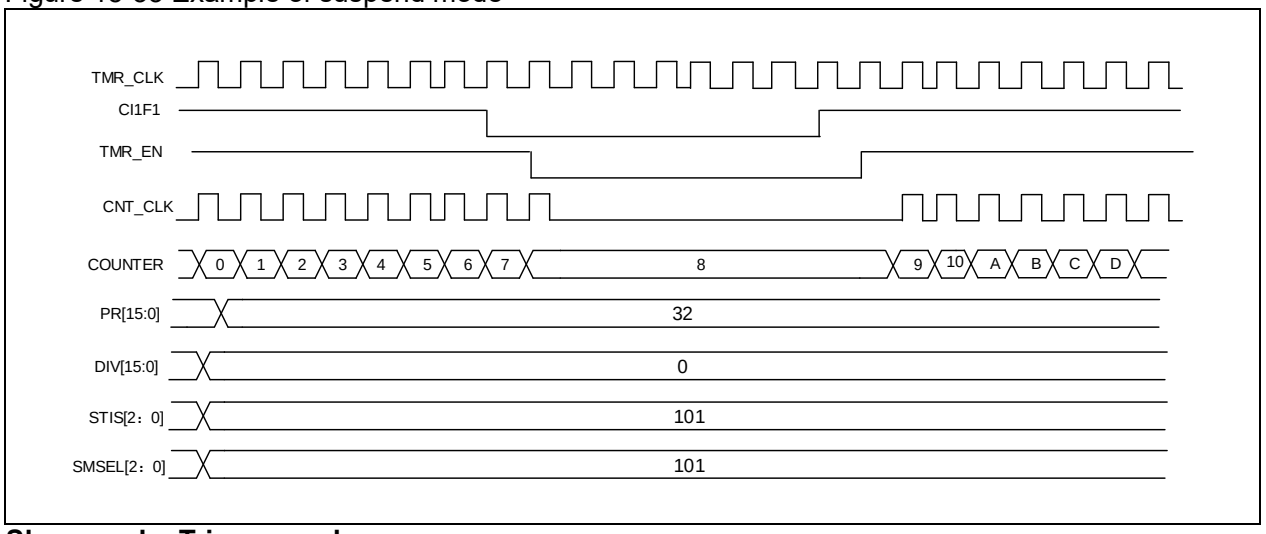
Figure 15-32 Example of reset mode



Slave mode: Suspend mode

In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.

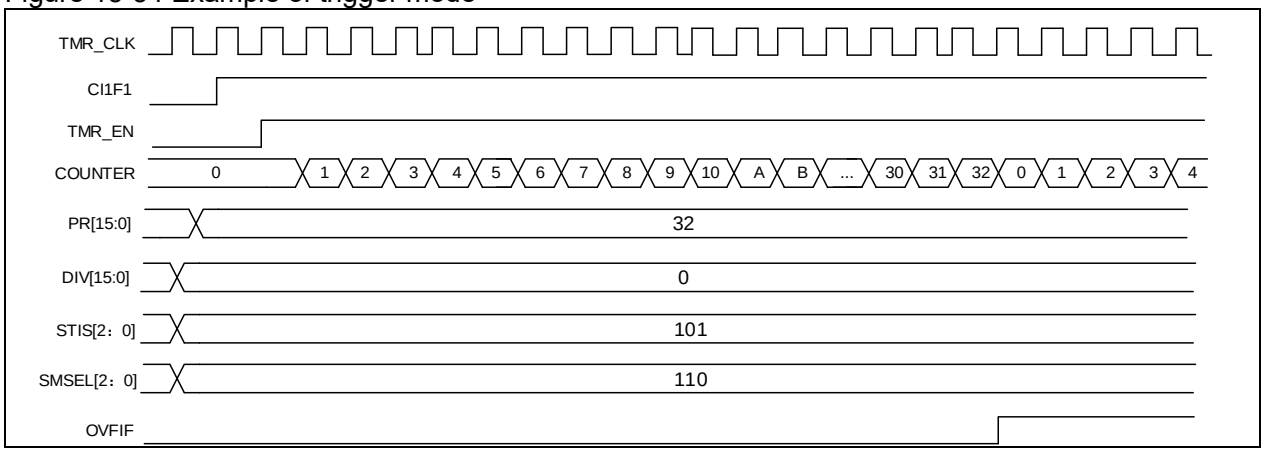
Figure 15-33 Example of suspend mode



Slave mode: Trigger mode

The counter can start counting on the rising edge of a selected trigger input (TMR_EN=1).

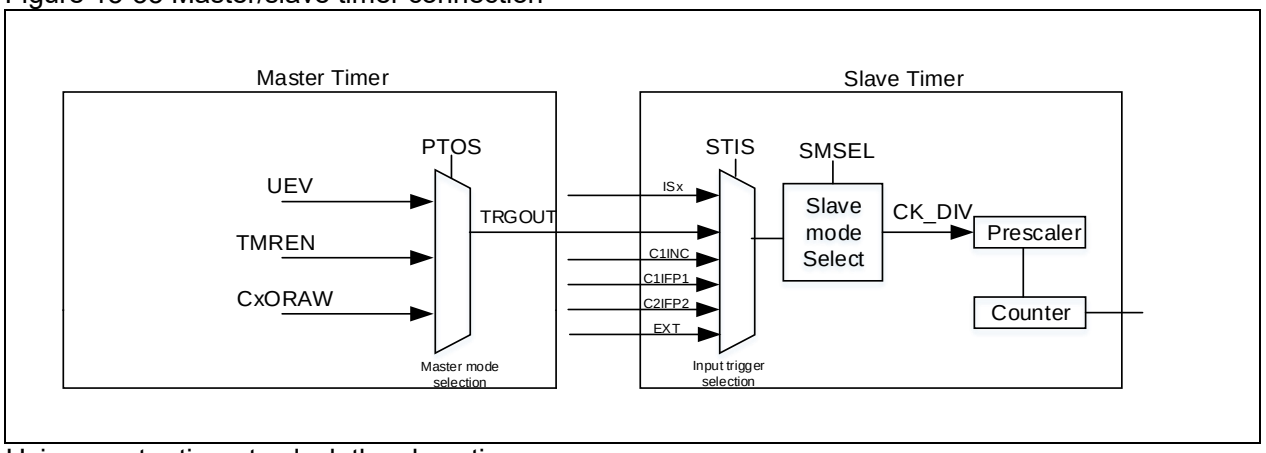
Figure 15-34 Example of trigger mode



Master/slave timer interconnection

Both Master and slave timer can be configured in different master and slave modes respectively. The combination of both them can be used for various purposes. The figure below provides an example of interconnection between master timer and slave timer.

Figure 15-35 Master/slave timer connection



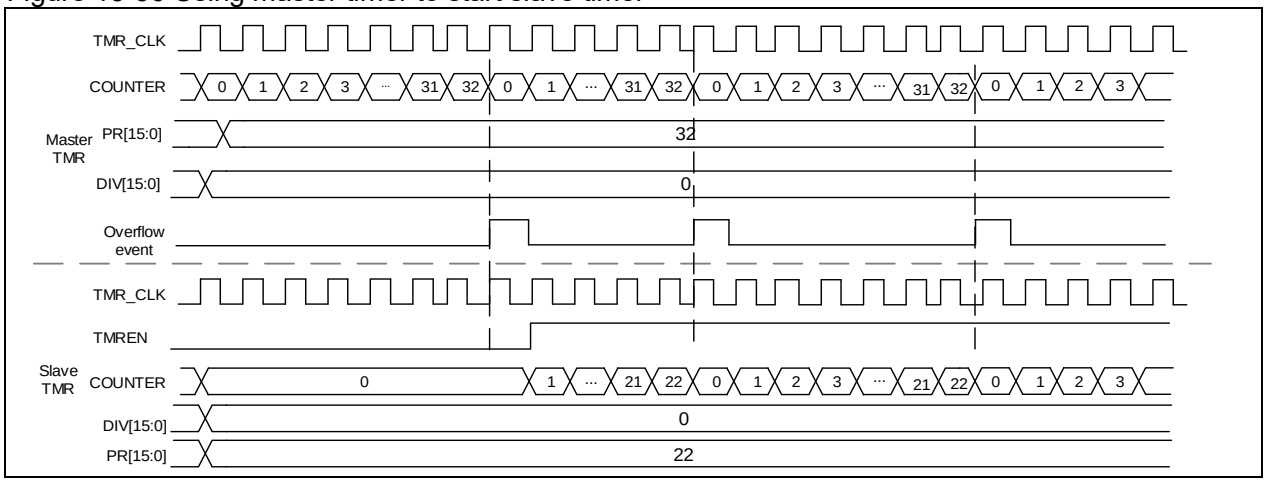
Using master timer to clock the slave timer:

- Configure master timer output signal TRGOUT as an overflow event (PTOS[2: 0]=3’b010) in the TMRx_CTRL2 register. The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure the master timer counting period (TMRx_PR registers).
- Configure the slave timer trigger input signal TRGIN as master timer output (STIS[2: 0] in the TMRx_STCTRL register).
- Configure the slave timer to use external clock mode A (SMSEL[2: 0]=3’b111 in the TMRx_STCTRL register).
- Set TMREN =1 in both master timer and slave timer to enable them.

Using master timer to start slave timer:

- Configure master timer output signal TRGOUT as an overflow event (PTOS[2: 0]=3’b010) in the TMRx_CTRL2 register. The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure master timer counting period (TMRx_PR registers).
- Configure slave timer trigger input signal TRGIN as master timer input.
- Configure slave timer as trigger mode (SMSEL=3’b110 in the TMRx_STCTRL register).
- Set TMREN=1 to enable master timer.

Figure 15-36 Using master timer to start slave timer

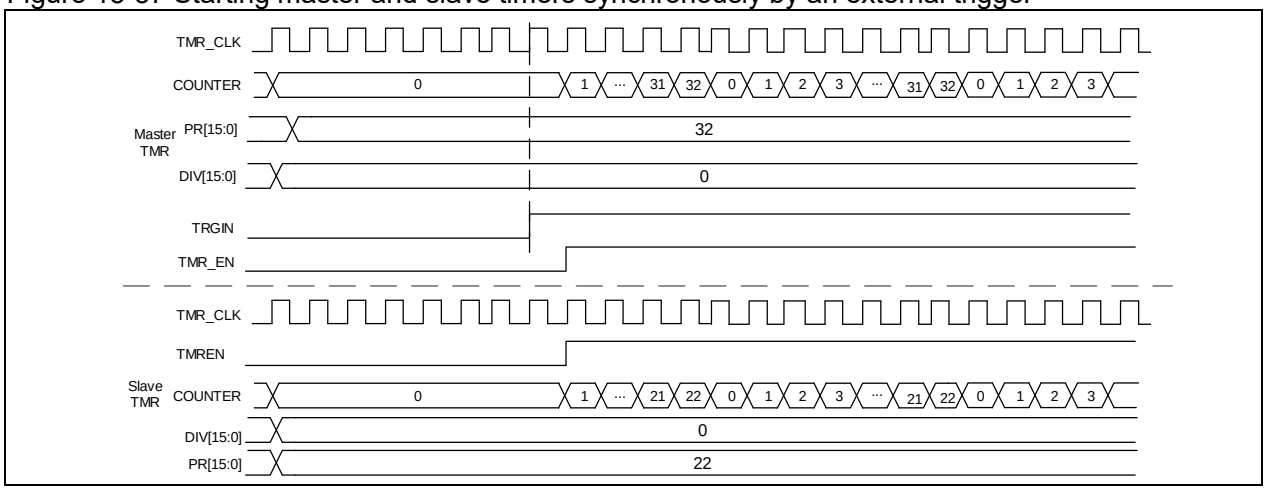


Starting master and slave timers synchronously by an external trigger:

In this example, configure the master timer as master/slave mode synchronously and enable its slave timer synchronization function. This mode is used for synchronization between master timer and slave timer.

- Set the STS bit to 1 of the master timer in the TMRx_STCTRL register
- Configure master timer output signal TRGOUT as an overflow event (PTOS[2: 0]=3'b010 in the TMRx_CTRL2 register). The master timer outputs a pulse signal at each counter overflow event, which is used as the counting clock of the slave timer.
- Configure the slave timer mode of the master timer as trigger mode, and select C1IN as trigger source.
- Configure slave timer trigger input signal TRGIN as master timer output.
- Configure slave timer as trigger mode (SMSEL=3'b110 in the TMRx_STCTRL register).

Figure 15-37 Starting master and slave timers synchronously by an external trigger



15.2.3.6 TMR DMA

TMR has the following events for DMA transfer request: overflow event DMA request, trigger event DMA rquest, Hall sensor DMA request and chanel event DMA request. It is possible to enable DMA request by setting the TMRx_IDEN register. Once enabled, upon an event generated, a DMA request is triggered and ouput to the DMA peripheral.

TMR DMA Burst feature

TMR also supports TMR DMA Burst feature. Thanks to this feature, DMA can be triggered to rewrite multiple TMR consecutive registers by enabling a certain DMA request through the TMRx_IDEN register.

As an overflow event to trigger TMR DMA Burst as an example:

- Enable overflow event to trigger DMA request using the OVFDEN bit in the TMRx_IDEN reigster
- Configure Burst transfer times using the DTB bit in the TMRx_DMACTRL register
- Configure the start address of Burst transfer using the ADDR bit in the TMRx_DMACTRL reigster
- Enable counter

After the above-mentioned configurations, here is the whole process of TMR DMA Burst transfer:

Upon an overflow event, the TMR will send a DMA request to the DMA peripheral. Then the DMA writes the data into the TMRx_DMADT register as requested. Inside TMR, the DMADT bit data is written into the start address register of Burst thransfer and an ACK single is sent to TMR. After receiving this ACK, the TMR clears the current DMA request; when the Burst transfer is detected by the TMR not to complete fully, a new overflow event for DMA request is released to the DMA so tthat the DMA writes the requested dta into the TMRx_DMADT again. In this case, inside tmr, the DMADT register data is written into the Burst transfer start address + 0x4 address register, and a new ACK is sent to TMR, and so on, and so on, until the last operation of Burst transfer. When a full Burst transfer is complete, the REQ for overflow event DMA request would not be set any more until the next overflow event.

Note: when using TMR DMA Burst feature, an empty register is prohibited and the TMRx_DMACTRL as well as TMRx_DMADT registers should not be available during the period from the start address to the end address.

15.2.3.7 Debug mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter stops counting by setting the TMRx_PAUSE in the DEBUG module..

15.2.4 TMRx registers

These peripheral registers must be accessed by words (32 bits).

Table 15-5 TMR2 and TMR 5 register map and reset value

Register	Offset	Reset value
TMRx_CTRL1	0x00	0x0000 0000
TMRx_CTRL2	0x04	0x0000 0000
TMRx_STCTRL	0x08	0x0000 0000
TMRx_IDEN	0x0C	0x0000 0000
TMRx_ISTS	0x10	0x0000 0000
TMRx_SWEVT	0x14	0x0000 0000
TMRx_CM1	0x18	0x0000 0000
TMRx_CM2	0x1C	0x0000 0000
TMRx_CCTRL	0x20	0x0000 0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000 0000
TMRx_PR	0x2C	0x0000 FFFF
TMRx_C1DT	0x34	0x0000 0000
TMRx_C2DT	0x38	0x0000 0000
TMRx_C3DT	0x3C	0x0000 0000
TMRx_C4DT	0x40	0x0000 0000
TMRx_DMACTRL	0x48	0x0000 0000
TMRx_DMADT	0x4C	0x0000 0000
TMR2_RMP	0x50	0x0000 0000
TMR5_RMP	0x50	0x0000 0000

15.2.4.1 TMR2 to TMR5 control register 1 (TMRx_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 11	Reserved	0x00 0000	resd	Kept at its default value.
Bit 10	PMEN	0x0	rw	Plus Mode Enable This bit is used to enable TMRx plus mode. In this mode, TMRx_CVAL, TMRx_PR and TMRx_CxDT are extended from 16-bit to 32-bit. 0: Disabled 1: Enabled *Note: This function is only valid for TMR2 and TMR5. It is not applicable to other TMRs.* *In plus mode or when disabled, only 16-bit value can be written to TMRx_CVAL, TMRx_PR and TMRx_CxDT registers.*
Bit 9: 8	CLKDIV	0x0	rw	Clock division This field is used to set the division ratio between digital filter sampling frequency (f_{DT}) and timer clock frequency (f_{CK_INT}). 00: No division, $f_{DTS}=f_{CK\_INT}$ 01: Divided by 2, $f_{DTS}=f_{CK\_INT}/2$

				10: Divided by 4, $f_{DTS}=f_{CK\_INT}/4$ 11: Reserved
Bit 7	PRBEN	0x0	rw	Period buffer enable 0: Period buffer is disabled 1: Period buffer is enabled
Bit 6: 5	TWCMSEL	0x0	rw	Two-way counting mode selection 00: One-way counting mode, depending on the OWCDIR bit 01: Two-way up/down counting mode1, count up and down alternately, the output flag bit is set only when the counter counts down 10: Two-way up/down counting mode2, count up and down alternately, the output flag bit is set only when the counter counts up 11: Two-way up/down counting mode3, count up and down alternately, the output flag bit is set when the counter counts up / down
Bit 4	OWCDIR	0x0	rw	One-way count direction 0: Up 1: Down
Bit 3	OCMEN	0x0	rw	One cycle mode enable This bit is use to select whether to stop counting at an overflow event 0: The counter does not stop at an overflow event 1: The counter stops at an overflow event
Bit 2	OVFS	0x0	rw	Overflow event source This bit is used to select overflow event or DMA request sources. 0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller 1: Only counter overflow generates an overflow event
Bit 1	OVFEN	0x0	rw	Overflow event enable 0: Enabled 1: Disabled
Bit 0	TMREN	0x0	rw	TMR enable 0: Disabled 1: Enabled

15.2.4.2 TMR2 to TMR5 control register 2 (TMRx_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	C1INSEL	0x0	rw	C1IN selection 0: CH1 pin is connected to C1IRAW input 1: The XOR result of CH1, CH2 and CH3 pins is connected to C1IRAW input
Bit 6: 4	PTOS	0x0	rw	Master TMR output selection This field is used to select the TMRx signal sent to the slave timer. 000: Reset 001: Enable 010: Update 011: Compare pulse (C1DT) 100: C1ORAW signal 101: C2ORAW signal 110: C3ORAW signal 111: C4ORAW signal
Bit 3	DRS	0x0	rw	DMA request source 0: Capture/compare event 1: Overflow event
Bit 2: 0	Reserved	0x0	resd	Kept at its default value.

15.2.4.3 TMR2 to TMR5 slave timer control register (TMRx_STCTRL)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15	ESP	0x0	rw	External signal polarity 0: High or rising edge 1: Low or falling edge
Bit 14	ECMBEN	0x0	rw	External clock mode B enable This bit is used to enable external clock mode B 0: Disabled 1: Enabled
Bit 13: 12	ESDIV	0x0	rw	External signal divide This field is used to select the frequency division of an external trigger 00: Normal 01: Divided by 2 10: Divided by 4 11: Divided by 8
Bit 11: 8	ESF	0x0	rw	External signal filter This field is used to filter an external signal. The external signal can be sampled only after it has been generated N times 0000: No filter, sampling by 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	STS	0x0	rw	Subordinate TMR synchronization If enabled, master and slave timer can be synchronized. 0: Disabled 1: Enabled
Bit 6: 4	STIS	0x0	rw	Subordinate TMR input selection This field is used to select the subordinate TMR input. 000: Internal selection 0 (IS0) 001: Internal selection 1 (IS1) 010: Internal selection 2 (IS2) 011: Internal selection 3 (IS3) 100: C1IRAW input detector (C1INC) 101: Filtered input 1 (C1IFP1) 110: Filtered input 2 (C1IFP2) 111: External input (EXT) Please refer to Table 15-3 for more information on ISx for each timer.
Bit 3	Reserved	0x0	resd	Kept at its default value
Bit 2: 0	SMSEL	0x0	rw	Subordinate TMR mode selection 000: Slave mode is disabled 001: Encoder mode A 010: Encoder mode B 011: Encoder mode C 100: Reset mode —Rising edge of the TRGIN input reinitializes the counter 101: Suspend mode — The counter starts counting when the TRGIN is high

				110: Trigger mode — A trigger event is generated at the rising edge of the TRGIN input 111: External clock mode A — Rising edge of the TRGIN input clocks the counter Note: Please refer to count mode section for the details on encoder mode A/B/C.

15.2.4.4 TMR2 to TMR5 DMA/interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31:15	Reserved	0x0 0000	resd	Kept at its default value
Bit 14	TDEN	0x0	rw	Trigger DMA request enable 0: Disabled 1: Enabled
Bit 13	Reserved	0x0	resd	Kept at its default value
Bit 12	C4DEN	0x0	rw	Channel 4 DMA request enable 0: Disabled 1: Enabled
Bit 11	C3DEN	0x0	rw	Channel 3 DMA request enable 0: Disabled 1: Enabled
Bit 10	C2DEN	0x0	rw	Channel 2 DMA request enable 0: Disabled 1: Enabled
Bit 9	C1DEN	0x0	rw	Channel 1 DMA request enable 0: Disabled 1: Enabled
Bit 8	OVFDEN	0x0	rw	Overflow event DMA request enable 0: Disabled 1: Enabled
Bit 7	Reserved	0x0	resd	Kept at its default value
Bit 6	TIEN	0x0	rw	Trigger interrupt enable 0: Disabled 1: Enabled
Bit 5	Reserved	0x0	resd	Kept at its default value
Bit 4	C4IEN	0x0	rw	Channel 4 interrupt enable 0: Disabled 1: Enabled
Bit 3	C3IEN	0x0	rw	Channel 3 interrupt enable 0: Disabled 1: Enabled
Bit 2	C2IEN	0x0	rw	Channel 2 interrupt enable 0: Disabled 1: Enabled
Bit 1	C1IEN	0x0	rw	Channel 1 interrupt enable 0: Disabled 1: Enabled
Bit 0	OVFIEN	0x0	rw	Overflow interrupt enable 0: Disabled 1: Enabled

15.2.4.5 TMR2 to TMR5 interrupt status register (TMRx_ISTS)

Bit	Name	Reset value	Type	Description
Bit 31: 13	Reserved	0x0 0000	resd	Kept at its default value
Bit 12	C4RF	0x0	rw0c	Channel 4 recapture flag Please refer to C1RF description.
Bit 11	C3RF	0x0	rw0c	Channel 3 recapture flag Please refer to C1RF description.
Bit 10	C2RF	0x0	rw0c	Channel 2 recapture flag Please refer to C1RF description.
Bit 9	C1RF	0x0	rw0c	Channel 1 recapture flag

				This bit indicates whether a recapture is detected when C1IF=1. This bit is set by hardware, and cleared by writing “0”. 0: No capture is detected 1: Capture is detected.
Bit 8: 7	Reserved	0x0	resd	Kept at its default value
Bit 6	TRGIF	0x0	rw0c	Trigger interrupt flag This bit is set by hardware on a trigger event. It is cleared by writing “0”. 0: No trigger event occurs 1: Trigger event is generated. Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode.
Bit 5	Reserved	0x0	resd	Kept at its default value
Bit 4	C4IF	0x0	rw0c	Channel 4 interrupt flag Please refer to C1IF description.
Bit 3	C3IF	0x0	rw0c	Channel 3 interrupt flag Please refer to C1IF description.
Bit 2	C2IF	0x0	rw0c	Channel 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF	0x0	rw0c	Channel 1 interrupt flag If the channel 1 is configured as input mode: This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT 0: No capture event occurs 1: Capture event is generated If the channel 1 is configured as output mode: This bit is set by hardware on a compare event. It is cleared by software. 0: No compare event occurs 1: Compare event is generated
Bit 0	OVFIF	0x0	rw0c	Overflow interrupt flag This bit is set by hardware on an overflow event. It is cleared by software. 0: No overflow event occurs 1: Overflow event is generated. If OVFEN=0 and OVFS=0 in the TMRx_CTRL1 register: − An overflow event is generated when OVFSWTR= 1 in the TMRx_SWEVT register; − An overflow event is generated when the counter CVAL is reinitialized by a trigger event.

15.2.4.6 TMR2 to TMR5 software event register (TMRx_SWEVT)

Bit	Name	Reset value	Type	Description
Bit 31: 7	Reserved	0x000 0000	resd	Kept at its default value.
Bit 6	TRGSWTR	0x0	rw	Trigger event triggered by software This bit is set by software to generate a trigger event. 0: No effect 1: Generate a trigger event.
Bit 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	C4SWTR	0x0	wo	Channel 4 event triggered by software Please refer to C1SWTR description.
Bit 3	C3SWTR	0x0	wo	Channel 3 event triggered by software Please refer to C1SWTR description.
Bit 2	C2SWTR	0x0	wo	Channel 2 event triggered by software Please refer to C1SWTR description
Bit 1	C1SWTR	0x0	wo	Channel 1 event triggered by software This bit is set by software to generate a channel 1 event. 0: No effect 1: Generate a channel 1 event.
Bit 0	OVFSWTR	0x0	wo	Overflow event triggered by software This bit is set by software to generate an overflow event. 0: No effect 1: Generate an overflow event.

15.2.4.7 TMR2 to TMR5 channel mode register 1 (TMRx_CM1)

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at default value.
Bit 15	C2OSEN	0x0	rw	Channel 2 output switch enable
Bit 14: 12	C2OCTRL	0x0	rw	Channel 2 output control
Bit 11	C2OBEN	0x0	rw	Channel 2 output buffer enable
Bit 10	C2OIEN	0x0	rw	Channel 2 output enable immediately
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS register.
Bit 7	C1OSEN	0x0	rw	Channel 1 output switch enable 0: C1ORAW is not affected by EXT 1: Once high level is detect on EXT input, clear C1ORAW.
Bit 6: 4	C1OCTRL	0x0	rw	Channel 1 output control This field defines the behavior of the original signal C1ORAW. 000: Disconnected. C1ORAW is disconnected from C1OUT; 001: C1ORAW is high when TMRx_CVAL=TMRx_C1DT 010: C1ORAW is low when TMRx_CVAL=TMRx_C1DT 011: Switch C1ORAW level when TMRx_CVAL=TMRx_C1DT 100: C1ORAW is forced low 101: C1ORAW is forced high. 110: PWM mode A – OWCDIR=0, C1ORAW is high once TMRx_C1DT>TMRx_CVAL, else low; – OWCDIR=1, C1ORAW is low once TMRx_ C1DT <TMRx_CVAL, else high; 111: PWM mode B – OWCDIR=0, C1ORAW is low once TMRx_ C1DT >TMRx_CVAL, else high; – OWCDIR=1, C1ORAW is high once TMRx_ C1DT <TMRx_CVAL, else low. *Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.*
Bit 3	C1OBEN	0x0	rw	Channel 1 output buffer enable 0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately. 1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.
Bit 2	C1OIEN	0x0	rw	Channel 1 output enable immediately In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event. 0: Need to compare the CVAL with C1DT before generating an output 1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration

				This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 12	C2DF	0x0	rw	Channel 2 digital filter
Bit 11: 10	C2IDIV	0x0	rw	Channel 2 input divider
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7: 4	C1DF	0x0	rw	Channel 1 digital filter This field defines the digital filter of the channel 1. "N" refers to the number of filtering, meaning that 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: $f_{SMPLING}=f_{DTS}/4$, 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 3: 2	C1IDIV	0x0	rw	Channel 1 input divider This field defines Channel 1 input divider. 00: No divider. An input capture is generated at each active edge. 01: An input compare is generated every 2 active edges 10: An input compare is generated every 4 active edges 11: An input compare is generated every 8 active edges Note: the divider is reset once C1EN='0'
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

15.2.4.8 TMR2 to TMR5 channel mode register 2 (TMRx_CM2)

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at default value.
Bit 15	C4OSEN	0x0	rw	Channel 4 output switch enable
Bit 14: 12	C4OCTRL	0x0	rw	Channel 4 output control
Bit 11	C4OBEN	0x0	rw	Channel 4 output buffer enable
Bit 10	C4OIEN	0x0	rw	Channel 4 output enable immediately
Bit 9: 8	C4C	0x0	rw	Channel 4 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN='0': 00: Output 01: Input, C4IN is mapped on C4IFP4 10: Input, C4IN is mapped on C3IFP4 11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7	C3OSEN	0x0	rw	Channel 3 output switch enable
Bit 6: 4	C3OCTRL	0x0	rw	Channel 3 output control
Bit 3	C3OBEN	0x0	rw	Channel 3 output buffer enable
Bit 2	C3OIEN	0x0	rw	Channel 3 output enable immediately
Bit 1: 0	C3C	0x0	rw	Channel 3 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN='0': 00: Output 01: Input, C3IN is mapped on C3IFP3 10: Input, C3IN is mapped on C4IFP3 11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 12	C4DF	0x0	rw	Channel 4 digital filter
Bit 11: 10	C4IDIV	0x0	rw	Channel 4 input divider
Bit 9: 8	C4C	0x0	rw	Channel 4 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN='0': 00: Output 01: Input, C4IN is mapped on C4IFP4 10: Input, C4IN is mapped on C3IFP4 11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7: 4	C3DF	0x0	rw	Channel 3 digital filter
Bit 3: 2	C3IDIV	0x0	rw	Channel 3 input divider
Bit 1:0	C3C	0x0	rw	Channel 3 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN='0': 00: Output 01: Input, C3IN is mapped on C3IFP3 10: Input, C3IN is mapped on C4IFP3 11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

15.2.4.9 TMR2 to TMR5 channel control register (TMRx_CCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 14	Reserved	0x0 0000	resd	Kept at its default value.
Bit 13	C4P	0x0	rw	Channel 4 polarity Please refer to C1P description.

Bit 12	C4EN	0x0	rw	Channel 4 enable Please refer to C1EN description.
Bit 11	C3CP	0x0	rw	Channel 3 complementary polarity Please refer to C1P description.
Bit 10	Reserved	0x0	resd	Default value
Bit 9	C3P	0x0	rw	Channel 3 polarity Please refer to C1P description.
Bit 8	C3EN	0x0	rw	Channel 3 enable Please refer to C1EN description.
Bit 7	C2CP	0x0	rw	Channel 2 complementary polarity Please refer to C1P description.
Bit 6	Reserved	0x0	resd	Kept at its default value.
Bit 5	C2P	0x0	rw	Channel 2 polarity Please refer to C1P description.
Bit 4	C2EN	0x0	rw	Channel 2 enable Please refer to C1EN description.
Bit 3	C1CP	0x0	rw	Channel 1 complementary polarity Please refer to C1P description.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	C1P	0x0	rw	Channel 1 polarity When the channel 1 is configured as output mode: 0: C1OUT is active high 1: C1OUT is active low When the channel 1 is configured as input mode: 00: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted. 01: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. 10: Reserved 11: C1IN active edge is on rising/falling edge. When used as external trigger, C1IN is not inverted.
Bit0	C1EN	0x0	rw	Channel 1 enable 0: Input or output is disabled 1: Input or output is enabled

Table 15-6 Standard CxOUT channel output control bit

CxEN bit	CxOUT output state
0	Output disabled (CxOUT=0, Cx_EN=0)
1	CxOUT = CxORAW + polarity, Cx_EN=1

Note: The state of the external I/O pins connected to the standard CxOUT channel depends on the CxOUT channel state and the GPIO and IOMUX registers.

15.2.4.10 TMR2 to TMR5 counter value (TMRx_CVAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	CVAL	0x0000	rw	Counter value When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the CVAL is expanded to 32 bits.
Bit 15: 0	CVAL	0x0000	rw	Counter value

15.2.4.11 TMR2 to TMR5 division value (TMRx_DIV)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DIV	0x0000	rw	Divider value The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK}$ /(DIV[15: 0]+1). DIV contains the value written at an overflow event.

15.2.4.12 TMR2 to TMR5 period register (TMRx_PR)

Bit	Name	Reset value	Type	Description
Bit 31: 16	PR	0x0000	rw	Period value When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the PR is expanded to 32 bits.
Bit 15: 0	PR	0xFFFF	rw	Period value This defines the period value of the TMRx counter. The timer stops working when the period value is 0.

15.2.4.13 TMR2 to TMR5 channel 1 data register (TMRx_C1DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	C1DT	0x0000	rw	Channel 1 data register When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C1DT is expanded to 32 bits.
Bit 15: 0	C1DT	0x0000	rw	Channel 1 data register When the channel 1 is configured as input mode: The C1DT is the CVAL value stored by the last channel 1 input event (C1IN). When the channel 1 is configured as output mode: C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured.

15.2.4.14 TMR2 to TMR5 channel 2 data register (TMRx_C2DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	C2DT	0x0000	rw	Channel 2 data register When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C2DT is expanded to 32 bits.
Bit 15: 0	C2DT	0x0000	rw	Channel 2 data register When the channel 2 is configured as input mode: The C2DT is the CVAL value stored by the last channel 2 input event (C2IN). When the channel 2 is configured as output mode: C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured.

15.2.4.15 TMR2 to TMR5 channel 3 data register (TMRx_C3DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	C3DT	0x0000	rw	Channel 3 data register When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C3DT is expanded to 32 bits.
Bit 15: 0	C3DT	0x0000	rw	Channel 3 data register When the channel 3 is configured as input mode: The C3DT is the CVAL value stored by the last channel 3 input event (C3IN). When the channel 3 is configured as output mode: C3DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C3OBEN bit, and the corresponding output is generated on C3OUT as configured.

15.2.4.16 TMR2 to TMR5 channel 4 data register (TMRx_C4DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	C4DT	0x0000	rw	Channel 4 data register When TMR2 or TMR5 enables plus mode (the PMEN bit in the TMR_CTRL1 register), the C4DT is expanded to 32 bits.
Bit 15: 0	C4DT	0x0000	rw	Channel 4 data register When the channel 4 is configured as input mode: The C4DT is the CVAL value stored by the last channel 4 input event (C4IN). When the channel 4 is configured as output mode: C4DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C4OBEN bit, and the corresponding output is generated on C4OUT as configured.

15.2.4.17 TMR2 to TMR5 DMA control register (TMRx_DMACTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 13	Reserved	0x0 0000	resd	Kept at its default value.
Bit 12: 8	DTB	0x00	rw	DMA transfer bytes This field defines the number of DMA transfers: 00000: 1 byte 00001: 2 bytes 00010: 3 bytes 00011: 4 bytes .... 10000: 17 bytes 10001: 18 bytes
Bit 7: 5	Reserved	0x0	resd	Kept at its default value.
Bit 4: 0	ADDR	0x00	rw	DMA transfer address offset ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register. 00000: TMRx_CTRL1 00001: TMRx_CTRL2 00010: TMRx_STCTRL .....

15.2.4.18 TMR2 to TMR5 DMA data register (TMRx_DMADT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DMADT	0x0000	rw	DMA data register A read or write operation to the DMADT register accesses the TMR registers at the following address: TMRx peripheral address + ADDR*4 to TMRx peripheral address + ADDR*4 + DTB*4.

15.2.4.19 TMR5 channel input remapping register (TMR2_RMP)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x0 0000	resd	Kept at its default value.
Bit 11: 10	TMR2_IS1_IRMP	0x0	rw	TMR2 IS1 input remap 00: TMR8_TRGOUT 01: EMAC_PTP 1x: USB_SOF
Bit 9:0	Reserved	0x000	resd	Kept at its default value.

15.2.4.20 TMR5 channel input remapping register (TMR5_RMP)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7: 6	TMR5_CH4_IRMP	0x0	rw	TMR5 channel 4 input remap 00: TMR5 channel 4 input connected to GPIO 01: Internal clock LICK 10: Internal clock LEXT 11: ERTC wakeup interrupt
Bit 5: 0	Reserved	0x00	resd	Kept at its default value.

15.3 General-purpose timers (TMR9 to TMR14)

15.3.1 TMR9 to TMR14 introduction

The general-purpose timers (TMR9 to TMR14) consist of a 16-bit counter supporting up, down, up/down (bidirectional) counting modes. they can be used for input capture and programmable PWM output. These timers can be synchronized

TMR9/TMR12 include two capture/compare registers, and two independent channels and support dead-timer insertion.

TMR10/TMR11/TMR13/TMR14 incude one capture/compare register and one independent channel and support dead-time insertion.

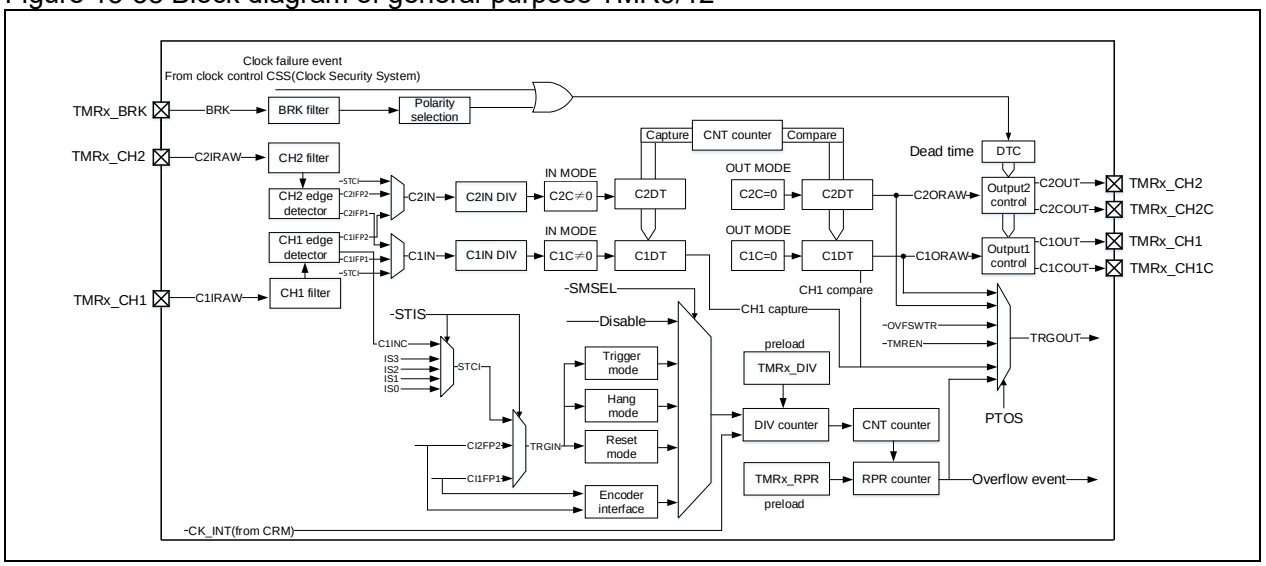
15.3.2 TMR9 to TMR14 main features

15.3.2.1 TMR9 and TMR12 main features

The main functions of general-purpose TMR9 and TMR12 include:

- Source of counter clock: internal clock and external clock
- 16-bit up, down, up/down counter, 8-bit repetition counter
- 2 independent channels for input capture, output compare, PWM generation, one-pulse mode output and dead-time insertion
- 2 independent channels for complementary output
- TMR break function
- Synchronization control between master and slave timers
- Interrupt/DMA interrupt generation at overflow event, trigger event, break input /channel event
- Support TMR DMA Burst transfers

Figure 15-38 Block diagram of general-purpose TMR9/12

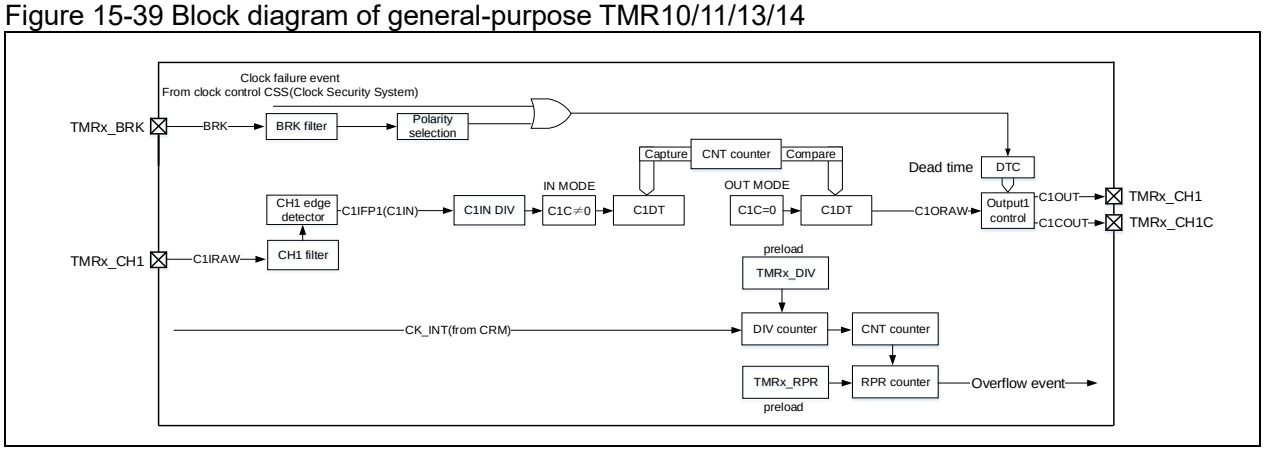


15.3.2.2 TMR10, TMR11, TMR13 and TMR14 main features

The main functions of general-purpose TMRx (TMR10, TMR11, TMR13 and TMR14) include:

- Source of counter clock: internal clock
- 16-bit up, down, up/downcounter, 8-bit repetition counter
- 1 independent channel for input capture, output compare, PWM generation, one-pulse mode and dead-time insertion
- TMR break function
- Interrupt/DMA generation at overflow event, break input and channel event
- Support TMR Burst DMA transfers

Figure 15-39 Block diagram of general-purpose TMR10/11/13/14

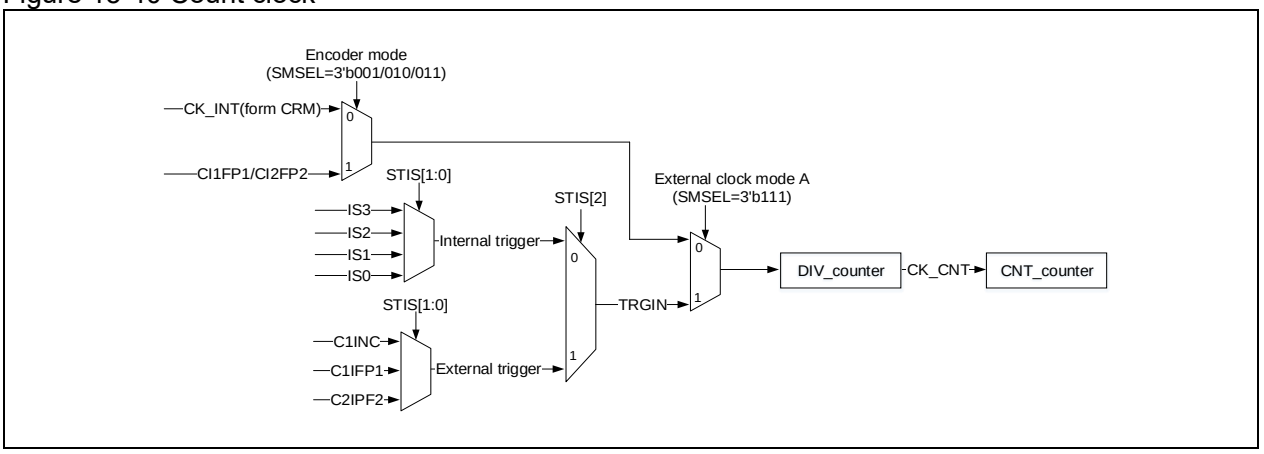


15.3.3 TMR9 to TMR14 functional overview

15.3.3.1 Counting clock

The count clock of general-purpose timers can be provided by the internal clock (CK_INT), external clock (external clock mode A) and internal trigger input (ISx).

Figure 15-40 Count clock



Internal clock (CK_INT)

By default, the CK_INT divided by the prescaler is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

The configuration process is as follows:

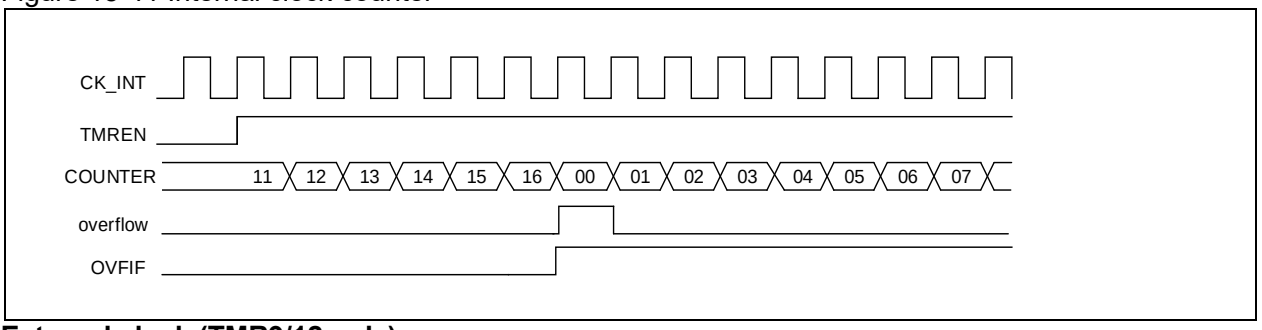
a) Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 register to select count mode. If the one-way count direction is set, configure OWCDIR bit in the TMRx_CTRL1 register to select the specific direction;

b) Set the TMRx_DIV register to set the counting frequency;

c) Set the TMRx_PR register to set the counting period;

d) Set the TMREN it in the TMRx_CTRL1 register to enable the counter.

Figure 15-41 Internal clock counter



External clock (TMR9/12 only)

The counter clock can be provided by TRGIN signal.

SMSEL=3'b111 in the TMRx_STCTRL register: External clock mode A is selected. Select an external clock source TRGIN signal by setting the STIS[2:0] bit in the TMRx_STCTRL register to drive the counter to start counting. The external clock sources include: C1INC (STIS=3'b100, channel 1 rising edge and falling edge), C1IFP1 (STIS=3'b101, channel 1 signal with filtering and polarity selection) and C2IFP2 (STIS=3'b110, channel 2 signal with filtering and polarity selection).

To use external clock mode A, follow the steps below:

- Set external source TRGIN parameters

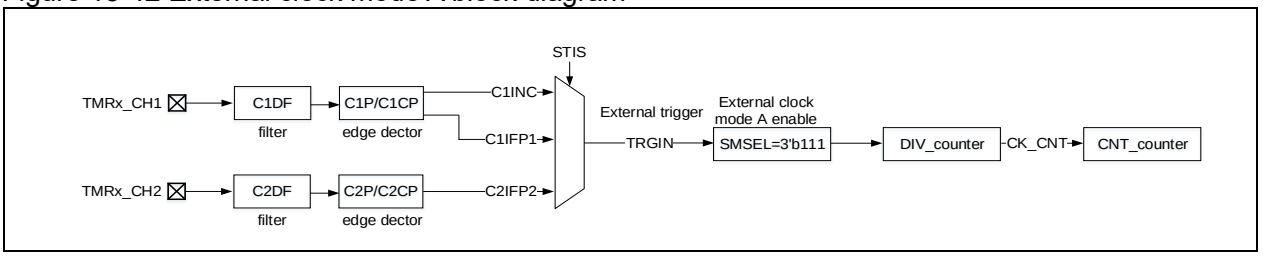
  If the TMRx_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx_CCTRL register);

  If the TMRx_CH2 is used as source of TRGIN, it is necessary to configure channel 1 input filter (C2DF[3:0] in TMRx_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx_CCTR register);

- Set TRGIN signal source using the STIS[2:0] bit in TMRx_STCTRL register
- Enable external clock mode A by setting SMSEL=3'b111 in TMRx_STCTR register
- Set counting frequency through the DIV[15:0] in TMRx_DIV register
- Set counting period through the PR[15:0] in TMRx_PR register

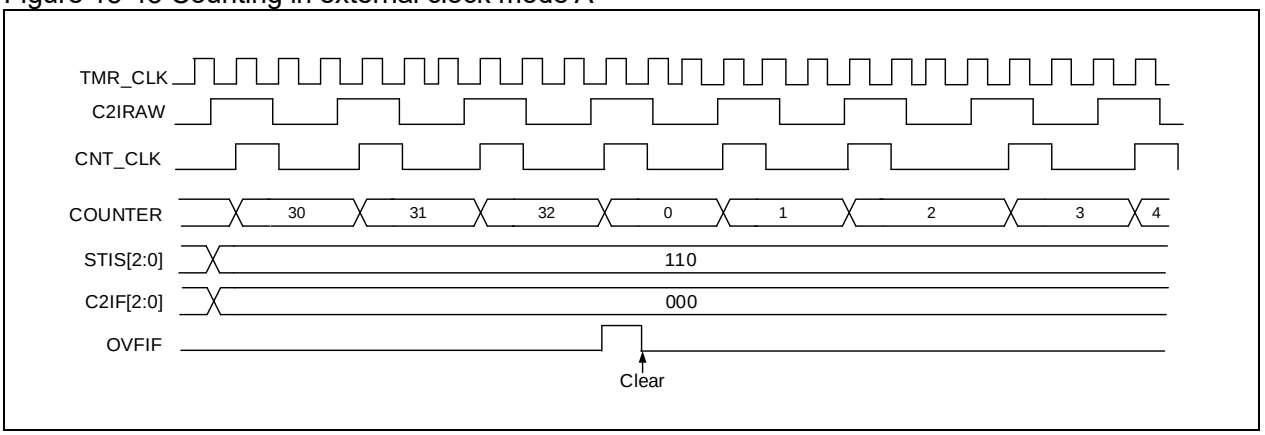
- Enable counter through the TMREN bit in TMRx_CTRL1 register

Figure 15-42 External clock mode A block diagram



Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

Figure 15-43 Counting in external clock mode A



Internal trigger input (ISx, TMR9/12 only)

Timer synchronization allows interconnection between several timers. The TMR_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2: 0] bit to select internal trigger signal to enable counting.

Each timer (TMR9 ~TMR14) consists of a 16-bit prescaler, which is used to generate the CK_CNT that enables the counter to count. The frequency division relationship between the CK_CNT and TMR_CLK can be adjusted by setting the value of the TMRx_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

The internal trigger input is configured as follows:

- Set the TMRx_PR register to set the counting period;
- Set the TMRx_DIV register to set the counting frequency;
- Set counter mode by setting the TWCMSEL[1:0] bit in the TMRx_CTRL1 register
- Set the STIS[2:0] bit (range: 3'b000~3'b011) in the TMRx_STCTRL register and select internal trigger;
- Set SMSEL[2:0]=3'b111 in the TMRx_STCTRL register and select external clock mode A;
- Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.

Figure 15-44 Counter timing with prescaler value changing from 0 to 3

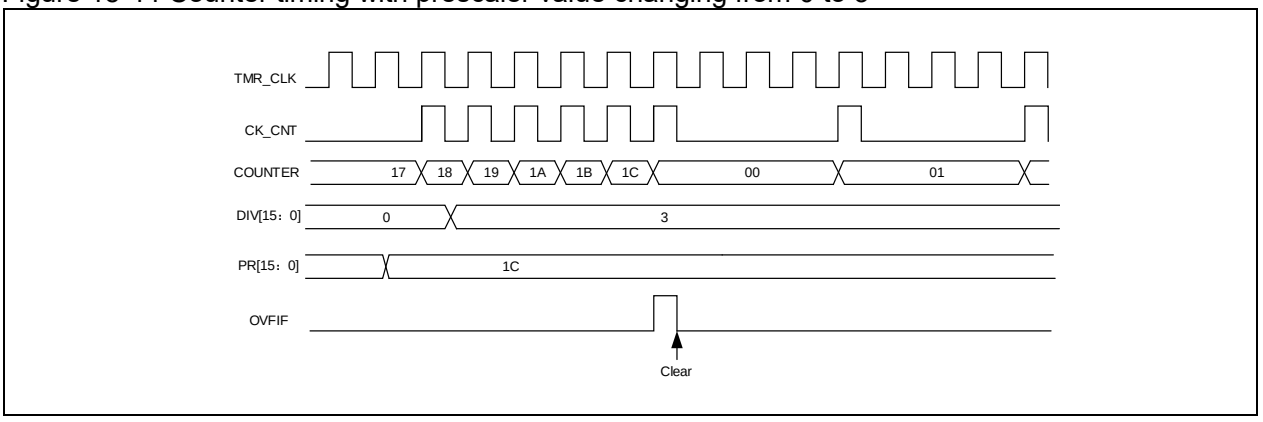


Table 15-7 TMRx internal trigger connection

Slave controller	IS0 (STIS=000)	IS1 (STIS = 001)	IS2 (STIS = 010)	IS3 (STIS = 011)
TMR9	TMR2	TMR3	TMR10_C1OUT	TMR11_C1OUT
TMR12	TMR4	TMR5	TMR13_C1OUT	TMR14_C1OUT

Note: If there is no corresponding timer in a device, the corresponding trigger signal ISx is not present.

15.3.3.2 Counting mode

TMR9 to TMR14 support multiple counting modes to meet diverse application scenarios. They consist of a 16-bit up, down, up/downcounter.

The TMRx_PR register issued to set the counting period. The value in the TMRx_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN=1 in the TMRx_CTRL1 register), the value in the TMRx_PR register is transferred to the shadow register only at an overflow event. The OVFEN and OVFS bits are used to configure the overflow event.

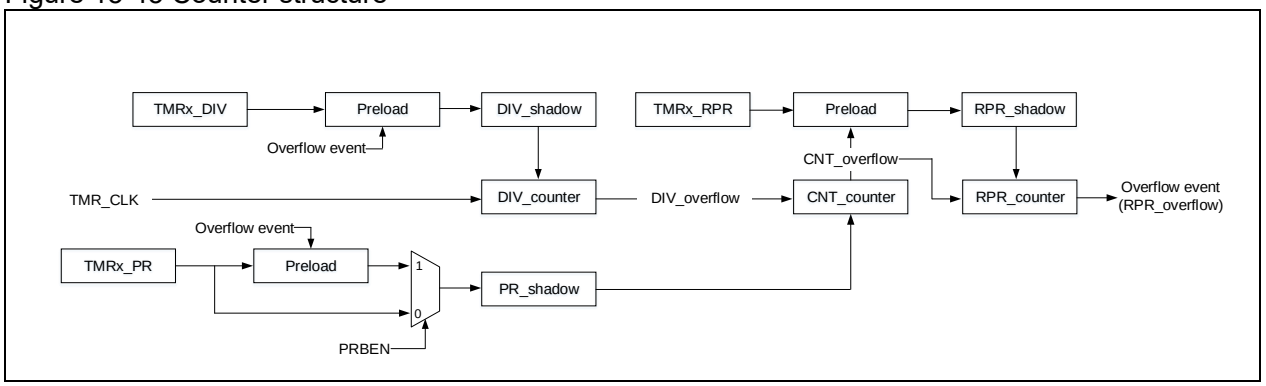
The TMRx_DIV register is used to configure the counting frequency. The counter counts once every count clock period (DIV[15:0]+1). The value in the TMRx_DIV register is updated to the shadow register at an overflow event.

Reading the TMRx_CNT register returns to the current counter value, and writing to the TMRx_CNT register updates the current counter value to the value being written.

An overflow event is generated by default. Set OVFEN=1 in the TMRx_CTRL1 to disable generation of overflow events. The OVFS bit in the TMRx_CTRL1 register is used to select overflow event source. By default, counter overflow/underflow, setting OVFSWTR bit and the reset signal generated by the slave timer controller in reset mode trigger the generation of an overflow event. When the OVFS bit is set, only counter overflow/underflow triggers an overflow event.

Setting TMREN=1 to enable the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR_EN is set 1 clock cycle after the TMREN is set.

Figure 15-45 Counter structure



Upcounting mode

Set TWCMSEL[1:0]=2'b00 and OWCDIR=1'b0 in the TMRx_CTRL1 register to enable upcounting mode. In this mode, the counter counts from 0 to the value programmed in the TMRx_PR register, then restarts from 0, and generates a counter overflow event, with the OVFIF bit being set to 1. If the overflow event is disabled, the counter is no longer reloaded with the preload value and period value at a counter overflow event; otherwise, the counter is updated with the preload value and period value on an overflow event.

Figure 15-46 Overflow event when PRBEN=0

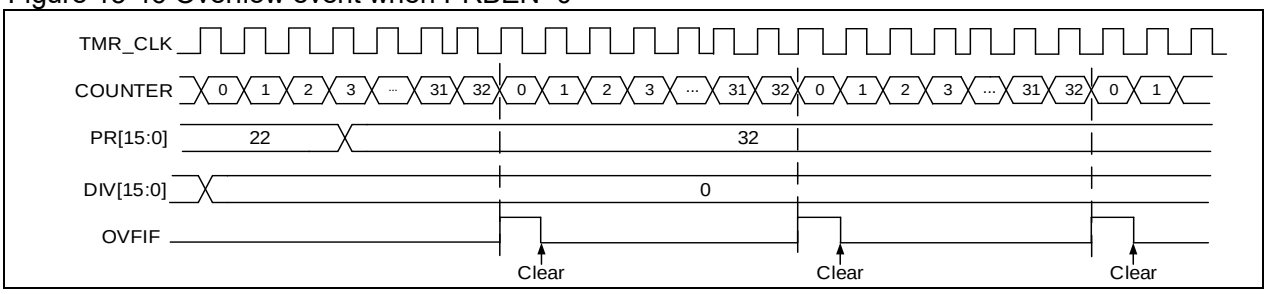
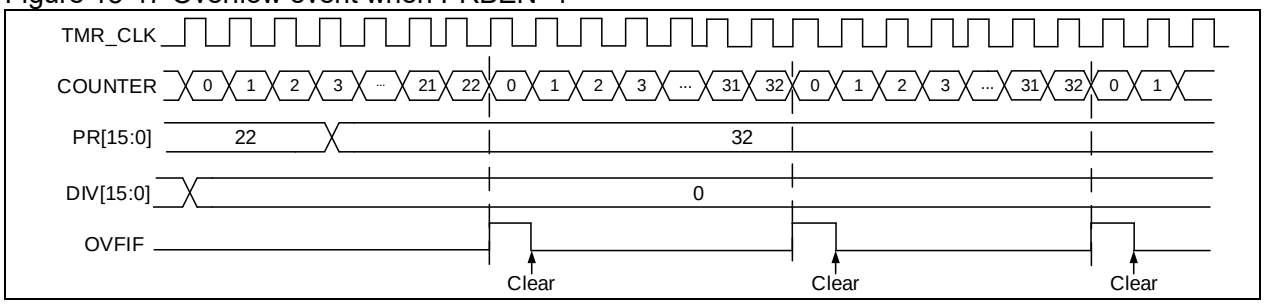


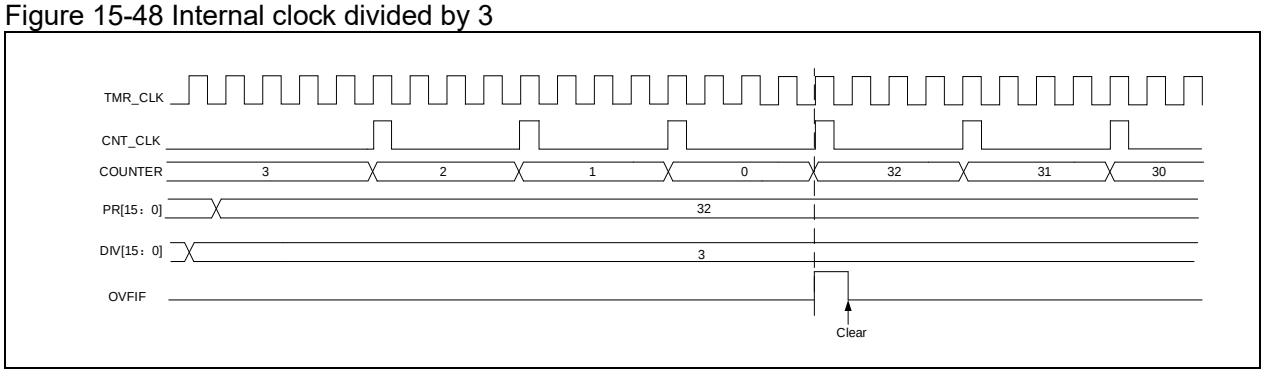
Figure 15-47 Overflow event when PRBEN=1



Downcounting mode

TWCMSEL[1:0]=2'b00 and OWCDIR=1'b1 in the TMRx_CTRL1register to enable downcounting mode. In this mode, the counter counts from the value programmed in the TMRx_PR register down to 0, and restarts from the value programmed, and generates a counter underflow event.

Figure 15-48 Internal clock divided by 3



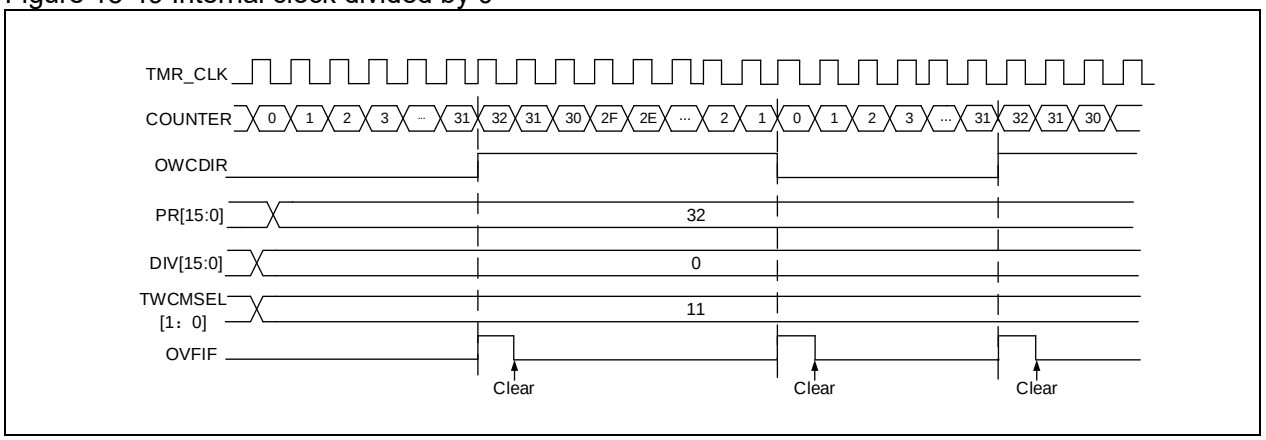
Up/down counting mode

Set TWCMSEL[1:0]≠2'b00 in the TMRx_CTRL1 register to enable up/down counting mode. In this mode, the counter counts up/down alternatively. When the counter counts from the value programmed in the TMRx_PR register down to 1, an underflow event is generated, and then restarts counting from 0; When the counter counts from 0 to the value of the TMRx_PR register -1, an overflow event is generated, and then restarts counting from the value of the TMRx_PR register. The OWCDIR bit indicates the current counting direction.

The TWCMSEL[1:0] bit in the TMRx_CTRL1 register is also used to select the CxIF flag setting method in up/down counting mode. In up/down counting mode 1 (TWCMSEL[1:0]=2'b01), CxIF flag can only be set when the counter counts down; in up/down counting mode 2 (TWCMSEL[1:0]=2'b10), CxIF flag can only be set when the counter counts up; in up/down counting mode 3 (TWCMSEL[1:0]=2'b11), CxIF flag can be set when the counter counts up/down.

Note: The OWCDIR is read-only in up/down counting mode.

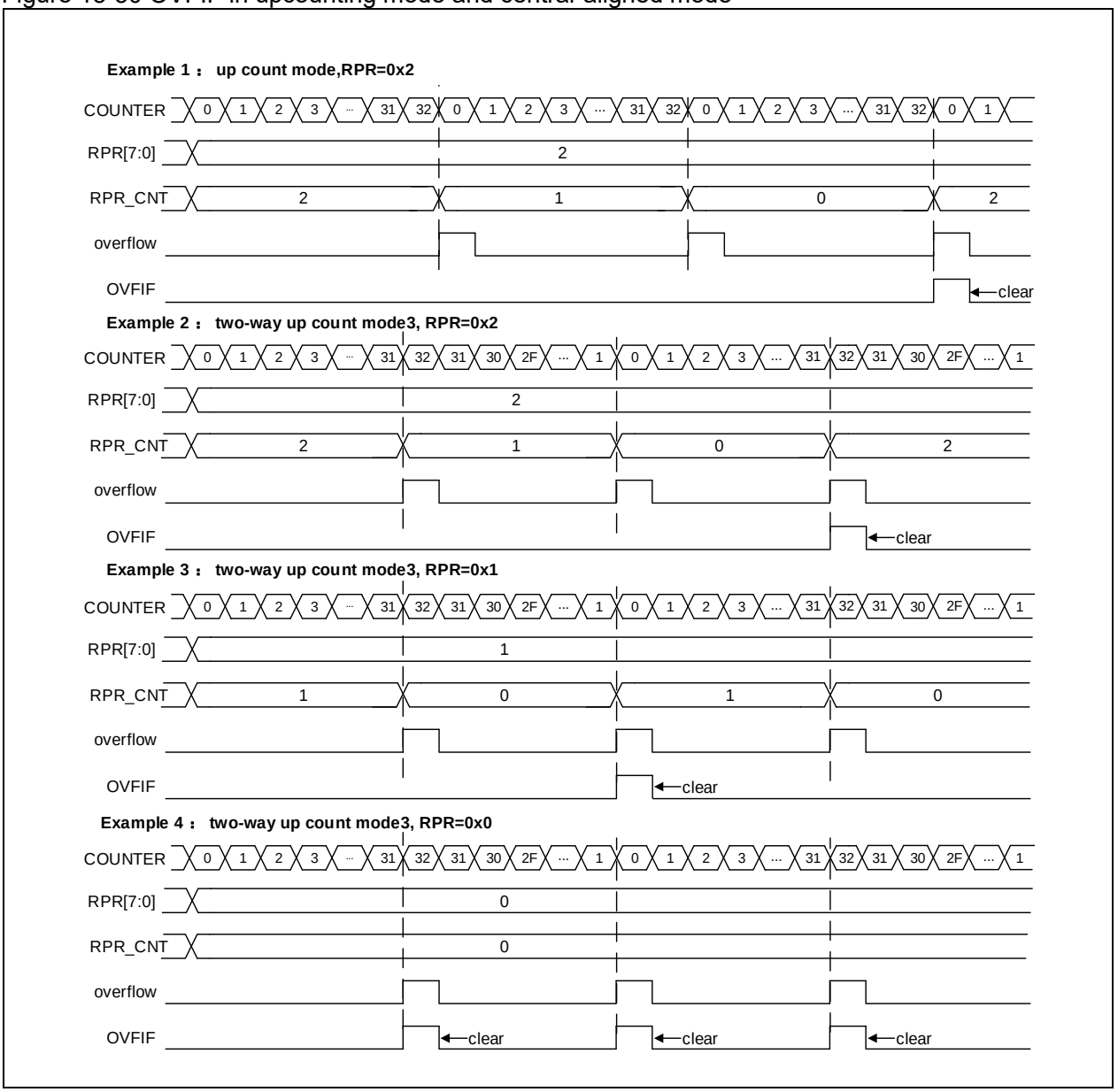
Figure 15-49 Internal clock divided by 0



Repetition counter mode:

The TMRx_RPR register is used to enable repetition counting mode. This mode is enabled when the repetition counter value is not equal to 0. In this mode, an overflow event is generated when a counter overflow occurs (RPR[7:0]+1). The repetition counter is decremented at each counter overflow. An overflow event is generated when the repetition counter reaches 0. The frequency of the overflow event generation can be adjusted by setting the repetition counter value.

Figure 15-50 OVFIF in upcounting mode and central-aligned mode



15.3.3.3 TMR input function

Each timer of TMR9 and TMR12 has two independent channels, while each of TMR10, TMR11, TMR13 and TMR14 has an independent channel. Each channel can be configured as input or output.

As input, each channel input signal is handled as follows:

- TMRx_CHx outputs the pre-processed CxIRAW.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit in the TMRx_CM1 register to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits in the TMRx_CCTRL register. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bit in the TMRx_CM1 register. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx (x≠y) is the CyIFPy signal that is from Y channel. The STCI comes from slave timer controller, and its source is selected by STIS bit. For single-channel TMR, it only supports CxIFPx as the source of CxIN.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit in the TMRx_CM1 register.

Figure 15-51 Input/output channel 1 main circuit

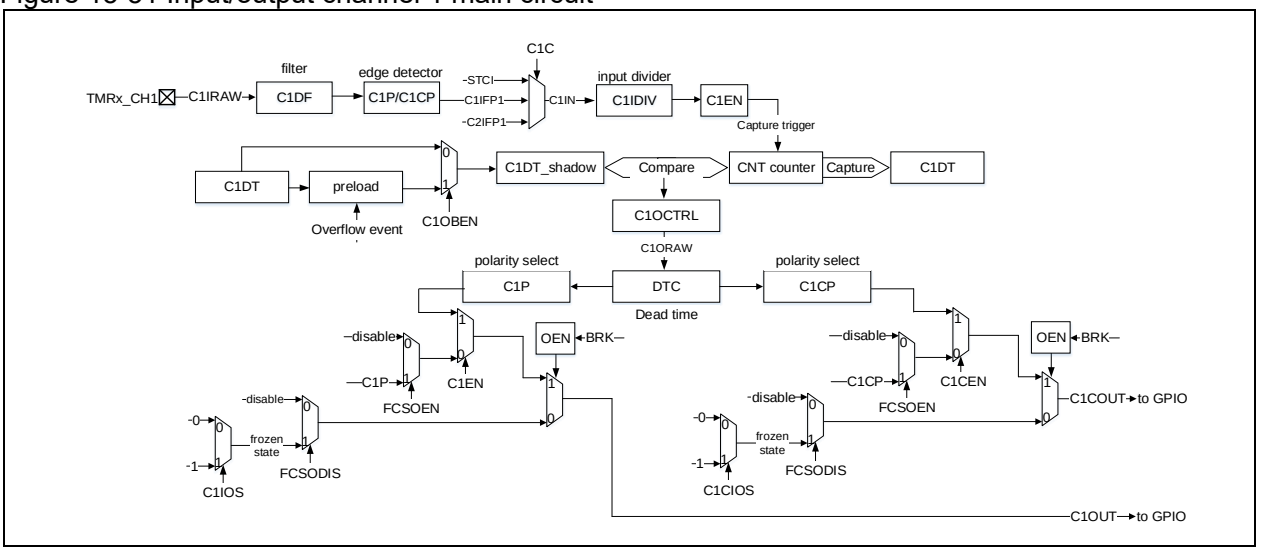
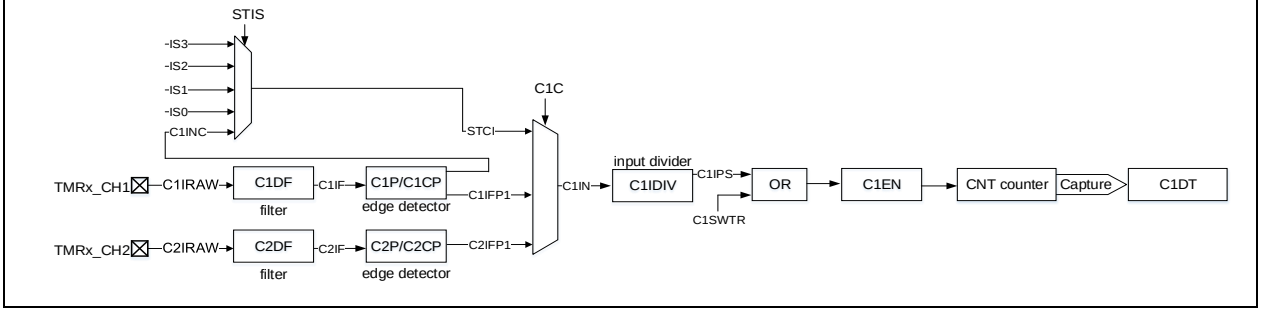


Figure 15-52 Channel 1 input stage



Input mode

In input mode, the TMRx_CxDT registers latch the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF in the TMRx_ISTS register) is set. An interrupt or a DMA request will be generated if the CxIEN or CxDEN bit in the TMRx_IDEN register is enabled. If the selected trigger signal is detected when the CxIF is set to 1, a capture overflow event occurs. The TMRx_CxDT register overwrites the recorded value with the current counter value, and the CxRF is set to 1 in the TMRx_ISTS register.

To capture the rising edge of C1IN input, following the configuration procedure mentioned below:

- Set C1C=2'b01 in the TMRx_CM1 register to select the C1IN as channel 1 input;
- Set the filter bandwidth of C1IN signal (CxDF[3: 0]);
- Set the active edge on the C1IN channel by writing C1P=2'b0 (rising edge) in the TMRx_CCTR Register;
- Program the capture frequency division of C1IN signal (C1DIV[1: 0]);
- Enable channel 1 input capture (C1EN=1);
- If needed, enable interrupt or DMA by setting C1IEN bit or C1DEN bit in the TMRx_IDEN register.

PWM input (TMR9/12 only)

The PWM input mode applies to channel 1 and channel 2. To enable this mode, map the C1IN and C2IN to the same TMRx_CHx, and configure the CxIFPx of channel 1/2 to trigger slave timer controller reset.

The PWM input mode can be used to measure the period and duty cycle of input signal. The period and duty cycle of channel 1 can be measured as follows:

- Set C1C=2'b01 in the TMRx_CM1 register to set C1IN as C1IFP1;
- Set C1P=1'b0 in the TMRx_CCTRL register to set C1IFP1 rising edge active;
- Set C2C=2'b10 in the TMRx_CM1 register to set C2IN as C1IFP2;
- Set C2P=1'b1 in the TMRx_CCTRL register to set C1IFP2 falling edge active;
- Set STIS=3'b101 in the TMRx_CCTRL register to set C1IFP1 as the slave timer trigger signal;
- Set SMSEL=3'b110 in the TMRx_STCTRL register to set the slave timer in reset mode;

- Set C1EN=1'b1 and C2EN=1'b1 in the TMRx_CCTRL register to enable channel 1 and channel 2 input capture.

In these configurations, the rising edge of channel 1 input signal triggers capture and saves captured values to the C1DT register, and channel 1 input signal rising edge resets the counter. The falling edge of channel 1 input signal triggers capture and saves captured values to the C2DT register. The period and duty of channel 1 input signal can be calculated through C1DT and C2DT respectively.

Figure 15-53 Example of PWM input mode configuration

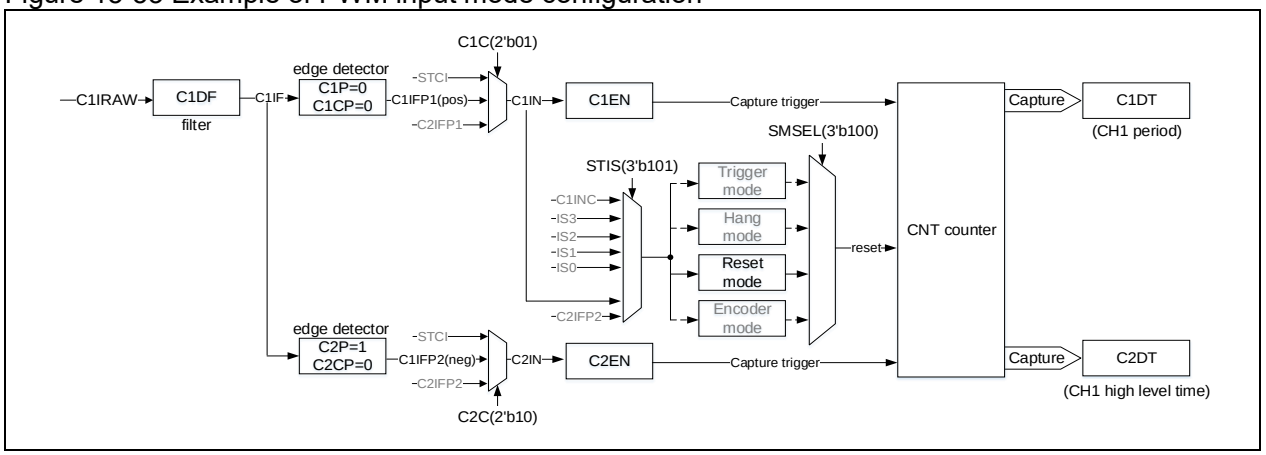
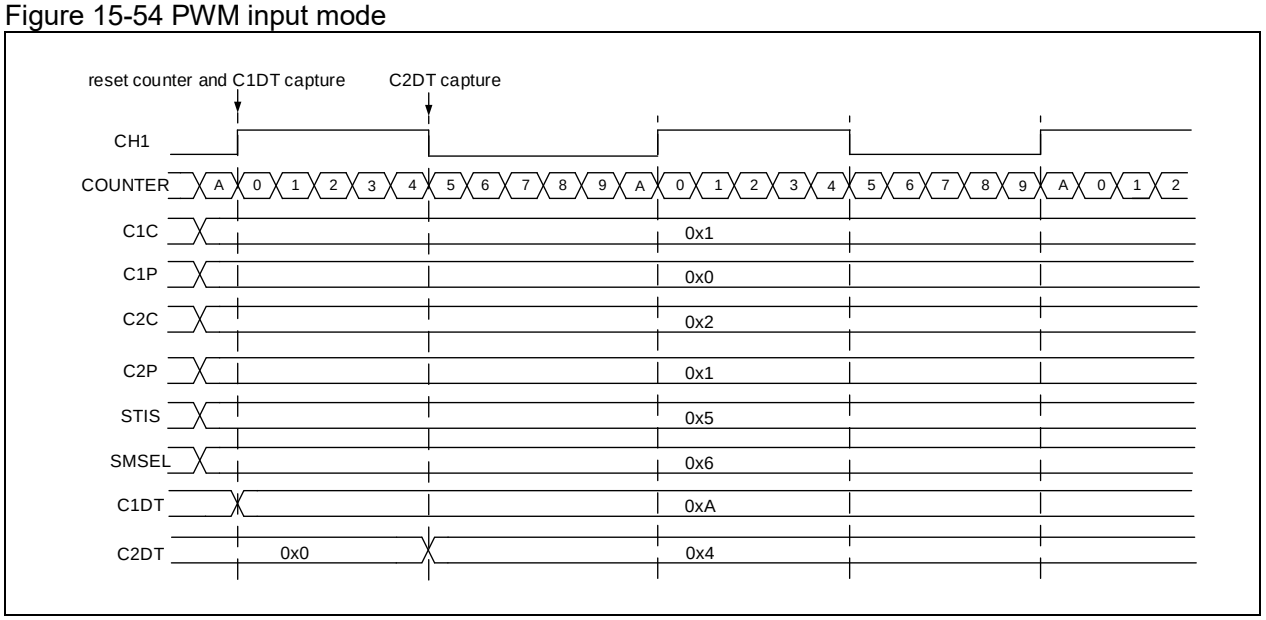


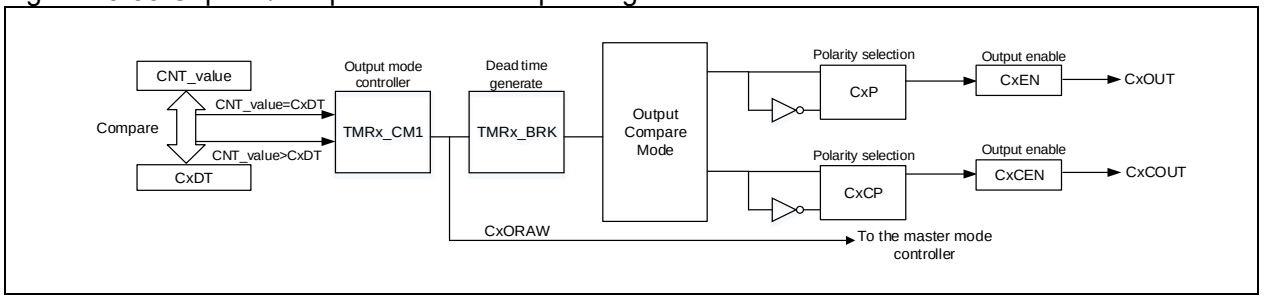
Figure 15-54 PWM input mode



15.3.3.4 TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal.

Figure 15-55 Capture/compare channel output stage



Output mode

Write CxC[1: 0]≠2'b00 to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the TMRx_CxDT register, and the intermediate signal CxORAW is generated according to the output mode selected by CxOCTRL[2: 0], which is sent

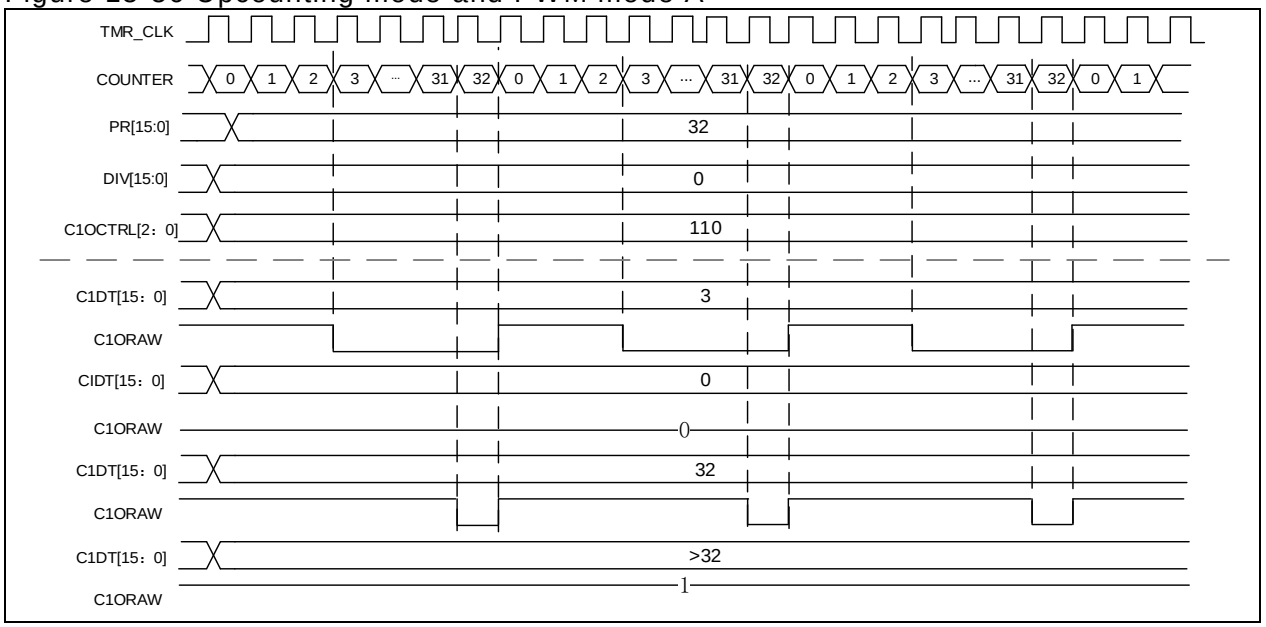
to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx_PR register, while the duty cycle by the TMRx_CxDT register.

Output compare modes include:

- **PWM mode A**: Set CxOCTRL=3'b110 to enable PWM mode A. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. Figure 15-56 gives an example of PWM mode A in upcounting mode, with PR=0x32, and CxDT with different values.

  To set PWM mode A, the following process is recommended:
  - Set the TMRx_PR register to set PWM period;
  - Set the TMRx_CxDT register to set PWM duty cycle;
  - Set CxOCTRL=3'b110 in TMRx_CM1/CM2 register and set output mode as PWM mode A;
  - Set the TMRx_DIV register to set the counting frequency;
  - Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 to set the count mode;
  - Set the CxP and CxCP bits in the TMRx_CCTRL register to set the output polarity;
  - Set the CxEN and CxCEN bits in the TMRx_CCTRL register to enable channel output;
  - Set the OEN bit in the TMRx_BRK register to enable TMRx output;
  - Set the corresponding GPIO of TMR output channel as the multiplexed mode;
  - Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.

Figure 15-56 Upcounting mode and PWM mode A

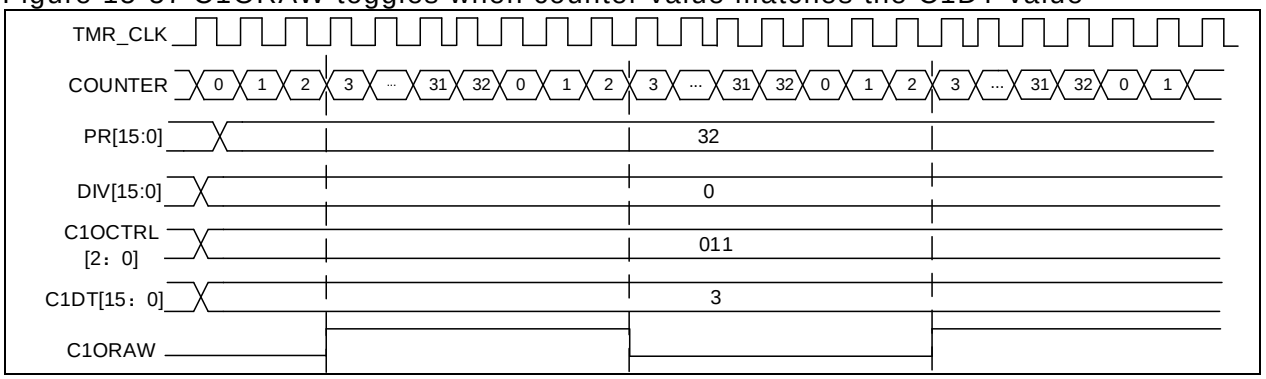


- **PWM mode B**: Set CxOCTRL=3'b111 to enable PWM mode B. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low.

- **Forced output mode:** Set CxOCTRL=3'b100/101 to enable forced output mode. In this case, the CxORAW is forced to be the programmed level, irrespective of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.

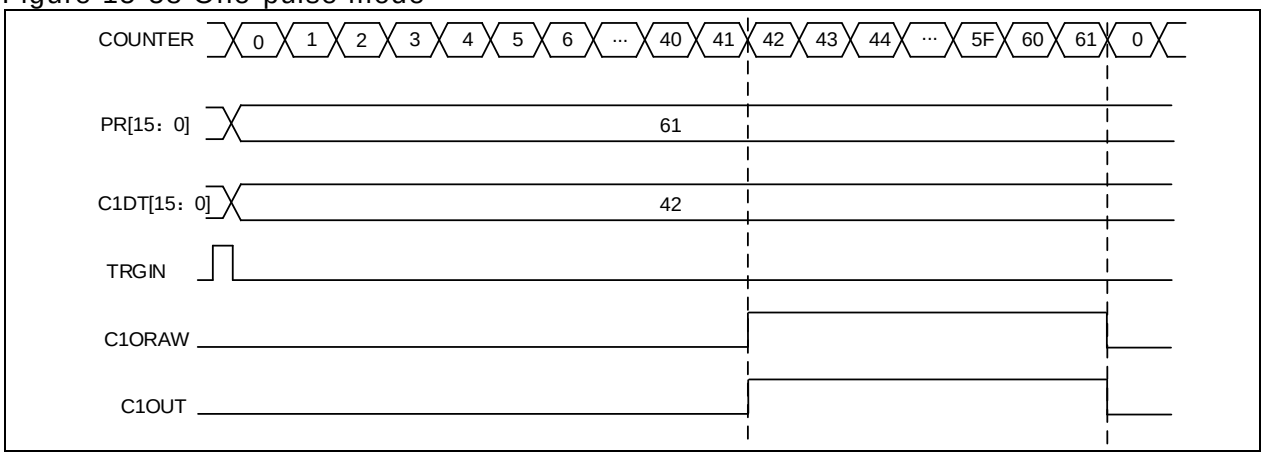
- **Output compare mode:** Set CxOCTRL=2'b001/010/011 to enable output compare mode. In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high (CxOCTRL=3'b001), low (CxOCTRL=3'b010) or toggling (CxOCTRL=3'b011). Figure 15-57 gives an example of output compare mode, with C1DT=0x3, and counter value is 0x3.

Figure 15-57 C1ORAW toggles when counter value matches the C1DT value



- **One-pulse mode (TMR9/12 only):** This is a particular case of PWM mode. Set OCMEN=1 in the TMRx_CTRL1 register to enable one-pulse mode. In this mode, the comparison match is performed in the current counting period. The TMREN bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When configured as in upcounting mode, the configuration must follow the rule: CVAL<CxDT≤PR; in downcounting mode, CVAL>CxDT is required. Figure 15-58 gives an example of PWM mode B in upcounting mode and one-pulse mode in which the counter counts for a single cycle and there is a single pulse output.

Figure 15-58 One-pulse mode



- **Fast output mode:** Set CxOIEN=1 in the TMRx_CM1 register to enable this mode. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx_CxDT register will determine the level of CxORAW in advance.

Master mode timer event output (TMR9/12 only)

When TMR is used as a master timer, one of the following source of signals can be selected as TRGOUT output to a slave mode timer. This is done by setting the PTOS bit in the TMRxCTRL2 register.

- PTOS=3’b000, TRGOUT output software overflow event (OVFSWTR bit in TMRx_SWEVT register) or reset event
- PTOS=3’b001, TRGOUT output counter enable
- PTOS=3’b010, TRGOUT output counter overflow event
- PTOS=3’b011, TRGOUT output capture and compare event
- PTOS=3’b100, TRGOUT output C1ORAW
- PTOS=3’b101, TRGOUT output C2ORAW

Dead-time insertion

TMR9 and TMR12 channel 1 include reverse channel output which is enabled by the CxCEN bit and its polarity is selected by CxCP. .

The dead-time is activated when switching to IDLEF state (OEN falling down to 0).

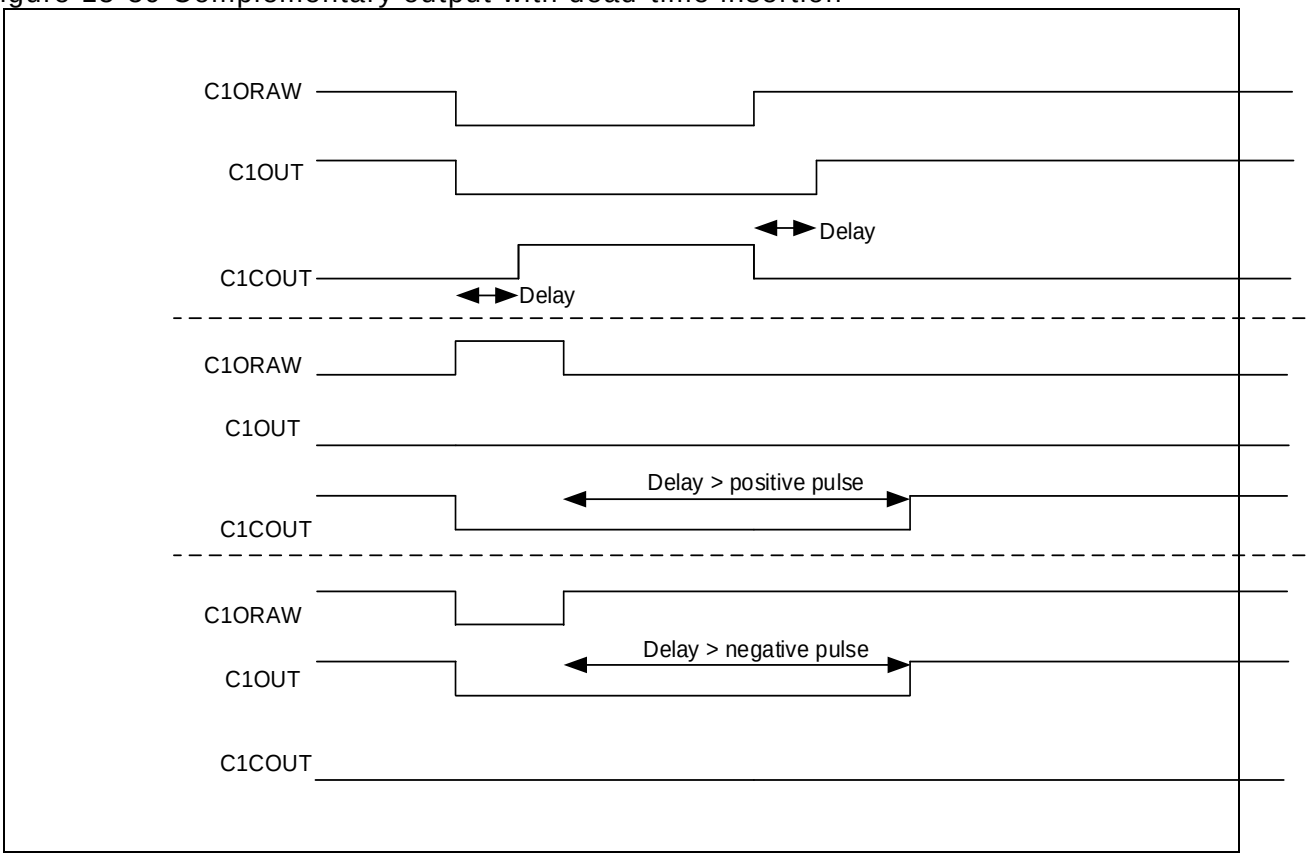
Setting both CxEN and CxCEN bits, and using DTC[7:0] bit to insert dead-time of different durations.

After the dead-time insertion, the rising edge of the CxOUT is delayed compared to the rising edge of the reference signal; the rising edge of the CxCOU is delayed compared to the falling edge of the reference signal.

If the delay is greater than the width of the active output, C1OUT and C1COUT will not generate corresponding pulses. Therefore, the dead-time should be less than the width of the active output.

The figure below gives an example of dead-time insertion when CxP=0, CxCP=0, OEN=1, CxEN=1 and CxCEN=1.

Figure 15-59 Complementary output with dead-time insertion



15.3.3.5 TMR break function

When the brake function is enabled (BRKEN=1 in the TMRx_BRK register), the CxOUT and CxCOUT are jointly controlled by OEN, FCSODIS, FCSOEN, CxIOS and CxCIOS. But, CxOUT and CxCOUT cannot be set both to active level at the same time. For details, refere to Table 15-11.

The brake source can be a break input pin, clock failure event. The polarity of break input is selected by BRKV bit.

When a brake event occurs, there are the following actions:

- The OEN bit is cleared asynchronously, and the channel output state is selected by setting the FCSODIS bit. This function works even if the MCU oscillator is off.
- Once OEN=0, the channel output level is defined by the CxIOS bit. If FCSODIS=0, the timer output is disabled; otherwise, the output enable remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state, that is, 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 active, then the dead-time generator is activated. The CxIOS and CxCIOS bits are used to program the level after dead-time. Even in this case, the CxIOS and CxCIOS cannot be driven to their active level at the same time.

Note: The dead-time duration is usually longer than usual (around 2 tmr_clk clock cycles) due to OEN synchronization logic.

— If FCSODIS=0, the timer releases the enable output; otherwise, it keeps the enable output; the enable output becomes high as soon as one of the CxEN and CxCEN bits becomes high.

- If the brake interrupt or DMA request is enabled, the brake statue flag is set, and a brake interrupt or DMA request can be generated.
- If AOEN=1, the OEN bit is automatically set again at the next overflow event.

Note: When the brake input is active, the OEN cannot be set, nor the status flag, BRKIF can be cleared.

Figure 15-60 TMR output control

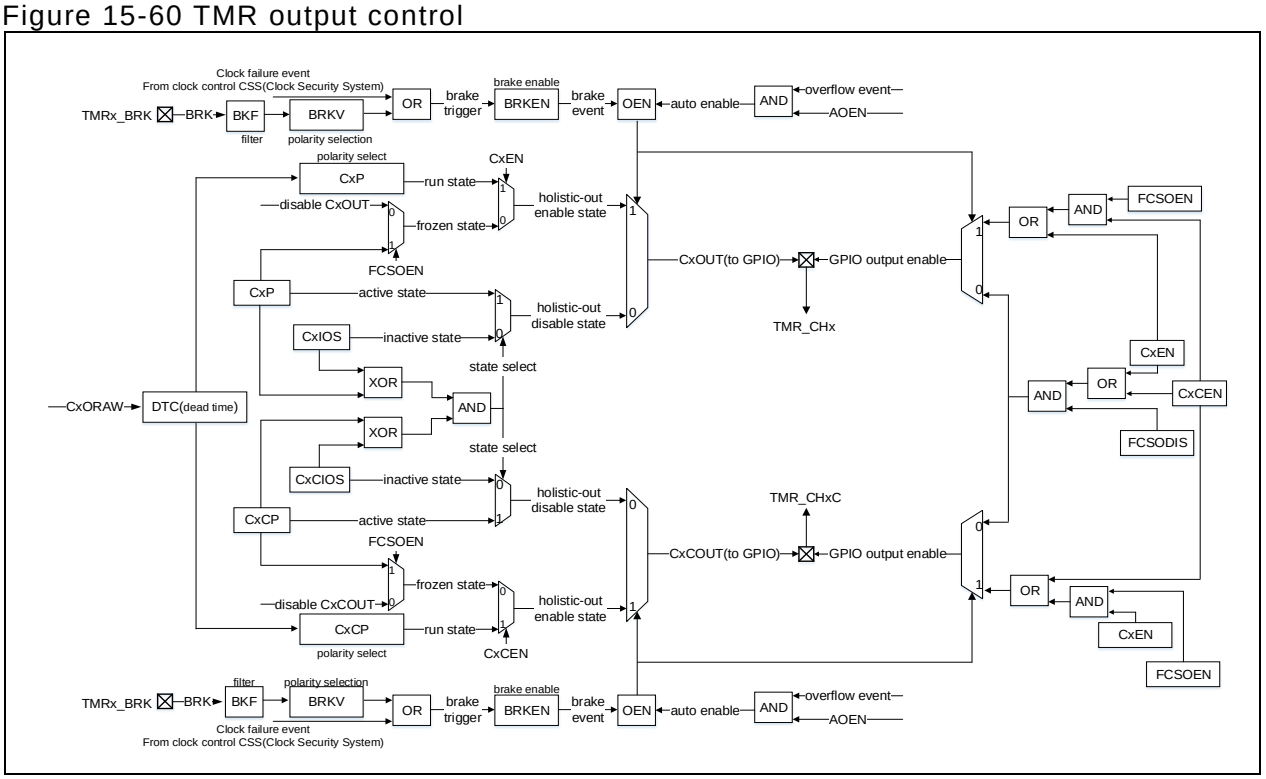
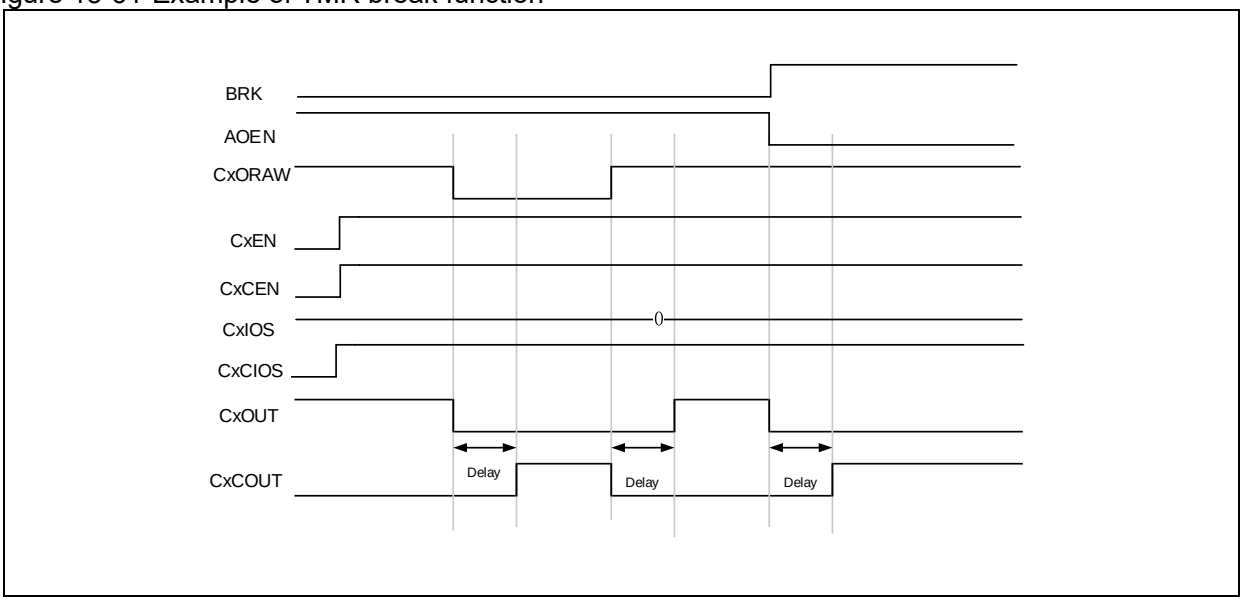


Figure 15-61 Example of TMR break function



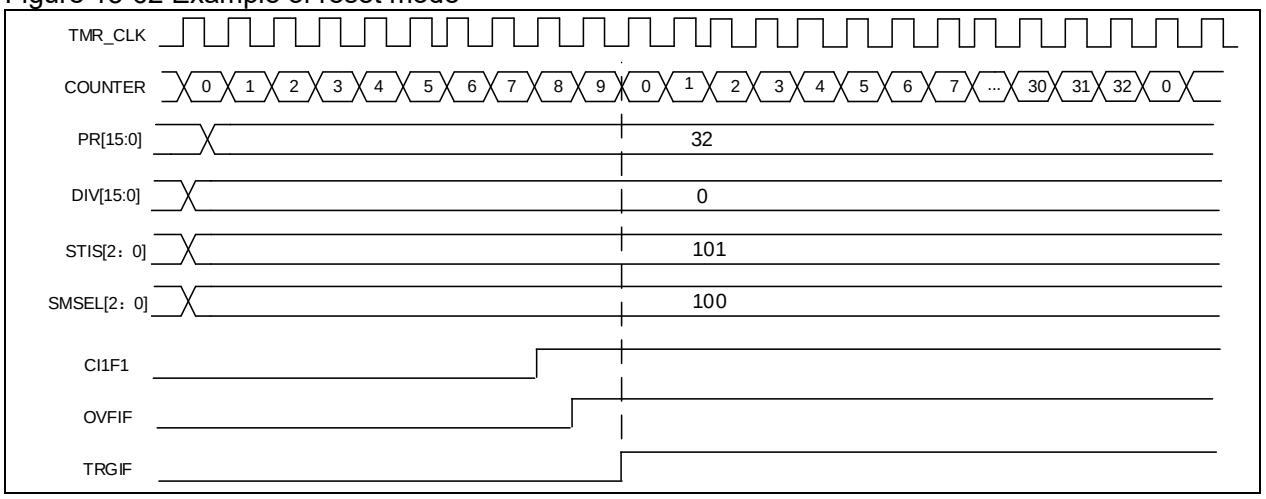
15.3.3.6 TMR synchronization

TMR9 and TMR12 are linked together internally for timer synchronization. Slave timer is selected by setting the SMSEL[2: 0] bit in the TMRx_STCTRL register.

Slave mode: Reset mode

The counter and its prescaler can be reset by a selected trigger signal. An overflow event is generated when OVFS=0.

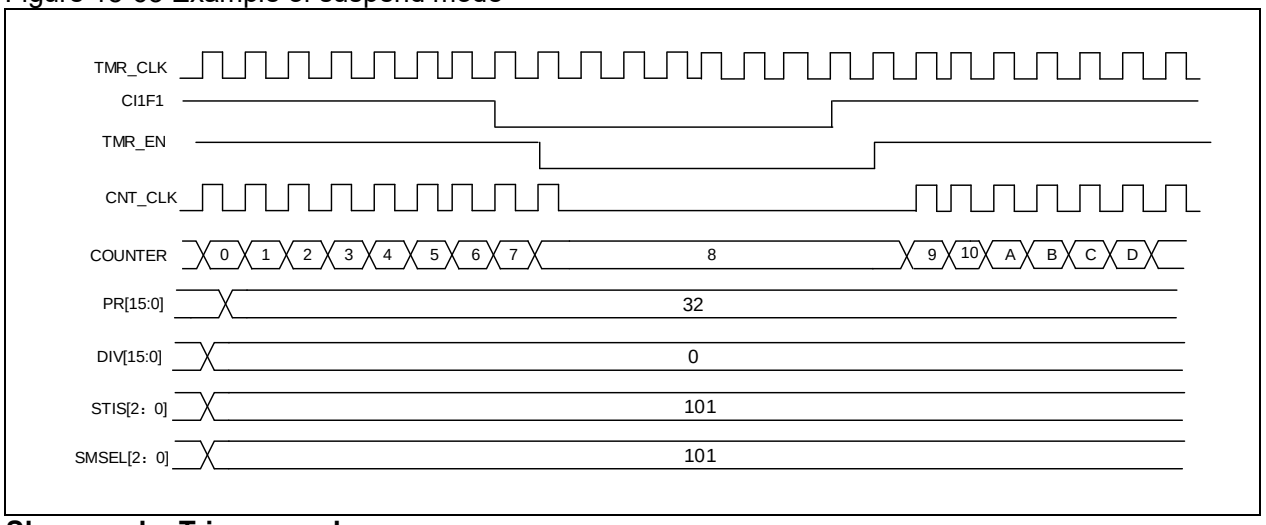
Figure 15-62 Example of reset mode



Slave mode: Suspend mode

In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.

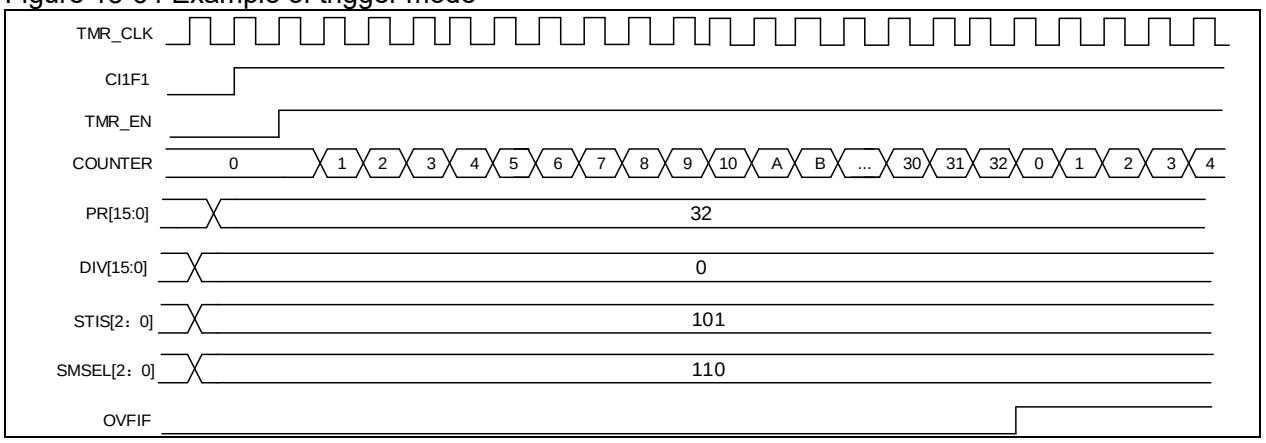
Figure 15-63 Example of suspend mode



Slave mode: Trigger mode

The counter can start counting on the rising edge of a selected trigger input (TMR_EN=1).

Figure 15-64 Example of trigger mode



15.3.3.7 TMR DMA

TMR has the following events for DMA transfer request: overflow event DMA request, trigger event DMA rquest, Hall sensor DMA request and chanel event DMA request. It is possible to enable DMA request by setting the TMRx_IDEN register. Once enabled, upon an event generated, a DMA request is triggered and ouput to the DMA peripheral.

TMR DMA Burst feature

TMR also supports TMR DMA Burst feature. Thanks to this feature, DMA can be triggered to rewrite multiple TMR consecutive registers by enabling a certain DMA request through the TMRx_IDEN register.

As an overflow event to trigger TMR DMA Burst as an example:

- Enable overflow event to trigger DMA request using the OVFDEN bit in the TMRx_IDEN reigster
- Configure Burst transfer times using the DTB bit in the TMRx_DMACTRL register
- Configure the start address of Burst transfer using the ADDR bit in the TMRx_DMACTRL reigster
- Enable counter

After the above-mentioned configurations, here is the whole process of TMR DMA Burst transfer:

Upon an overflow event, the TMR will send a DMA request to the DMA peripheral. Then the DMA writes the data into the TMRx_DMADT register as requested. Inside TMR, the DMADT bit data is written into the start address register of Burst thransfer and an ACK single is sent to TMR. After receiving this ACK, the TMR clears the current DMA request; when the Burst transfer is detected by the TMR not to complete fully, a new overflow event for DMA request is released to the DMA so tthat the DMA writes the requested dta into the TMRx_DMADT again. In this case, inside tmr, the DMADT register data is written into the Burst transfer start address + 0x4 address register, and a new ACK is sent to TMR, and so on, and so on, until the last operation of Burst transfer. When a full Burst transfer is complete, the REQ for overflow event DMA request would not be set any more until the next overflow event.

Note: when using TMR DMA Burst feature, an empty register is prohibited and the TMRx_DMACTRL as well as TMRx_DMADT registers should not be available during the period from the start address to the end address.

15.3.3.8Debug mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter stops counting by setting the TMRx_PAUSE in the DEBUG module..

15.3.4 TMR9 and TMR12 registers

These peripheral registers must be accessed by words (32 bits).

Table 15-8 TMR9 and TMR12 register map and reset value

Register	Offset	Reset value
TMRx_CTRL1	0x00	0x0000 0000
TMRx_CTRL2	0x04	0x0000 0000
TMRx_STCTRL	0x08	0x0000 0000
TMRx_IDEN	0x0C	0x0000 0000
TMRx_ISTS	0x10	0x0000 0000
TMRx_SWEVT	0x14	0x0000 0000
TMRx_CM1	0x18	0x0000 0000
TMRx_CCTRL	0x20	0x0000 0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000 0000
TMRx_PR	0x2C	0x0000 FFFF
TMRx_RPR	0x30	0x0000 0000
TMRx_C1DT	0x34	0x0000 0000
TMRx_C2DT	0x38	0x0000 0000

TMRx_BRK	0x44	0x0000 0000
TMRx_DMACTRL	0x48	0x0000 0000
TMRx_DMADT	0x4C	0x0000 0000

15.3.4.1 TMR9 and TMR12 control register 1 (TMRx_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x00 0000	resd	Kept at its default value
Bit 9: 8	CLKDIV	0x0	rw	Clock divider This field is used to define the division ratio between digital filter sampling frequency (f_{DTS}) and timer clock frequency (f_{CK_INT}). 00: No division, $f_{DTS}=f_{CK\_INT}$ 01: Divided by 2, $f_{DTS}=f_{CK\_INT}/2$ 10: Divided by 4, $f_{DTS}=f_{CK\_INT}/4$ 11: Reserved
Bit 7	PRBEN	0x0	rw	Period buffer enable 0: Period buffer is disabled 1: Period buffer is enabled
Bit 6: 5	TWCMSEL	0x0	rw	Two-way counting mode selection 00: One-way counting mode, depending on the OWCDIR bit 01: Two-way up/down counting mode1, count up and down alternately, the CxIF bit is set only when the counter counts down 10: Two-way up/down counting mode2, count up and down alternately, the CxIFbit is set only when the counter counts up 11: Two-way up/down counting mode3, count up and down alternately, the CxIF bit is set when the counter counts up / down
Bit 4	OWCDIR	0x0	rw	One-way count direction 0：upcount 1：downcount
Bit 3	OCMEN	0x0	rw	One cycle mode enable This bit is use to select whether to stop counting at an update event 0: The counter does not stop at an update event 1: The counter stops at an update event
Bit 2	OVFS	0x0	rw	Overflow event source This bit is used to select overflow event or DMA request sources. 0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller 1: Only counter overflow generates an overflow event
Bit 1	OVFEN	0x0	rw	Overflow event enable 0: Enabled 1: Disabled
Bit 0	TMREN	0x0	rw	TMR enable 0: Enabled 1: Disabled

15.3.4.2 TMR9 and TMR12 control register 2 (TMRx_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x0 0000	resd	Kept at its default value.
Bit 11	C2CIOS	0x0	rw	hannel 2 complementary idle output state
Bit 10	C2IOS	0x0	rw	Channel 2 idle output state
Bit 9	C1CIOS	0x0	rw	Channel 1 complementary idle output state When OEN = 0 is asserted, after dead-time generation: 0：C1COUT=0 1：C1COUT=1

Bit 8	C1IOS	0x0	rw	Channel 1 idle output state When OEN = 0 is asserted, after dead-time generation: 0：C1OUT=0。 1：C1OUT=1。
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6: 4	PTOS	0x0	rw	Master TMR output selection This field is used to select the TMRx signal sent to the slave timer. 000: Reset 001: Enable 010: Update 011: Compare pulse 100: C1ORAW signal 101: C2ORAW signal 110: C3ORAW signal 111: C4ORAW signal
Bit 3	DRS	0x0	rw	DMA request source 0: Capture/compare event 1: Overflow event
Bit 2	CCFS	0x0	rw	Channel control bit flash selection This bit only acts on channels that have complementary output. If the channel control bits are buffered: 0: Control bits are updated by setting the HALLSWTR bit 1: Control bits are updated by setting the HALLSWTR bit or a rising edge on TRGIN.
Bit 1	Reserved	0x0	resd	Kept at default value.
Bit 0	CBCTRL	0x0	rw	Channel buffer control This bit acts on channels that have complementary output. 0: CxEN, CxCEN and CxOCTRL bits are not buffered. 1: CxEN, CxCEN and CxOCTRL bits are buffered. Note: When CBCTRL=”1”, CxEN, CxCEN and CxOCTRL bits are updated only upon a HALL event.

15.3.4.3 TMR9 and TMR12 slave timer control register (TMRx_STCTRL)

Bit	Name	Reset value	Type	Description
Bit 31:8	Reserved	0x00 0000	resd	Kept at its default value
Bit 7	STS	0x0	rw	Subordinate TMR synchronization If enabled, master and slave timer can be synchronized. 0: Disabled 1: Enabled
Bit 6: 4	STIS	0x0	rw	Subordinate TMR input selection This field is used to select the subordinate TMR input. 000: Internal selection 0 (IS0) 001: Internal selection 1 (IS1) 010: Internal selection 2 (IS2) 011: Internal selection 3 (IS3) 100: C1IRAW input detector (C1INC) 101: Filtered input 1 (C1IFP1) 110: Filtered input 2 (C1IFP2) 111: Reserved Please refer to Table 15-7 for more information on ISx for each timer.
Bit 3	Reserved	0x0	resd	Kept at its default value

Bit 2: 0	SMSEL	0x0	rw	Subordinate TMR mode selection 000: Slave mode is disabled 001: Reserved 010: Reserved 011: Reserved 100: Reset mode — Rising edge of the TRGIN input reinitializes the counter 101: Suspend mode — The counter starts counting when the TRGIN is high 110: Trigger mode — A trigger event is generated at the rising edge of the TRGIN input 111: External clock mode A — Rising edge of the TRGIN input clocks the counter

15.3.4.4 TMR9 and TMR12 DMA/interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31:15	Reserved	0x0 0000	resd	Kept at its default value.
Bit 14	TDEN	0x0	rw	Trigger DMA request enable 0: Disabled 1: Enabled
Bit 13	HALLDE	0x0	rw	HALL DMA request enable 0: Disabled 1: Enabled
Bit 12: 11	Reserved	0x00	resd	Kept at default value.
Bit 10	C2DEN	0x0	rw	Channel 2 DMA request enable 0: Disabled 1: Enabled
Bit 9	C1DEN	0x0	rw	Channel 1 DMA request enable 0: Disabled 1: Enabled
Bit 8	OVFDEN	0x0	rw	Overflow event DMA request enable 0: Disabled 1: Enabled
Bit 7	BRKIE	0x0	rw	Brake interrupt enable 0: Disabled 1: Enabled
Bit 6	TIEN	0x0	rw	Trigger interrupt enable 0: Disabled 1: Enabled
Bit 5	HALLIEN	0x0	rw	HALL interrupt enable 0: Disabled 1: Enabled
Bit 4:3	Reserved	0x00	resd	Kept at its default value.
Bit 2	C2IEN	0x0	rw	Channel 2 interrupt enable 0: Disabled 1: Enabled
Bit 1	C1IEN	0x0	rw	Channel 1 interrupt enable 0: Disabled 1: Enabled
Bit 0	OVFIEN	0x0	rw	Overflow interrupt enable 0: Disabled 1: Enabled

15.3.4.5 TMR9 and TMR12 interrupt status register (TMRx_ISTS)

Bit	Name	Reset value	Type	Description
Bit 31: 11	Reserved	0x00 0000	resd	Kept at its default value.
Bit 10	C2RF	0x0	rw0c	Channel 2 recapture flag Please refer to C1RF description.
Bit 9	C1RF	0x0	rw0c	Channel 1 recapture flag This bit indicates whether a recapture is detected when C1IF=1. This bit is set by hardware, and cleared by writing “0”. 0: No capture is detected 1: Capture is detected.
Bit 8	Reserved	0x0	resd	Kept at its default value.
Bit 7	BRKIF	0x0	rw0c	Brake interrupt flag This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing “0” 0: Inactive level 1: Active level
Bit 6	TRGIF	0x0	rw0c	Trigger interrupt flag This bit is set by hardware on a trigger event. It is cleared by writing “0”. 0: No trigger event occurs 1: Trigger event is generated. Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode.
Bit 5	HALLIF	0x0	rw0c	HALL interrupt flag This bit is set by hardware on HALL event. It is cleared by writing “0”. 0: No Hall event occurs. 1: Hall event is detected. HALL even: CxEN, CxCEN and CxOCTRL are updated.
Bit 4:3	Reserved	0x0	resd	Kept at its default value.
Bit 2	C2IF	0x0	rw0c	Channel 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF	0x0	rw0c	Channel 1 interrupt flag If the channel 1 is configured as input mode: This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT 0: No capture event occurs 1: Capture event is generated If the channel 1 is configured as output mode: This bit is set by hardware on a compare event. It is cleared by software. 0: No compare event occurs 1: Compare event is generated
Bit 0	OVFIF	0x0	rw0c	Overflow interrupt flag This bit is set by hardware on an overflow event. It is cleared by software. 0: No overflow event occurs 1: Overflow event is generated. If OVFEN=0 and OVFS=0 in the TMRx_CTRL1 register: − An overflow event is generated when OVFSWTR= 1 in the TMRx_SWEVT register; − An overflow event is generated when the counter CVAL is reinitialized by a trigger event.

15.3.4.6 TMR9 and TMR12 software event register (TMRx_SWEVT)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	BRKSWTR	0x0	wo	Brake event triggered by software This bit is set by software to generate a brake event. 0: No effect 1: Generate a brake event.
Bit 6	TRGSWTR	0x0	rw	Trigger event triggered by software This bit is set by software to generate a trigger event. 0: No effect 1: Generate a trigger event.
Bit 5	HALLSWTR	0x0	wo	HALL event triggered by software This bit is set by software to generate a HALL event. 0: No effect 1: Generate a HALL event. Note: This bit acts only on channels that have complementary output.
Bit 4:3	Reserved	0x0	resd	Kept at its default value.
Bit 2	C2SWTR	0x0	wo	Channel 2 event triggered by software Please refer to C1SWTR description
Bit 1	C1SWTR	0x0	wo	Channel 1 event triggered by software This bit is set by software to generate a channel 1 event. 0: No effect 1: Generate a channel 1 event.
Bit 0	OVFSWTR	0x0	wo	Overflow event triggered by software This bit is set by software to generate an overflow event. 0: No effect 1: Generate an overflow event.

15.3.4.7 TMR9 and TMR12 channel mode register1 (TMRx_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:15	Reserved	0x0 0000	resd	Kept at its default value.
Bit 14: 12	C2OCTRL	0x0	rw	Channel 2 output control
Bit 11	C2OBEN	0x0	rw	Channel 2 output buffer enable
Bit 10	C2OIEN	0x0	rw	Channel 2 output enable immediately
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS register.
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6: 4	C1OCTRL	0x0	rw	Channel 1 output control This field defines the behavior of the original signal C1ORAW. 000: Disconnected. C1ORAW is disconnected from C1OUT; 001: C1ORAW is high when TMRx_CVAL=TMRx_C1DT

				010: C1ORAW is low when TMRx_CVAL=TMRx_C1DT 011: Switch C1ORAW level when TMRx_CVAL=TMRx_C1DT 100: C1ORAW is forced low 101: C1ORAW is forced high. 110: PWM mode A – OWCDIR=0, C1ORAW is high once TMRx_C1DT>TMRx_CVAL, else low; – OWCDIR=1, C1ORAW is low once TMRx_ C1DT <TMRx_CVAL, else high; 111: PWM mode B – OWCDIR=0, C1ORAW is low once TMRx_ C1DT >TMRx_CVAL, else high; – OWCDIR=1, C1ORAW is high once TMRx_ C1DT <TMRx_CVAL, else low. *Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.*
Bit 3	C1OBEN	0x0	rw	Channel 1 output buffer enable 0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately. 1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.
Bit 2	C1OIEN	0x0	rw	Channel 1 output enable immediately In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event. 0: Need to compare the CVAL with C1DT before generating an output 1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 12	C2DF	0x0	rw	Channel 2 digital filter
Bit 11: 10	C2IDIV	0x0	rw	Channel 2 input divider
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7: 4	C1DF	0x0	rw	Channel 1 digital filter This field defines the digital filter of the channel 1. "N" refers to the number of filtering, meaning that 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: $f_{SMPLING}$=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 3: 2	C1IDIV	0x0	rw	Channel 1 input divider This field defines Channel 1 input divider. 00: No divider. An input capture is generated at each active edge. 01: An input compare is generated every 2 active edges 10: An input compare is generated every 4 active edges 11: An input compare is generated every 8 active edges Note: the divider is reset once C1EN='0'
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

15.3.4.8 TMR9 and TMR12 channel control register (TMRx_CCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	C2CP	0x0	rw	Channel 2 complementary polarity Please refer to C1CP description.
Bit 6	C2CEN	0x0	rw	Channel 2 complementary enable Please refer to C1CEN description.
Bit 5	C2P	0x0	rw	Channel 2 polarity Please refer to C1P description.
Bit 4	C2EN	0x0	rw	Channel 2 enable Please refer to C1EN description.
Bit 3	C1CP	0x0	rw	Channel 1 complementary polarity 0: C1COUT is active high. 1: C1COUT is active low.
Bit 2	C1CEN	0x0	rw	Channel 1 complementary enable 0: Output is disabled. 1: Output is enabled.
Bit 1	C1P	0x0	rw	Channel 1 polarity When the channel 1 is configured as output mode: 0: C1OUT is active high 1: C1OUT is active low When the channel 1 is configured as input mode: 00: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted. 01: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. 10: Reserved 11: C1IN active edge is on rising/falling edge. When used as external trigger, C1IN is not inverted.
Bit0	C1EN	0x0	rw	Channel 1 enable 0: Input or output is disabled 1: Input or output is enabled

Table 15-9 Complementary output channel OCxand OCxN control bits with break feature

Control bit					Output state (1)	
OEN bit	FCSODIS bit	FCSOEN bit	CxEN bit	CxCEN bit	CxOUT output state	CxCOUT output state
1	X	0	0	0	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	0	1	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		0	1	0	CxORAW+ polarity CxOUT= CxORAW xor CxP, Cx_EN=1	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	1	1	CxORAW+polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
		1	0	0	Output disabled (no driven by the timer) CxOUT=CxP, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=CxCP, CxCEN=0
		1	0	1	Off-state (Output enabled with inactive level) CxOUT=CxP, Cx_EN=1	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		1	1	0	CxORAW + polarity, CxOUT= CxORAW xor CxP, Cx_EN=1	Off-state (Output enabled with inactive level) CxCOUT=CxCP, CxCEN=1
		1	1	1	CxORAW+ polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
0	0	X	0	0	Output disabled (no driven by the timer)	
	0		0	1	Asynchronously: CxOUT=CxP, Cx_EN=0, CxCOUT=CxCP, CxCEN=0;	
	0		1	0	If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	
	0		1	1		
	1		0	0	Off-state (output enabled and with invalid level) Asynchronously: CxOUT =CxP, Cx_EN=1, CxCOUT=CxCP, CxCEN=1; If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	

Note: If two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

Note: The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.

15.3.4.9 TMR9 and TMR12 counter value (TMRx_CVAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	CVAL	0x0000	rw	Counter value

15.3.4.10 TMR9 and TMR12 division value (TMRx_DIV)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DIV	0x0000	rw	Divider value The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK}$ /(DIV[15: 0]+1). DIV contains the value written at an overflow event.

15.3.4.11 TMR9 and TMR12 period register (TMRx_PR)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	PR	0xFFFF	rw	Period value This defines the period value of the TMRx counter. The timer stops working when the period value is 0.

15.3.4.12 TMR9 and TMR12 repetition period register (TMRx_RPR)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at default value.
Bit 7: 0	RPR	0x00	rw	Repetition of period value This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0.

15.3.4.13 TMR9 and TMR12 channel 1 data register (TMRx_C1DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	C1DT	0x0000	rw	Channel 1 data register When the channel 1 is configured as input mode: The C1DT is the CVAL value stored by the last channel 1 input event (C1IN). When the channel 1 is configured as output mode: C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured.

15.3.4.14 TMR9 and TMR12 channel 2 data register (TMRx_C2DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	C2DT	0x0000	resd	Kept at its default value.
Bit 15: 0	C2DT	0x0000	rw	Channel 2 data register When the channel 2 is configured as input mode: The C2DT is the CVAL value stored by the last channel 2 input event (C2IN). When the channel 2 is configured as output mode: C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured.

15.3.4.15 TMR9 and TMR12 break register (TMRx_BRK)

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x000	resd	Kept at its default value.
Bit 19: 16	BRKF	0x0	rw	Brake input filter This field is used to set the filter for brake input. “N” refers to the number of filtering, meaning that 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: $f_{SMPLING}=f_{DTS}/4$, 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 15	OEN	0x0	rw	Output enable This bit acts on the channels as output. It is used to enable CxOUT and CxCOUT outputs. 0: Disabled 1: Enabled
Bit 14	AOEN	0x0	rw	Automatic output enable OEN is set automatically at an overflow event. 0: Disabled 1: Enabled
Bit 13	BRKV	0x0	rw	Brake input validity This bit is used to select the active level of a brake input. 0: Brake input is active low. 1: Brake input is active high.
Bit 12	BRKEN	0x0	rw	Brake enable This bit is used to enable brake input. 0: Brake input is disabled. 1: Brake input is enabled.
Bit 11	FCSOEN	0x0	rw	Frozen channel status when holistic output enable This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=1. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output inactive level.
Bit 10	FCSODIS	0x0	rw	Frozen channel status when holistic output disable

				This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=0. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output idle level.
Bit 9: 8	WPC	0x0	rw	Write protection configuration This field is used to enable write protection. 00: Write protection is OFF. 01: Write protection level 3, and the following bits are write protected: TMRx_BRK: BRKF,DTC, BRKEN, BRKV and AOEN, TMRx_CTRL2: CxIOS and CxCIOS 10: Write protection level 2. The following bits and all bits in level 3 are write protected: TMRx_CCTRL: CxP and CxCP TMRx_BRK: FCSODIS and FCSOEN 11: Write protection level 1. The following bits and all bits in level 2 are write protected: TMRx_CMx: CxOCTRL and CxOBEN Note: Once WPC>0, its content remains frozen until the next system reset.
Bit 7: 0	DTC	0x00	rw	Dead-time configuration This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7: 0] is used for function selection: 0xx: DT = DTC [7: 0] * TDTS 10x: DT = (64+ DTC [5: 0]) * TDTS * 2 110: DT = (32+ DTC [4: 0]) * TDTS * 8 111: DT = (32+ DTC [4: 0]) * TDTS * 16

Note: BRKF, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7: 0] can be write protected, which needs to be configured when first writing TMR9_BRK register.

15.3.4.16 TMR9 and TMR12 DMA control register (TMRx_DMACTRL)

Bit	Name	Reset value	Type	Description
Bit 31:13	Reserved	0x0 0000	resd	Kept at default value.
Bit 12:8	DTB	0x00	rw	DMA transfer bytes This field defines the number of DMA transfers: 00000: 1 byte 00001: 2 bytes 00010: 3 bytes 00011: 4 bytes .... 10000: 17 bytes 10001: 18 bytes
Bit 7:5	Reserved	0x0	resd	Kept at default value.
Bit 4: 0	ADDR	0x00	rw	DMA transfer address offset ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register: 00000: TMRx_CTRL1 00001: TMRx_CTRL2 00010: TMRx_STCTRL .....

15.3.4.17 TMR9 and TMR12 DMA data register (TMRx_DMADT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DMADT	0x0000	rw	DMA data register A write/read operation to the DMADT register accesses any TMR register located at the following address: TMRx peripheral address + ADDR*4 to TMRx peripheral address + ADDR*4 + DTB*4

15.3.5 TMR10, TMR11, TMR13 and TMR14 registers

These peripheral registers must be accessed by words (32 bits).

Table 15-10 TMR10, TMR11, TMR13 and TMR14 register map and reset value

Register	Offset	Reset value
TMRx_CTRL1	0x00	0x0000 0000
TMRx_CTRL2	0x04	0x0000 0000
TMRx_IDEN	0x0C	0x0000 0000
TMRx_ISTS	0x10	0x0000 0000
TMRx_SWEVT	0x14	0x0000 0000
TMRx_CM1	0x18	0x0000 0000
TMRx_CCTRL	0x20	0x0000 0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000 0000
TMRx_PR	0x2C	0x0000 FFFF
TMRx_RPR	0x30	0x0000 0000
TMRx_C1DT	0x34	0x0000 0000
TMRx_BRK	0x44	0x0000 0000
TMRx_DMACTRL	0x48	0x0000 0000
TMRx_DMADT	0x4C	0x0000 0000
TMR14_RMP	0x50	0x0000 0000

15.3.5.1 TMR10, TMR11, TMR13 and TMR14 control register 1 (TMRx_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x00 0000	resd	Kept at its default value
Bit 9: 8	CLKDIV	0x0	rw	Clock divider This field is used to define the division ratio between digital filter sampling frequency (f_{DTS}) and timer clock frequency (f_{CK_INT}). 00: No division, $f_{DTS}=f_{CK\_INT}$ 01: Divided by 2, $f_{DTS}=f_{CK\_INT}/2$ 10: Divided by 4, $f_{DTS}=f_{CK\_INT}/4$ 11: Reserved
Bit 7	PRBEN	0x0	rw	Period buffer enable 0: Period buffer is disabled 1: Period buffer is enabled
Bit 6: 5	TWCMSEL	0x0	rw	Two-way counting mode selection 00: One-way counting mode, depending on the OWCDIR

				bit 01: Two-way up/downcounting mode1. The counter counts up and down alternately, the CxIF bit is set only when the counter is counting down 10: Two-way up/downcounting mode 2. The counter counts up and down alternately, the CxIF bit is set only when the counter is counting up 11: Two-way up/downcounting mode 3. The counter counts up and down alternately, the CxIF bit is set when the counter counting up or down
Bit 4	OWCDIR	0x0	rw	One-way count direction 0: Up 1: Down
Bit 3	OCMEN	0x0	rw	One cycle mode enable This bit is use to select whether to stop counting at an overflow event 0: The counter does not stop at an overflow event 1: The counter stops at an overflow event
Bit 2	OVFS	0x0	rw	Overflow event source This bit is used to select overflow event or DMA request sources. 0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller 1: Only counter overflow generates an overflow event
Bit 1	OVFEN	0x0	rw	Overflow event enable 0: Enabled 1: Disabled
Bit 0	TMREN	0x0	rw	TMR enable 0: Enabled 1: Disabled

15.3.5.2 TMR10, TMR11, TMR13 and TMR14 DMA/interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31:10	Reserved	0x00 0000	resd	Kept at its default value
Bit 9	C1CIOS	0x0	rw	Channel 1 complementary idle output state Output disabled (OEN= 0) after dead-time: 0: C1OUTL=0 1: C1OUTL=1
Bit 8	C1IOS	0x0	rw	Channel 1 idle output state Output disabled (OEN = 0) after dead-time: 0: C1OUT=0 1: C1OUT=1
Bit 7:4	Reserved	0x0	resd	Kept at default value.
Bit 3	DRS	0x0	rw	DMA request source 0: Capture/compare event 1: Overflow event
Bit 2	CCFS	0x0	rw	Channel control bit flash selection This bit only acts on channels that have complementary output. If the channel control bits are buffered: 0: Control bits are updated by setting the HALLSWTR bit

				1: Control bits are updated by setting the HALLSWTR bit or a rising edge on TRGIN.
Bit 1	Reserved	0x0	resd	Kept at default value.
Bit 0	CBCTRL	0x0	rw	Channel buffer control This bit acts on channels that have complementary output. 0: CxEN, CxCEN and CxOCTRL bits are not buffered. 1: CxEN, CxCEN and CxOCTRL bits are buffered. Note: when CBCTRL="1" is asserted, CxEN, CxCEN and CxOCTRL bits are updated only upon a HALL event.

15.3.5.3 TMR10, TMR11, TMR13 and TMR14 DMA/ interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31:10	Reserved	0x00 0000	resd	Kept at default value.
Bit 9	C1DEN	0x0	rw	Channel 1 DMA request enable 0: Disabled 1: Enabled
Bit 8	OVFDEN	0x0	rw	Overflow event DMA request enable 0: Disabled 1: Enabled
Bit 7	BRKIE	0x0	rw	Brake interrupt enable 0: Disabled 1: Enabled
Bit 6	Reserved	0x0	resd	Kept at default value.
Bit 5	HALLIEN	0x0	rw	HALL interrupt enable 0: Disabled 1: Enabled
Bit 4: 2	Reserved	0x0	resd	Kept at default value.
Bit 1	C1IEN	0x0	rw	Channel 1 interrupt enable 0: Disabled 1: Enabled
Bit 0	OVFIEN	0x0	rw	Overflow interrupt enable 0: Disabled 1: Enabled

15.3.5.4 TMR10, TMR11, TMR13 and TMR14 interrupt status register (TMRx_ISTS)

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x00 0000	resd	Kept at its default value.
Bit 9	C1RF	0x0	rw0c	Channel 1 recapture flag This bit indicates whether a recapture is detected when C1IF=1. This bit is set by hardware, and cleared by writing "0". 0: No capture is detected 1: Capture is detected.
Bit 8:	Reserved	0x0	resd	Kept at its default value.
Bit 7	BRKIF	0x0	rw0c	Brake interrupt flag This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing "0"

				0: Inactive level 1: Active level
Bit 6	Reserved	0x0	resd	Default value
Bit 5	HALLIF	0x0	rw0c	HALL interrupt flag This bit is set by hardware upon a trigger event and cleared by writing 0. 0：No HALL event occurred 1：HALL event occurred HALL event：CxEN, CxCEN and CxOCTRL are updated.
Bit 4:2	Reserved	0x0	resd	Default value
Bit 1	C1IF	0x0	rw0c	Channel 1 interrupt flag If the channel 1 is configured as input mode: This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT 0: No capture event occurs 1: Capture event is generated If the channel 1 is configured as output mode: This bit is set by hardware on a compare event. It is cleared by software. 0: No compare event occurs 1: Compare event is generated
Bit 0	OVFIF	0x0	rw0c	Overflow interrupt flag This bit is set by hardware on an overflow event. It is cleared by software. 0: No overflow event occurs 1: Overflow event is generated. If OVFEN=0 and OVFS=0 in the TMRx_CTRL1 register: − An overflow event is generated when OVFSWTR= 1 in the TMRx_SWEVT register; − An overflow event is generated when the counter CVAL is reinitialized by a trigger event.

15.3.5.5 TMR10, TMR11, TMR13 and TMR14 software event register (TMRx_SWEVT)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	BRKSWTR	0x0	wo	Brake event triggered by software This bit is set by software to generate a brake event. 0: No effect 1: Generate a brake event.
Bit 6	Reserved	0x0	resd	Kept at default value.
Bit 5	HALLSWTR	0x0	wo	HALL event triggered by software This bit is set by software to generate a HALL event. 0: No effect 1: Generate a HALL event. Note: This bit acts only on channels that have complementary output.
Bit 4: 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	C1SWTR	0x0	wo	Channel 1 event triggered by software This bit is set by software to generate a channel 1 event. 0: No effect 1: Generate a channel 1 event.
Bit 0	OVFSWTR	0x0	wo	Overflow event triggered by software

				This bit is set by software to generate an overflow event. 0: No effect 1: Generate an overflow event.

15.3.5.6 TMR10, TMR11, TMR13 and TMR14 channel mode register 1 (TMRx_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:7	Reserved	0x000 0000	resd	Kept at its default value.
Bit 6: 4	C1OCTRL	0x0	rw	Channel 1 output control This field defines the behavior of the original signal C1ORAW. 000: Disconnected. C1ORAW is disconnected from C1OUT; 001: C1ORAW is high when TMRx_CVAL=TMRx_C1DT 010: C1ORAW is low when TMRx_CVAL=TMRx_C1DT 011: Switch C1ORAW level when TMRx_CVAL=TMRx_C1DT 100: C1ORAW is forced low 101: C1ORAW is forced high. 110: PWM mode A – OWCDIR=0, C1ORAW is high once TMRx_C1DT>TMRx_CVAL, else low; – OWCDIR=1, C1ORAW is low once TMRx_ C1DT <TMRx_CVAL, else high; 111: PWM mode B – OWCDIR=0, C1ORAW is low once TMRx_ C1DT >TMRx_CVAL, else high; – OWCDIR=1, C1ORAW is high once TMRx_ C1DT <TMRx_CVAL, else low. *Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.*
Bit 3	C1OBEN	0x0	rw	Channel 1 output buffer enable 0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately. 1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.
Bit 2	C1OIEN	0x0	rw	Channel 1 output enable immediately In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event. 0: Need to compare the CVAL with C1DT before generating an output 1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Reserved 11: Reserved

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7: 4	C1DF	0x0	rw	Channel 1 digital filter This field defines the digital filter of the channel 1. “N” refers to the number of filtering, meaning that 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: $f_{SMPLING}=f_{DTS}/4$, 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 3: 2	C1IDIV	0x0	rw	Channel 1 input divider This field defines Channel 1 input divider. 00: No divider. An input capture is generated at each active edge. 01: An input compare is generated every 2 active edges 10: An input compare is generated every 4 active edges 11: An input compare is generated every 8 active edges Note: the divider is reset once C1EN=’0’
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN=’0’: 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Reserved 11: Reserved

15.3.5.7 TMR10, TMR11, TMR13 and TMR14 channel control register (TMRx_CCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 4	Reserved	0x000 0000	resd	Kept at its default value.
Bit 3	C1CP	0x0	rw	Channel 1 complementary polarity 0: C1COUT is active high. 1: C1COUT is active low.
Bit 2	C1CEN	0x0	rw	Channel 1 complementary enable 0: Output is disabled. 1: Output is enabled.
Bit 1	C1P	0x0	rw	Channel 1 polarity When the channel 1 is configured as output mode: 0: C1OUT is active high 1: C1OUT is active low When the channel 1 is configured as input mode: C1CP/C1P together are used to define the active edge of an input signal. 00: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted. 01: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted. 10: Reserved 11 C1IN active edge are rising and falling edges. When used as external trigger, C1IN is not inverted.
Bit 0	C1EN	0x0	rw	Channel 1 enable 0: Input or output is disabled 1: Input or output is enabled

Table 15-11 Complementary output channel OCx and OCxN control bits with break feature

Control bit					Output state (1)	
OEN bit	FCSODIS bit	FCSOEN bit	CxEN bit	CxCEN bit	CxOUT output state	CxCOUT output state
1	X	0	0	0	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	0	1	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		0	1	0	CxORAW+ polarity CxOUT= CxORAW xor CxP, Cx_EN=1	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	1	1	CxORAW+polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
		1	0	0	Output disabled (no driven by the timer) CxOUT=CxP, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=CxCP, CxCEN=0
		1	0	1	Off-state (Output enabled with inactive level) CxOUT=CxP, Cx_EN=1	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		1	1	0	CxORAW + polarity, CxOUT= CxORAW xor CxP, Cx_EN=1	Off-state (Output enabled with inactive level) CxCOUT=CxCP, CxCEN=1
		1	1	1	CxORAW+ polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
0	0	X	0	0	Output disabled (no driven by the timer)	
	0		0	1	Asynchronously: CxOUT=CxP, Cx_EN=0, CxCOUT=CxCP, CxCEN=0;	
	0		1	0	If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	
	0		1	1		
	1		0	0	Off-state (output enabled and with inactive level)	
	1		0	1	Asynchronously: CxOUT =CxP, Cx_EN=1, CxCOUT=CxCP, CxCEN=1;	
	1		1	0	If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	
	1		1	1		

Note: If the two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

Note: The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.

15.3.5.8 TMR10, TMR11, TMR13 and TMR14 counter value (TMRx_CVAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	CVAL	0x0000	rw	Counter value

15.3.5.9 TMR10, TMR11, TMR13 and TMR14 division value (TMRx_DIV)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DIV	0x0000	rw	Divider value The counter clock frequency fCK_CNT = fTMR_CLK /(DIV[15: 0]+1). DIV contains the value written at an overflow event.

15.3.5.10 TMR10, TMR11, TMR13 and TMR14 period register (TMRx_PR)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	PR	0xFFFF	rw	Period value This defines the period value of the TMRx counter. The timer stops working when the period value is 0.

15.3.5.11 TMR10, TMR11, TMR13 and TMR14 repetition period register register (TMRx_RPR) (x=10/11/13/14)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at default value.
Bit 7: 0	RPR	0x00	rw	Repetition of period value This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0.

15.3.5.12 TMR10, TMR11, TMR13 and TMR14 channel 1 data register (TMRx_C1DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	C1DT	0x0000	rw	Channel 1 data register When the channel 1 is configured as input mode: The C1DT is the CVAL value stored by the last channel 1 input event (C1IN). When the channel 1 is configured as output mode: C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured.

15.3.5.13 TMR10, TMR11, TMR13 and TMR14 break register (TMRx_BRK) (x=10/11/13/14)

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x000	resd	Kept at default value.
Bit 19: 16	BKF	0x0	rw	Brake input filter This field is used to set the filter for brake input. “N” refers to the number of filtering, meaning that 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: $f_{SMPLING}$=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 Note: if there is no way to guarantee clean break input, it is recommended to enable break input filtering.
Bit 15	OEN	0x0	rw	Output enable This bit acts on the channels as output. It is used to enable CxOUT and CxCOUT outputs. 0: Disabled 1: Enabled
Bit 14	AOEN	0x0	rw	Automatic output enable OEN is set automatically at an overflow event. 0: Disabled 1: Enabled
Bit 13	BRKV	0x0	rw	Brake input validity This bit is used to select the active level of a brake input. 0: Brake input is active low. 1: Brake input is active high.
Bit 12	BRKEN	0x0	rw	Brake enable This bit is used to enable brake input. 0: Brake input is disabled. 1: Brake input is enabled.
Bit 11	FCSOEN	0x0	rw	Frozen channel status when holistic output enable This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=1. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output inactive level.
Bit 10	FCSODIS	0x0	rw	Frozen channel status when holistic output disable This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=0. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output idle level.
Bit 9: 8	WPC	0x0	rw	Write protection configuration

				This field is used to enable write protection. 00: Write protection is OFF. 01: Write protection level 3, and the following bits are write protected: TMRx_BRK: BRKF. AOEN. BRLV and DTC TMRx_CTRL2: CxIOS and CxCIOS 10: Write protection level 2. The following bits and all bits in level 3 are write protected: TMRx_CCTRL: CxP and CxCP TMRx_BRK: FCSODIS and FCSOEN 11: Write protection level 1. The following bits and all bits in level 2 are write protected: TMRx_CMx: CxOCTRL and CxOBEN Note: Once WPC>0, its content remains frozen until the next system reset.
Bit 7: 0	DTC	0x00	rw	Dead-time configuration This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7: 0] is used for function selection: 0xx: DT = DTC [7: 0] * TDTS 10x: DT = (64+ DTC [5: 0]) * TDTS * 2 110: DT = (32+ DTC [4: 0]) * TDTS * 8 111: DT = (32+ DTC [4: 0]) * TDTS * 16

Note: BRKF, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7: 0] can be write protected, which needs to be configured when first writing TMRx_BRK register.

15.3.5.14 TMR10, TMR11, TMR13 and TMR14 DMA control register (TMRx_DMACTRL) (x=10/11/13/14)

Bit	Name	Reset value	Type	Description
Bit 31:13	Reserved	0x0 0000	resd	Kept at default value.
Bit 12:8	DTB	0x00	rw	DMA transfer bytes This field defines the number of DMA transfers: 00000: 1 byte 00001: 2 bytes 00010: 3 bytes 00011: 4 bytes .... 10000: 17 bytes 10001: 18 bytes
Bit 7:5	Reserved	0x0	resd	Kept at default value.
Bit 4: 0	ADDR	0x00	rw	DMA transfer address offset ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register: 00000: TMRx_CTRL1 00001: TMRx_CTRL2 00010: TMRx_STCTRL .....

15.3.5.15 TMR10, TMR11, TMR13 and TMR14 DMA data register (TMRx_DMADT) (x=10/11/13/14)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DMADT	0x0000	rw	DMA data register

	A write/read operation to the DMADT register accesses any TMR register located at the following address: TMRx peripheral address + ADDR*4 to TMRx peripheral address + ADDR*4 + DTB*4

15.3.5.16 TMR14 channel 1 channel input remapping register (TMR14_RMP)

Bit	Name	Reset value	Type	Description
Bit 31: 2	Reserved	0x0000 0000	resd	Kept at default value.
Bit 1: 0	TMR14_CH1_IRMP	0x0	rw	TMR14 channel 1 input remap 00：TMR14 channel 1 input tied to GPIO 01：ERTC_CLK 10：HEXT_DIV 11：CLK_OUT

15.4 Advanced-control timers (TMR1 and TMR8)

15.4.1 TMR1 and TMR8 introduction

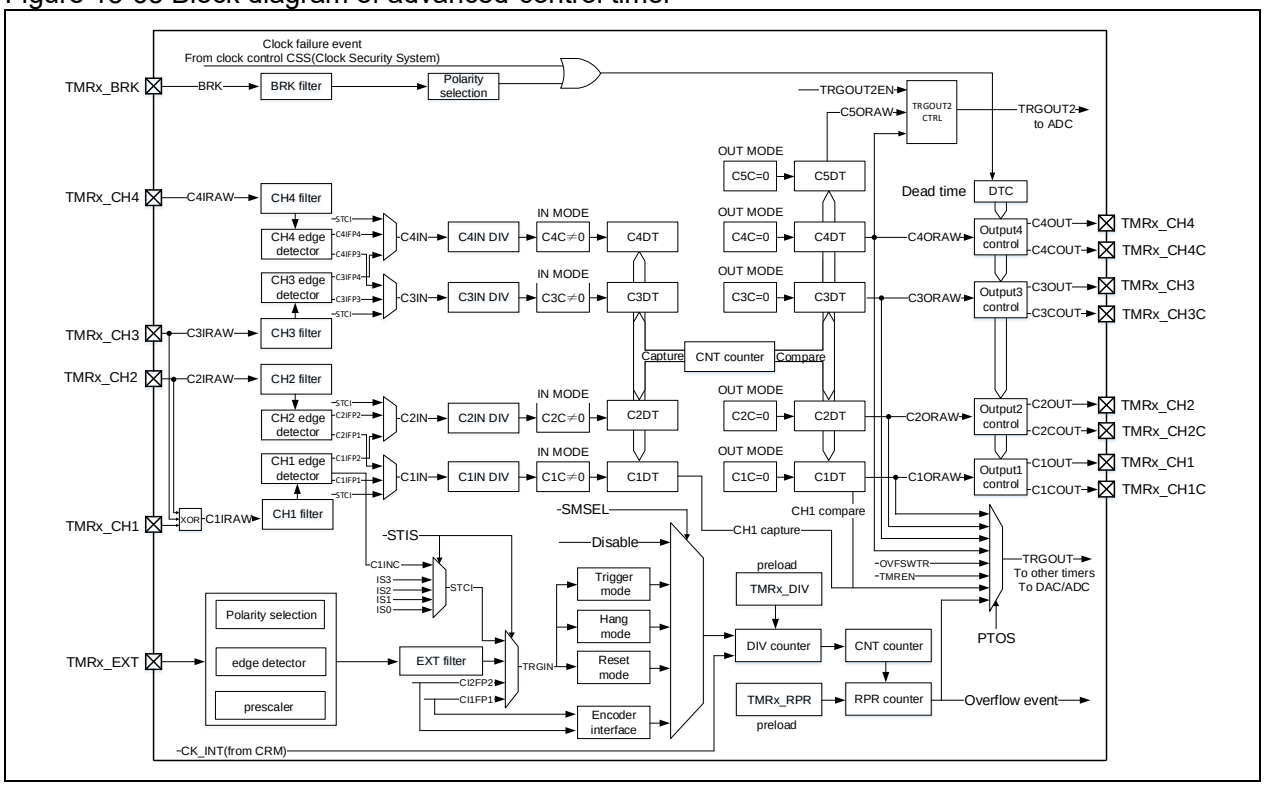
Each of the advanced-control timer (TMR1 and TMR8) consists of a 16-bit counter supporting up and down counting modes, four capture/compare registers, and five independent channels. They can be used for dead-time insertion, input capture and programmable PWM output.

15.4.2 TMR1 and TMR8 main features

The main functions of general-purpose TMR1 and TMR8 include:

- Source of counter clock: internal clock, external clock an internal trigger input
- 16-bit up, down, up/down, repetition and encoder mode counter
- Five independent channels: channel 5 for output compare only via TRGOUT2, and channel 1 to 4 for input compare, output compare, PWM generation, one-pulse mode and dead-time insertion output and embedded dead-time
- Four independent channels for complementary output
- TMR break function
- Synchronization control between master and slave timers
- Interrupt/DMA is generation at overflow event, trigger event, break signal input and channel event
- Support TMR burst DMA transfers

Figure 15-65 Block diagram of advanced-control timer

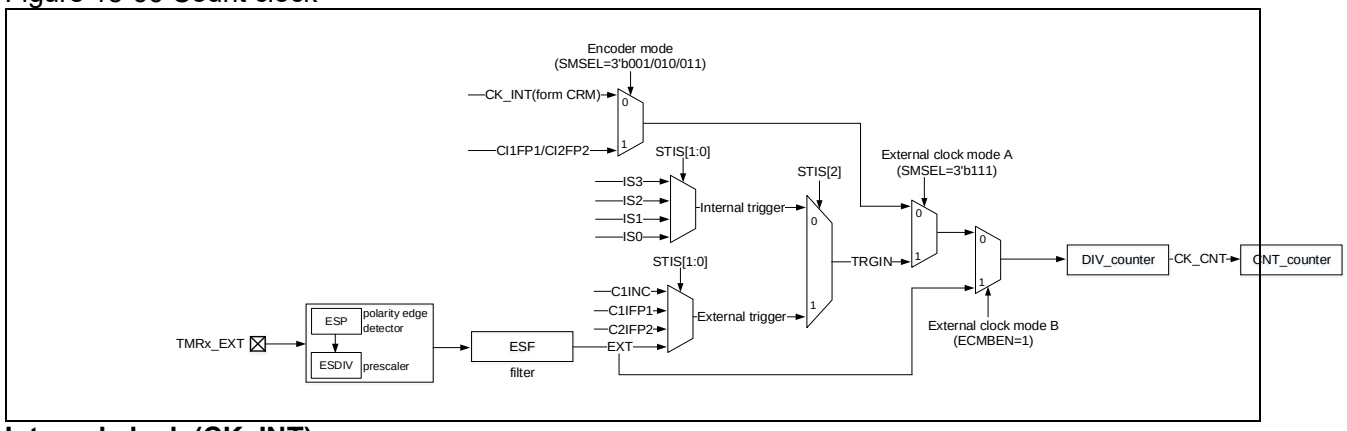


15.4.3 TMR1 and TMR8 functional overview

15.4.3.1 Counting clock

The count clock of TMR1 and TMR8 can be provided by the internal clock (CK_INT), external clock (external clock mode A and B) and internal trigger input (ISx).

Figure 15-66 Count clock



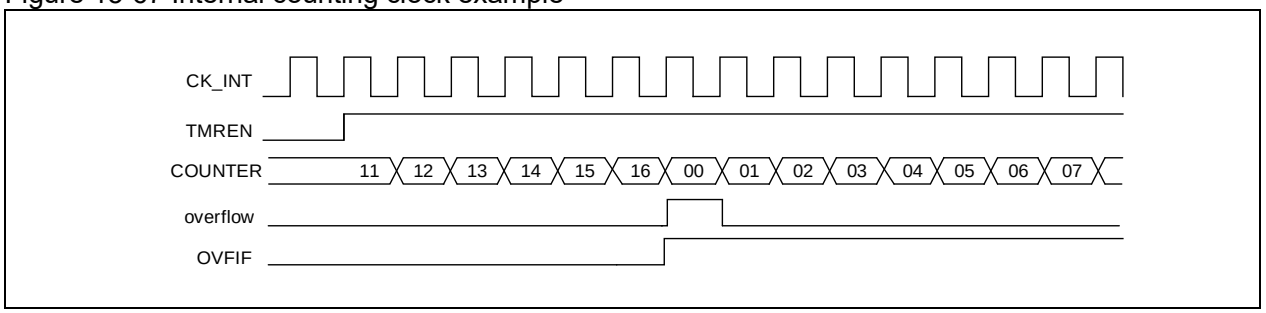
Internal clock (CK_INT)

By default, the CK_INT divided by the prescaler is used to drive the counter to start counting. When TMR's APB clock prescaler factor is 1, the CK_INT frequency is equal to that of APB; otherwise, it doubles the APB clock frequency.

The configuration process is as follows:

- Configure the TWCMSEL[1:0] bit in the TMRx_CTRL1 register and select count mode. If the one-way count direction is set, configure OWCDIR bit in the TMRx_CTRL1 register to select the specific direction.
- Configure the TMRx_DIV register and set counting frequency.
- Configure the TMRx_PR register and set counting period.
- Configure the TMREN bit in the TMRx_CTRL1 register and enable the counter.

Figure 15-67 Internal counting clock example



External clock (TRGIN/EXT)

The counter clock can be provided by two external clock sources, namely, TRGIN and EXT signals.

SMSEL=3'b111 in the TMRx_STCTRL register: External clock mode A is selected. Select an external clock source TRGIN signal by setting the STIS[2:0] bit in the TMRx_STCTRL register to drive the counter to start counting. The external clock sources include: C1INC (STIS=3'b100, channel 1 rising edge and falling edge), C1IFP1 (STIS=3'b101, channel 1 signal with filtering and polarity selection), C2IFP2 (STIS=3'b110, channel 2 signal with filtering and polarity selection) and EXT (STIS=3'b111, external input signal with polarity selection, frequency division and filtering).

ECMBEN=1 in the TMRx_STCTRL register: External clock mode B is selected. The counter is driven by external input that has gone through polarity selection, frequency division and filtering. The external clock mode B is equivalent to the external clock mode A which selects EXT signal as an external force TRGIN.

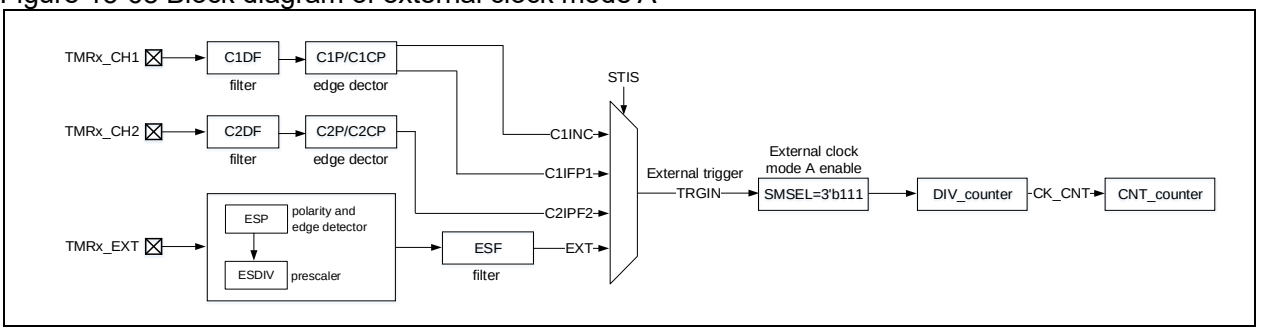
To use external clock mode A, follow the steps below:

- Set external source TRGIN parameters.
  - If the TMRx_CH1 is used as a source of TRGIN, it is necessary to configure channel 1 input filter (C1DF[3:0] in TMRx_CM1 register) and channel 1 input polarity (C1P/C1CP in TMRx_CCTRL register);
  - If the TMRx_CH2 is used as source of TRGIN, it is necessary to configure channel 1 input filter (C2DF[3:0] in TMRx_CM1 register) and channel 2 input polarity (C2P/C2CP in TMRx_CCTR register);
  - If the TMRx_EXT is used as a source of TRGIN, it is necessary to configure the external signal polarity (ESP in TMRx_STCTRL register), external signal frequency division (ESDIV[1:0] in TMRx_STCTRL) and external signal filter (ESF[3:0] in TMRx_STCTRL register).
- Set TRGIN signal source using the STIS[2:0] bit in TMRx_STCTRL register.
- Enable external clock mode A by setting SMSEL=3'b111 in TMRx_STCTR register.
- Set counting frequency through the DIV[15:0] in TMRx_DIV register.
- Set counting period through the PR[15:0] in TMRx_PR register.
- Enable counter through the TMREN bit in TMRx_CTRL1 register.

To use external clock mode B, follow the steps below:

- Set external signal polarity through the ESP bit in TMRx_STCTRL register.
- Set external signal frequency division through the ESDIV[1:0] bit in TMRx_STCTRL register.
- Set external signal filter through the ESF[3:0] bit in TMRx_STCTRL register.
- Enable external clock mode B through the ECMBEN bit in TMRx_STCTR register.
- Set counting frequency through the DIV[15:0] bit in TMRx_DIV register.
- Set counting period through the PR[15:0] bit in TMRx_PR register.
- Enable counter through the TMREN in TMRx_CTRL1 register.

Figure 15-68 Block diagram of external clock mode A



Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

Figure 15-69 Counting in external clock mode A

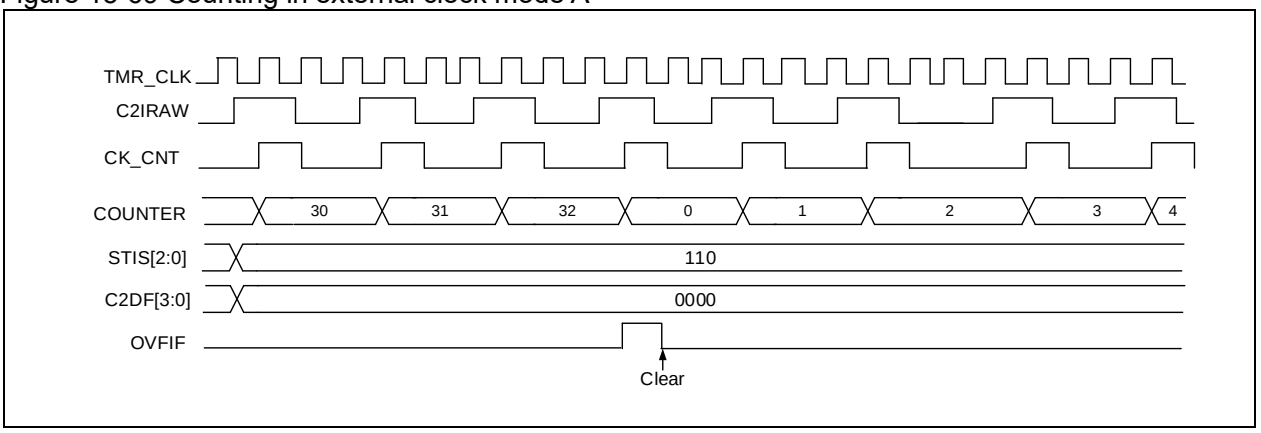
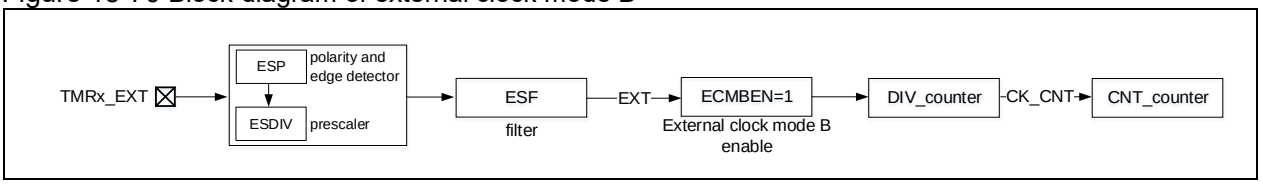
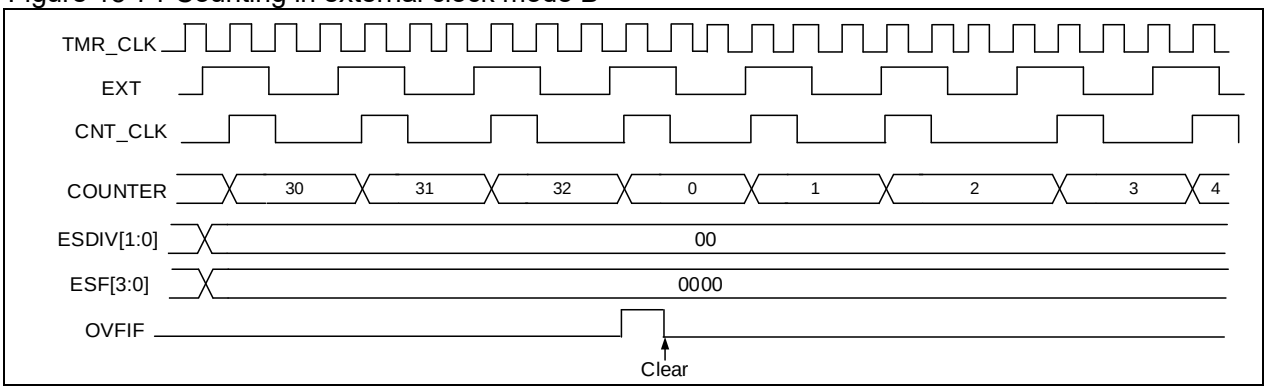


Figure 15-70 Block diagram of external clock mode B



Note: The delay between the signal on the input side and the actual clock of the counter is due to the synchronization circuit.

Figure 15-71 Counting in external clock mode B



Internal trigger input (ISx)

Timer synchronization allows interconnection between several timers. The TMR_CLK of one timer can be provided by the TRGOUT signal output by another timer. Set the STIS[2: 0] bit in the TMRx_STCTRL register to select internal trigger signal to enable counting.

Each timer consists of a 16-bit prescaler, which is used to generate the CK_CNT that enables the counter to count. The frequency division relationship between the CK_CNT and TMR_CLK can be adjusted by setting the value of the TMRx_DIV register. The prescaler value can be modified at any time, but it takes effect only when the next overflow event occurs.

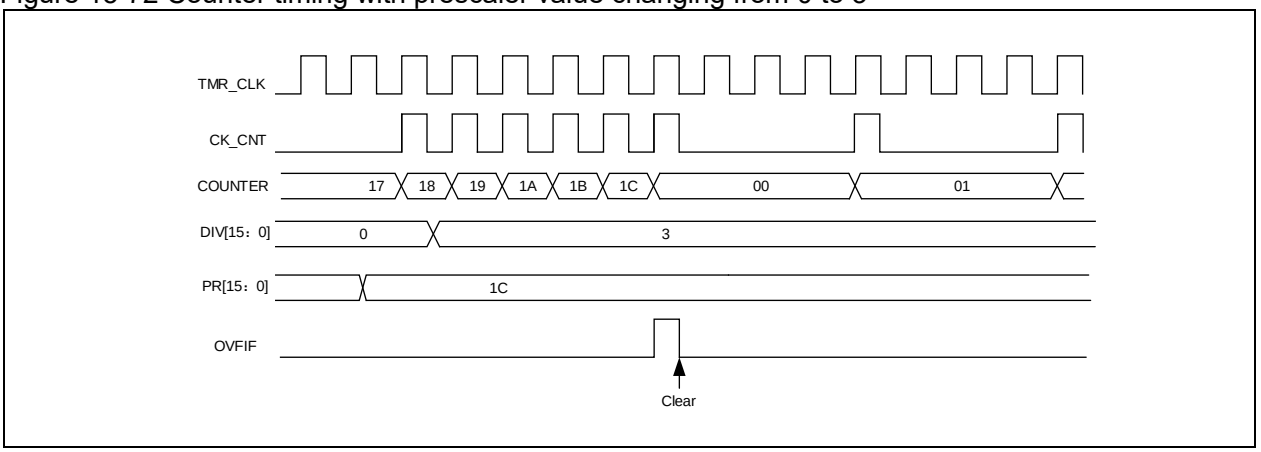
The internal trigger input is configured as follows:

- Set the TMRx_PR register to set counting period.
- Set the TMRx_DIV register to set counting frequency.
- Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 register to set count mode.
- Set the STIS[2:0] bit (range: 3'b000~3'b011) in the TMRx_STCTRL register and select internal trigger.
- Set SMSEL[2:0]=3'b111 in the TMRx_STCTRL register and select external clock mode A.
- Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.

Table 15-12 TMRx internal trigger connection

Slave timer	IS0 (STIS=000)	IS1 (STIS=001)	IS2 (STIS=010)	IS3 (STIS=011)
TMR1	TMR5	TMR2	TMR3	TMR4
TMR8	TMR1	TMR2	TMR4	TMR5

Figure 15-72 Counter timing with prescaler value changing from 0 to 3



15.4.3.2 Counting mode

The advanced-control timer consists of an internal 16-bit counter supporting up, down, up/down counting modes to meet different application scenarios.

The TMRx_PR register is used to set the counting period. The value in the TMRx_PR is immediately moved to the shadow register by default. When the periodic buffer is enabled (PRBEN=1 in the TMRx_CTRL1 register), the value in the TMRx_PR register is transferred to the shadow register only at an overflow event. The OVFEN and OVFS bits are used to configure the overflow event.

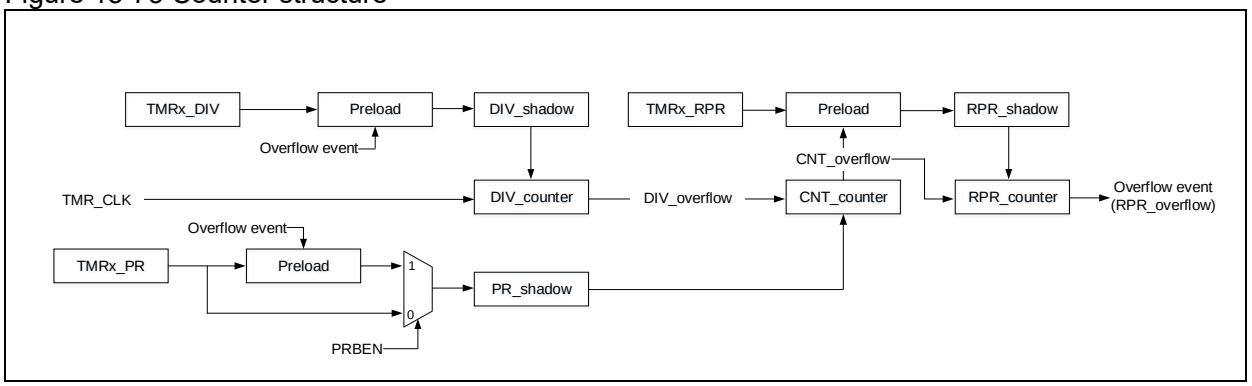
The TMRx_DIV register is used to configure the counting frequency. The counter counts once every count clock period (DIV[15:0]+1). The value in the TMRx_DIV register is updated to the shadow register at an overflow event.

Reading the TMRx_CNT register returns to the current counter value, and writing to the TMRx_CNT register updates the current counter value to the value being written.

An overflow event is generated by default. Set OVFEN=1 in the TMRx_CTRL1 to disable generation of overflow events. The OVFS bit in the TMRx_CTRL1 register is used to select overflow event source. By default, counter overflow/underflow, setting OVFSWTR bit and the reset signal generated by the slave timer controller in reset mode trigger the generation of an overflow event. When the OVFS bit is set, only counter overflow/underflow triggers an overflow event.

Setting TMREN=1 to enable the timer to start counting. Base on synchronization logic, however, the actual enable signal TMR_EN is set 1 clock cycle after the TMREN is set.

Figure 15-73 Counter structure



Upcounting mode

Set TWCMSEL[1:0]=2'b00 and OWCDIR=1'b0 in the TMRx_CTRL1 register to enable upcounting mode. In this mode, the counter counts from 0 to the value programmed in the TMRx_PR register, then restarts from 0, and generates a counter overflow event, with the OVFIF bit being set to 1. If the overflow event is disabled, the counter is no longer reloaded with the preload value and period value at a counter overflow event; otherwise, the counter is updated with the preload value and period value on an overflow event.

Figure 15-74 Overflow event when PRBEN=0

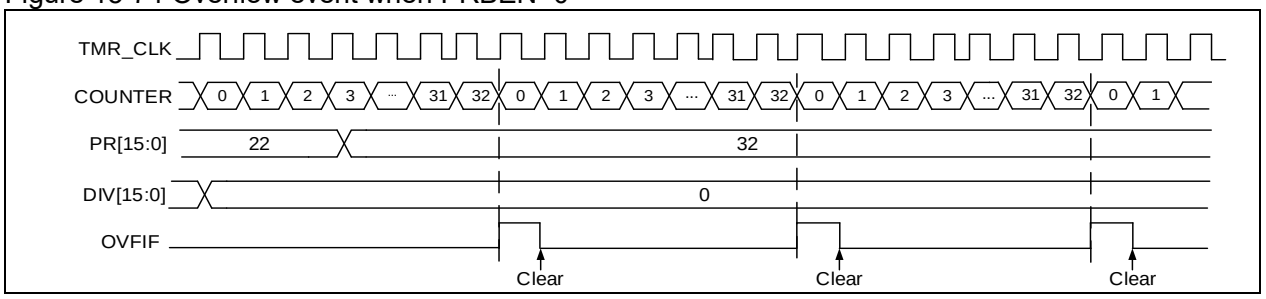
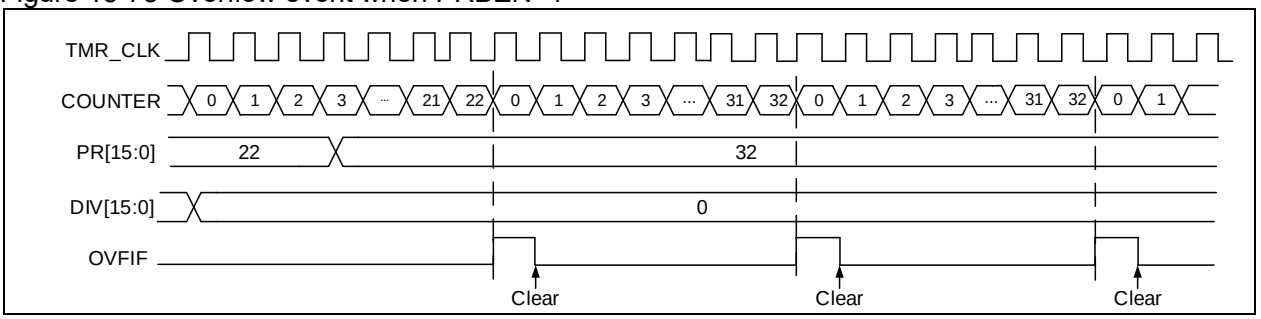


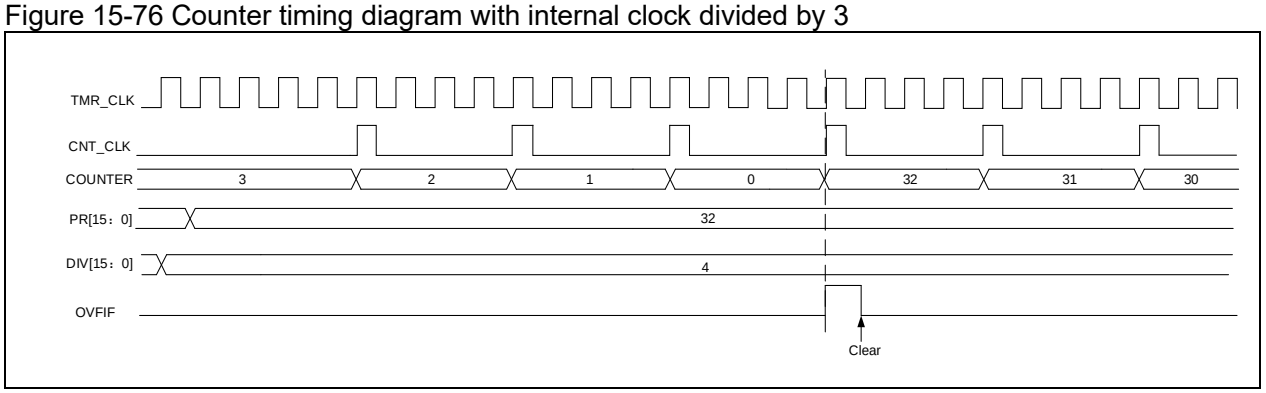
Figure 15-75 Overflow event when PRBEN=1



Downcounting mode

Set TWCMSEL[1:0]=2'b00 and OWCDIR=1'b1 in the TMRx_CTRL1 register to enable downcounting mode. In this mode, the counter counts from the value programmed in the TMRx_PR register down to 0, and restarts from the value programmed in the TMRx_PR register, and generates a counter underflow event.

Figure 15-76 Counter timing diagram with internal clock divided by 3



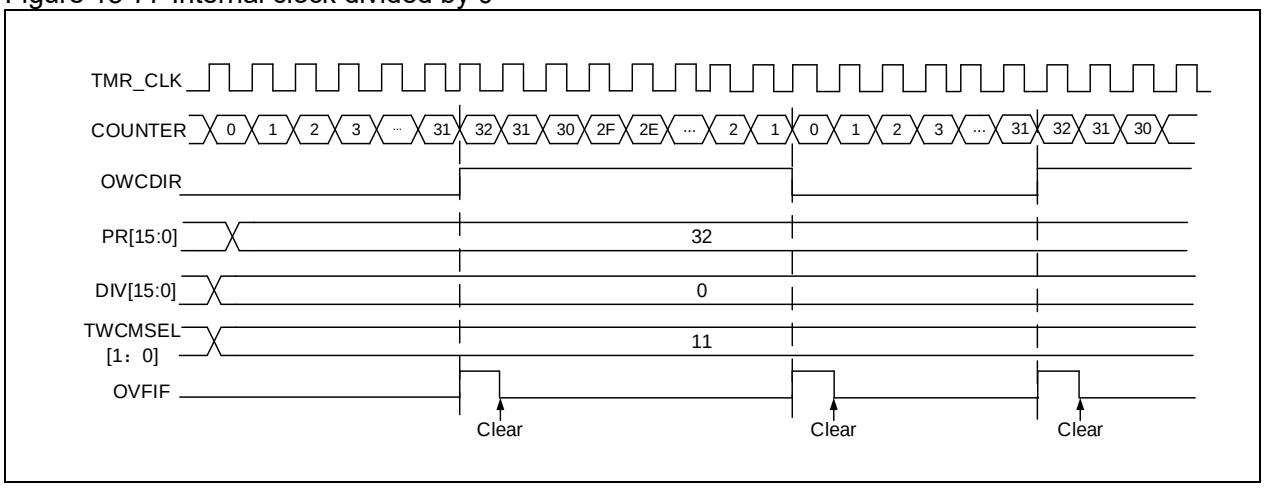
Up/down counting mode

Set TWCMSEL[1:0]≠2'b00 in the TMRx_CTRL1 register to enable up/down counting mode. In this mode, the counter counts up/down alternatively. When the counter counts from the value programmed in the TMRx_PR register down to 1, an underflow event is generated, and then restarts counting from 0; when the counter counts from 0 to the value of the TMRx_PR register -1, an overflow event is generated, and then restarts counting from the value of the TMRx_PR register. The OWCDIR bit indicates the current counting direction.

The TWCMSEL[1:0] bit in the TMRx_CTRL1 register is also used to select the CxIF flag setting method in up/down counting mode. In up/down counting mode 1 (TWCMSEL[1:0]=2'b01), CxIF flag can only be set when the counter counts down; in up/down counting mode 2 (TWCMSEL[1:0]=2'b10), CxIF flag can only be set when the counter counts up; in up/down counting mode 3 (TWCMSEL[1:0]=2'b11), CxIF flag can be set when the counter counts up/down.

Note: The OWCDIR is read-only in up/down counting mode.

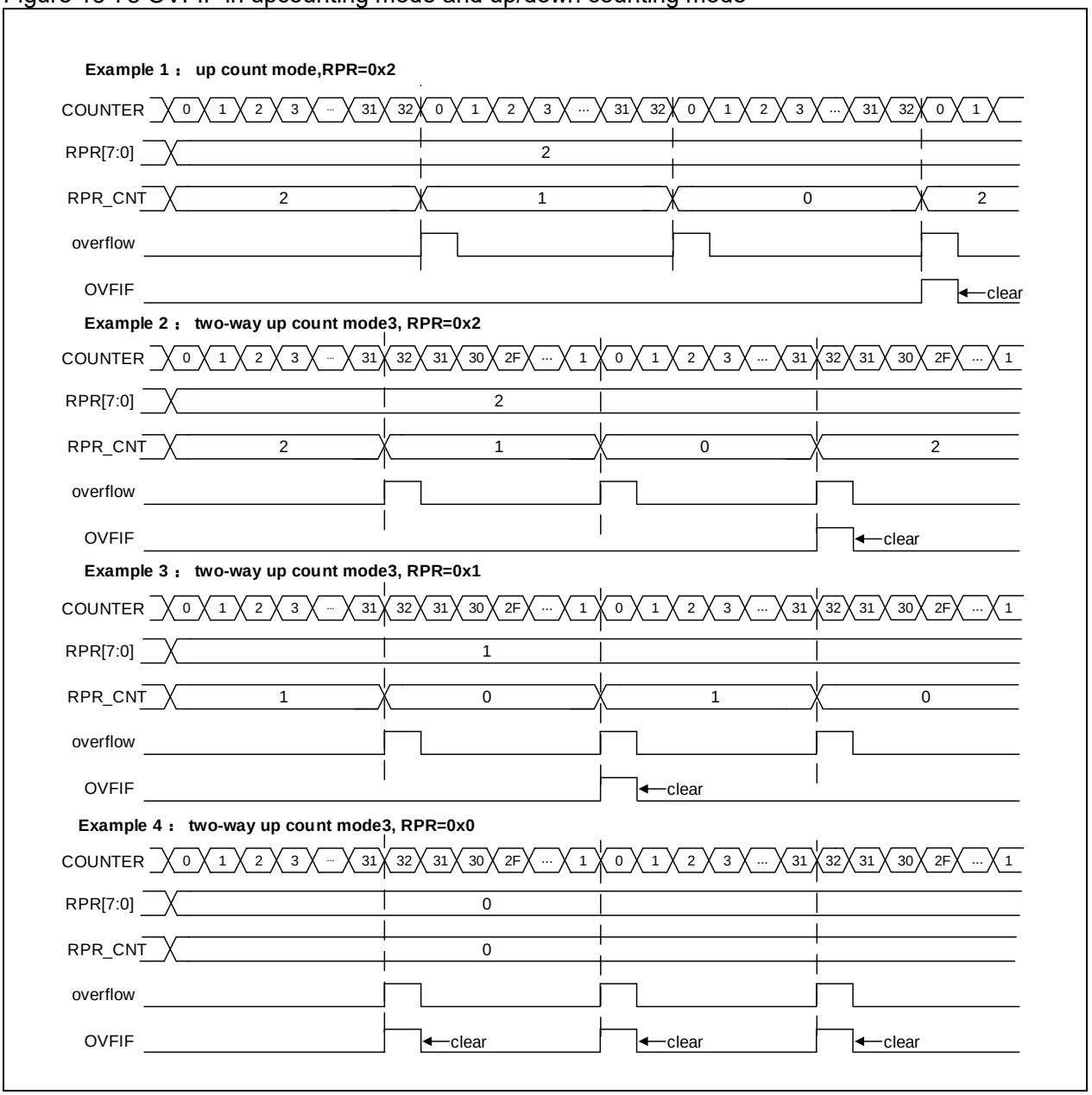
Figure 15-77 Internal clock divided by 0



Repetition counter mode

The TMRx_RPR register is used to set the counting period of repetition counter. The repetition counter mode is enabled when the repetition counter value is not equal to 0. In this mode, the repetition counter is decremented at each counter overflow (RPR[15:0]+1). An overflow event is generated when the repetition counter reaches 0. The frequency of the overflow event can be adjusted by setting the repetition counter value.

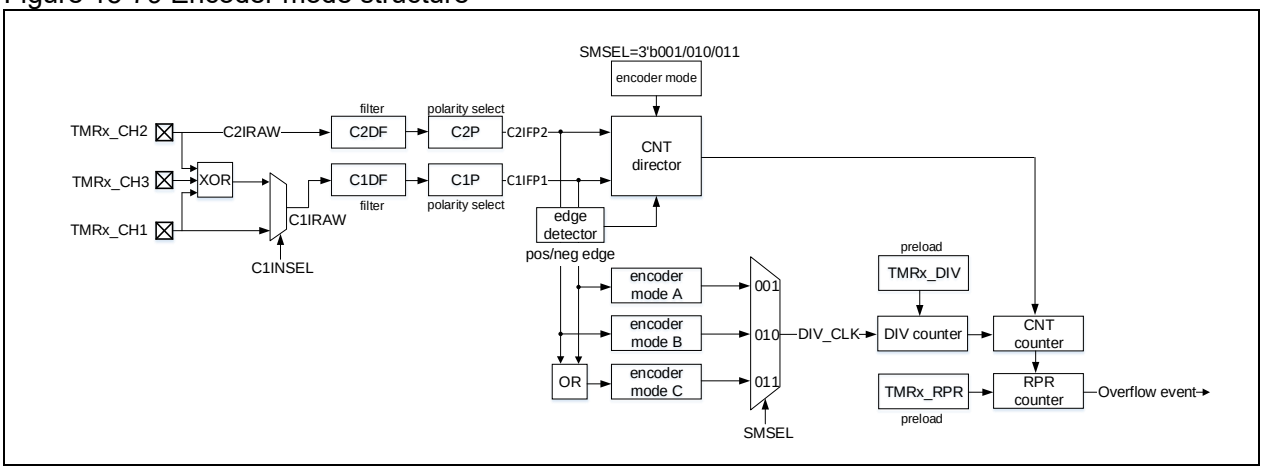
Figure 15-78 OVFIF in upcounting mode and up/down counting mode



Encoder interface mode

In this mode, the two inputs (C1IN/C2IN) are required. Depending on the level on one input, the counter counts up or down on the edge of the other input. The OWCDIR bit indicates the direction of the counter.

Figure 15-79 Encoder mode structure



Encoder mode A: SMSEL=3'b001. The counter counts on the selected C1IFP1 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C1IFP1 and the level of C2IFP2.

Encoder mode B: SMSEL=3'b010. The counter counts on the selected C2IFP2 edge (rising and falling edges), and the counting direction is dependent on the edge direction of C2IFP2 and the level of C1IFP1.

Encoder mode C: SMSEL=3'b011. The counter counts on both C1IFP1 and C2IFP2 edges (rising and falling edges). The counting direction is dependent on the C1IFP1 edge direction and C2IFP2 level, and C2IFP2 edge direction and C1IFP1 level.

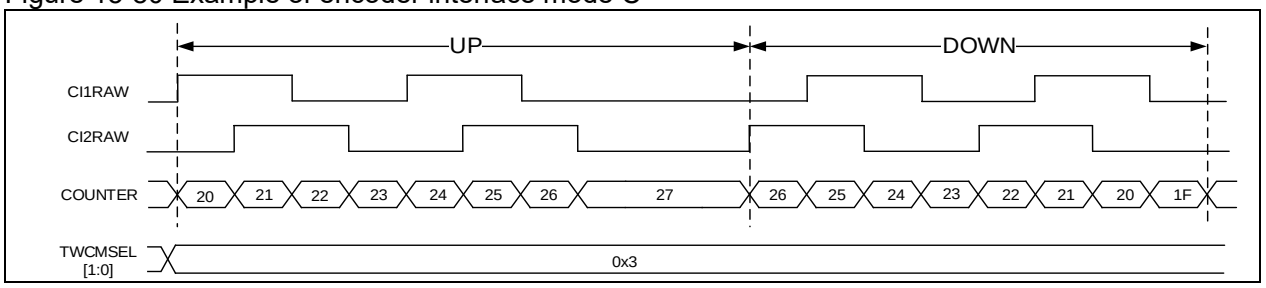
To use encoder mode, follow the procedures below:

- Set channel 1 input signal filtering through the C1DF[3:0] bit in the TMRx_CM1 register;
  Set channel 1 input signal active level through the C1P bit in the TMRx_CCTRL register.
- Set channel 2 input signal filtering through the C2DF[3:0] bit in the TMRx_CM1 register;
  Set channel 2 input signal active level through the C2P bit in the TMRx_CCTRL register.
- Set channel 1 as input mode through the C1C[1:0] bit in the TMRx_CM1 register;
  Set channel 2 as input mode through the C2C[1:0] bit in the TMRx_CM1 register.
- Select encoder mode A (SMSEL=3'b001), encoder mode B (SMSEL=3'b010), or the encoder mode C (SMSEL=3'b011) by setting the SMSEL[2:0] bit in the TMRx_STCTRL register.
- Set counting cycles through the PR[15:0] bit in the TMRx_PR register.
- Set counting frequency through the DIV[15:0] bit in the TMRx_DIV register.
- Configure the corresponding IOs of TMRx_CH1 and TMRx_CH2 as multiplexed mode.
- Enable counter through the TMREN bit in the TMRx_CTRL1 register.

Table 15-13 Counting direction versus encoder signals

Active edge	Level on opposite signal (C1IFP1 to C2IFP2, C2IFP2to C1IFP1)	C1IFP1signal		C2IFP2signal	
		Rising	Falling	Rising	Falling
C1IFP1	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
C2IFP2	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
C1IFP1 and C2IFP2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

Figure 15-80 Example of encoder interface mode C



15.4.3.3 TMR input function

Each timer of TMR1and TMR8 has four independent channels. Each channel can be configured as input or output.

As input, each channel input signal is handled as follows:

- TMRx_CHx outputs the pre-processed CxIRAW. The C1INSEL bit is used to select the source of C1IRAW from TMRx_CH1 or the XOR-ed TMRx_CH1, TMRx_CH2 and TMRx_CH3. The sources of C2IRAW, C3IRAW and C4IRAW are TMRx_CH2, TMRx_CH3 and TMRx_CH4, respectively.
- CxIRAW inputs digital filter and outputs filtered CxIF signal. The digital filter uses the CxDF bit in the TMRx_CM1/CM2 register to program sampling frequency and sampling times.
- CxIF inputs edge detector, and outputs the CxIFPx signal after edge selection. The edge selection depends on both CxP and CxCP bits in the TMRx_CCTRL register. It is possible to select input rising edge, falling edge or both edges.
- CxIFPx inputs capture signal selector, and outputs the CxIN signal after capture signal selection. The capture signal selection is defined by CxC bit in the TMRx_CM1/CM2 register. It is possible to select CxIFPx, CyIFPx or STCI as CxIN source. Of those, CyIFPx (x≠y) is the CyIFPy signal that is from Y channel. The STCI comes from slave timer controller, and its source is selected by STIS bit.
- CxIN outputs the CxIPS signal that is divided by input channel divider. The divider factor can be defined as No division, /2, /4 or /8, by the CxIDIV bit.

Figure 15-81 Input/output channel 1 main circuit

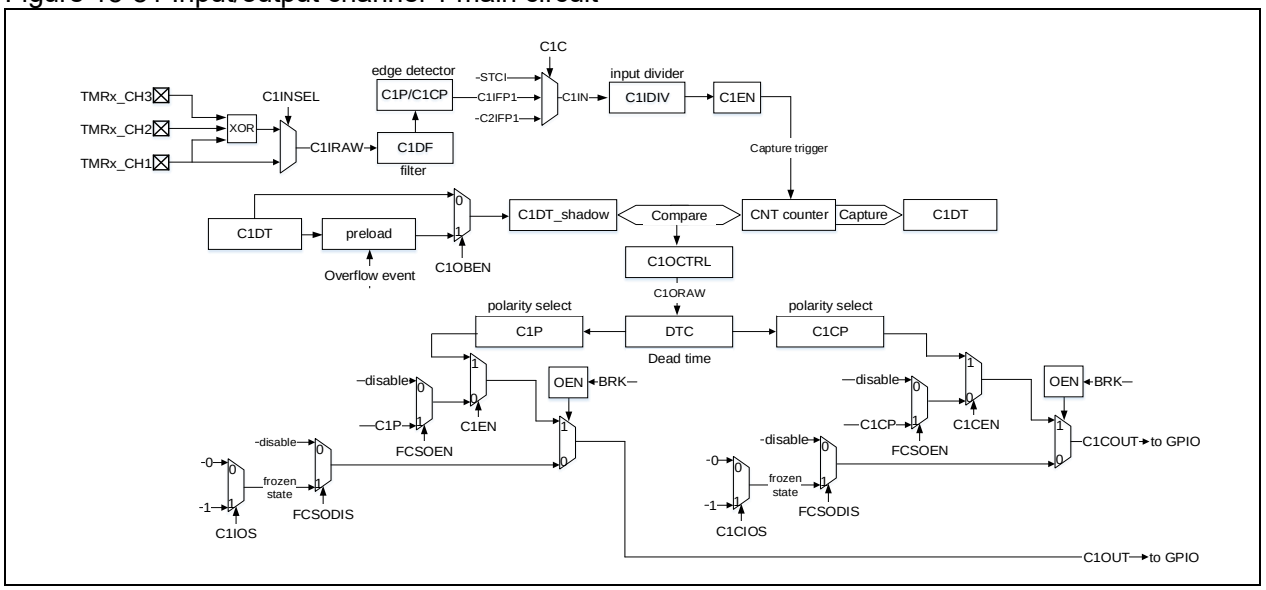
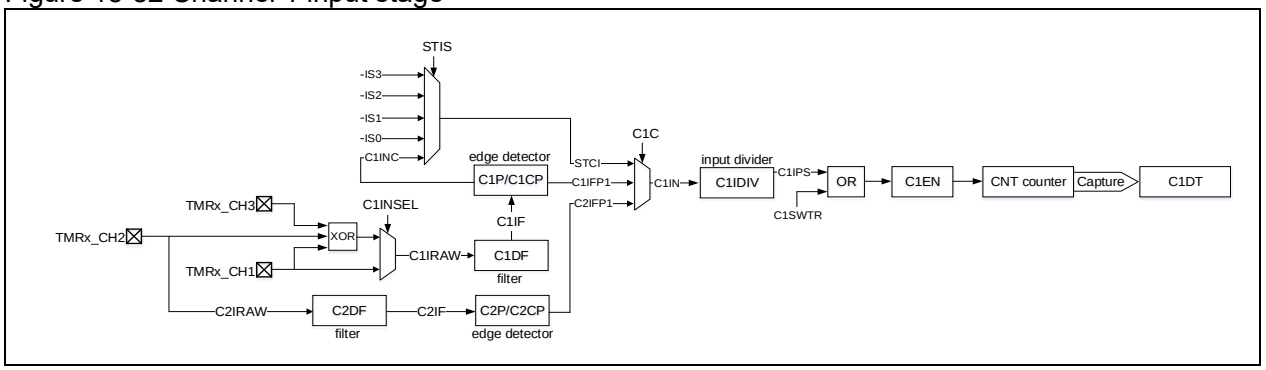


Figure 15-82 Channel 1 input stage



Input mode

In input mode, the TMRx_CxDT registers latch the current counter values after the selected trigger signal is detected, and the capture compare interrupt flag bit (CxIF in the TMRx_ISTS register) is set to 1. An interrupt/DMA request will be generated if the CxIEN or CxDEN bit in the TMRx_IDEN register are enabled. If the selected trigger signal is detected when the CxIF is set, a capture overflow event occurs. The TMRx_CxDT register overwrites the recorded value with the current counter value, and the CxRF is set to 1 in the TMRx_ISTS register.

To capture the rising edge of C1IN input, following the configuration procedure mentioned below:

- Set C1C=2'b01 in the TMRx_CxDT register to select the C1IN as channel 1 input.
- Set the filter bandwidth of C1IN signal (CxDF[3: 0]).
- Set the active edge on the C1IN channel by writing C1P=2'b0 (rising edge) in the TMRx_CCTR register.
- Program the capture frequency division of C1IN signal (C1DIV[1: 0]).
- Enable channel 1 input capture (C1EN=1).
- If needed, enable interrupt or DMA request by setting the C1IEN bit in the TMRx_IDEN register or the C1DEN bit in the TMRx_IDEN register.

Timer input XOR function

The timer input pins (TMRx_CH1, TMRx_CH2 and TMRx_CH3) are connected to the channel 1 (selected by setting the C1INSE in the TMRx_CTRL2 register) through an XOR gate.

The XOR gate can be used to connect Hall sensors. For example, connect the three XOR inputs to the three Hall sensors respectively so as to calculate the position and speed of the rotation by analyzing three Hall sensor signals.

PWM input

The PWM input mode applies to channel 1 and channel 2. To enable this mode, map the C1IN and C2IN to the same TMRx_CHx, and configure the CxIFPx of channel 1/2 to trigger slave timer controller reset.

The PWM input mode can be used to measure the period and duty cycle of input signal. The period and duty cycle of channel 1 can be measured as follows:

- Set C1C=2'b01 in the TMRx_CM1 register to set C1IN as C1IFP1;
- Set C1P=1'b0 in the TMRx_CCTRL register to set C1IFP1 rising edge active;
- Set C2C=2'b10 in the TMRx_CM1 register to set C2IN as C1IFP2;
- Set C2P=1'b1 in the TMRx_CCTRL register to set C1IFP2 falling edge active;
- Set STIS=3'b101 in the TMRx_STCTRL register to set C1IFP1 as the slave timer trigger signal;
- Set SMSEL=3'b110 in the TMRx_ STCTRL register to set the slave timer in reset mode;
- Set C1EN=1'b1 and C2EN=1'b1 in the TMRx_CCTRL register to enable channel 1 and channel 2 input capture.

In these configurations, the rising edge of channel 1 input signal triggers capture and saves captured values to the C1DT register, and channel 1 input signal rising edge resets the counter. The falling edge of channel 1 input signal triggers capture and saves captured values to the C2DT register. The period and duty of channel 1 input signal can be calculated through C1DT and C2DT respectively.

Figure 15-83 Example of PWM input mode configuration

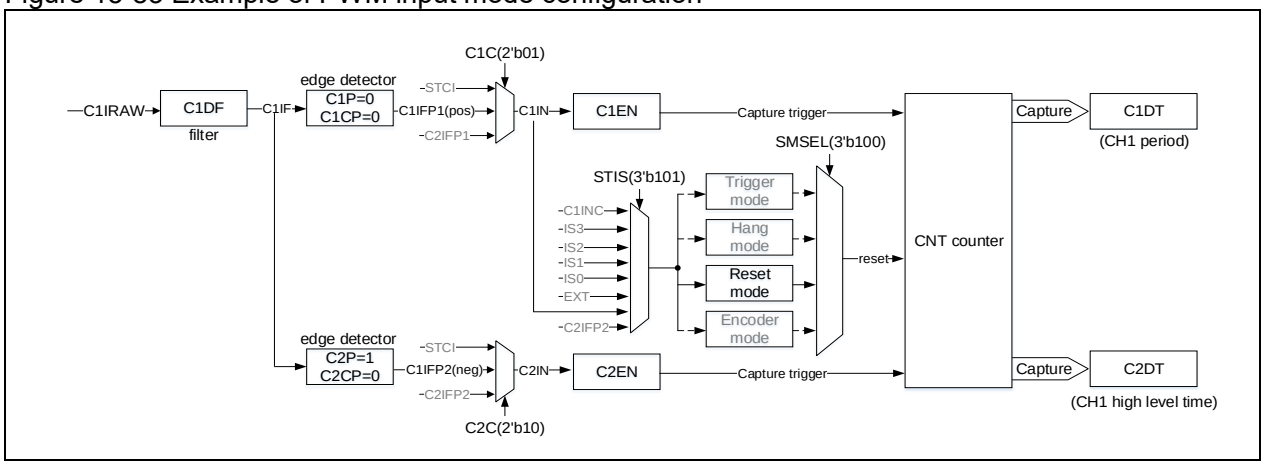
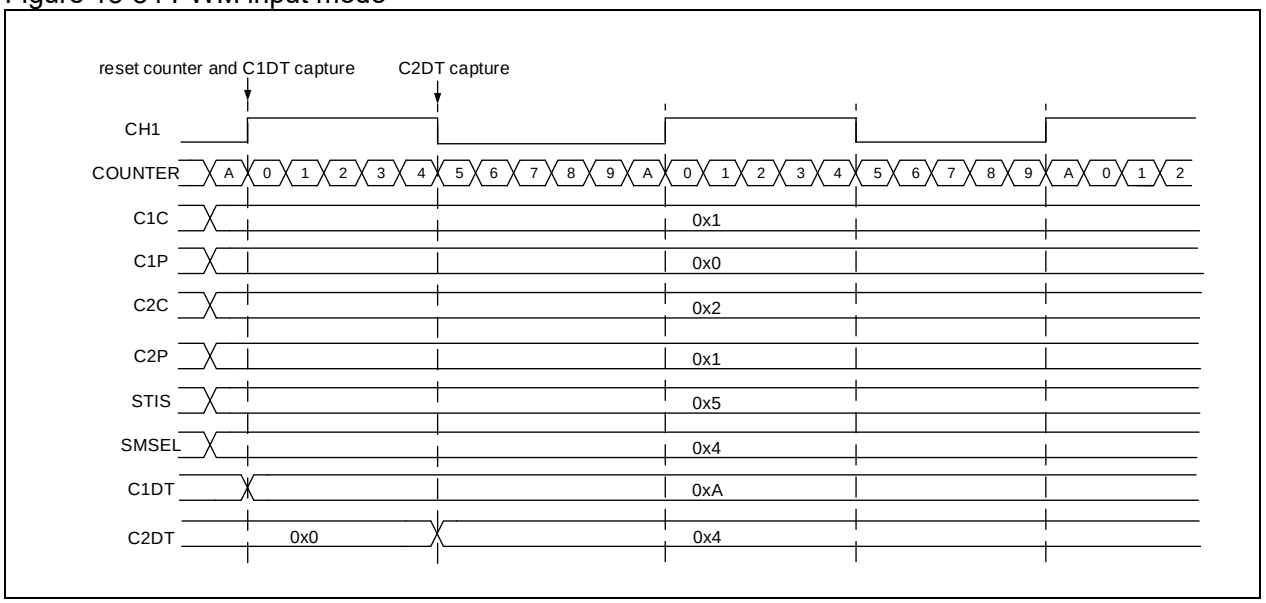


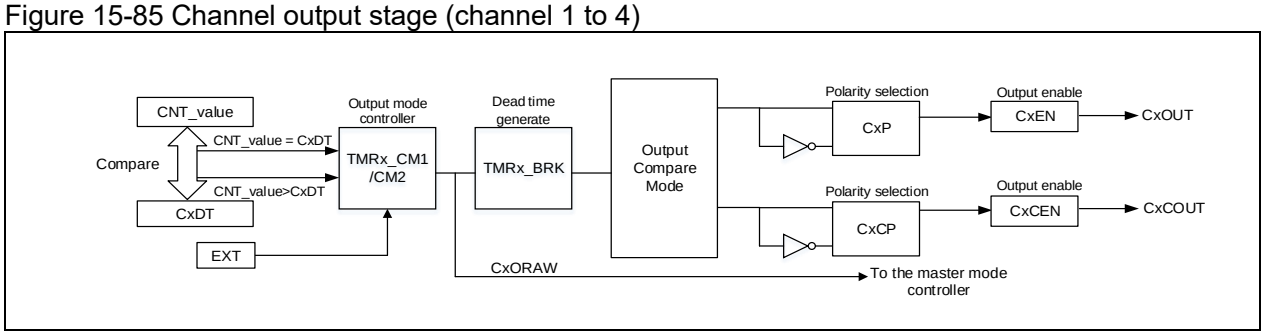
Figure 15-84 PWM input mode



15.4.3.4 TMR output function

The TMR output consists of a comparator and an output controller. It is used to program the period, duty cycle and polarity of the output signal. The advanced-control timer output function varies from one channel to one channel. Channel 5 output is connected to TRGOUT2 only. When TRGOUT2EN=1 in the TMRx_CTRL2 register, TRGOUT2 outputs at the rising edge of C4ORAW or the falling edge of C5ORAW.

Figure 15-85 Channel output stage (channel 1 to 4)



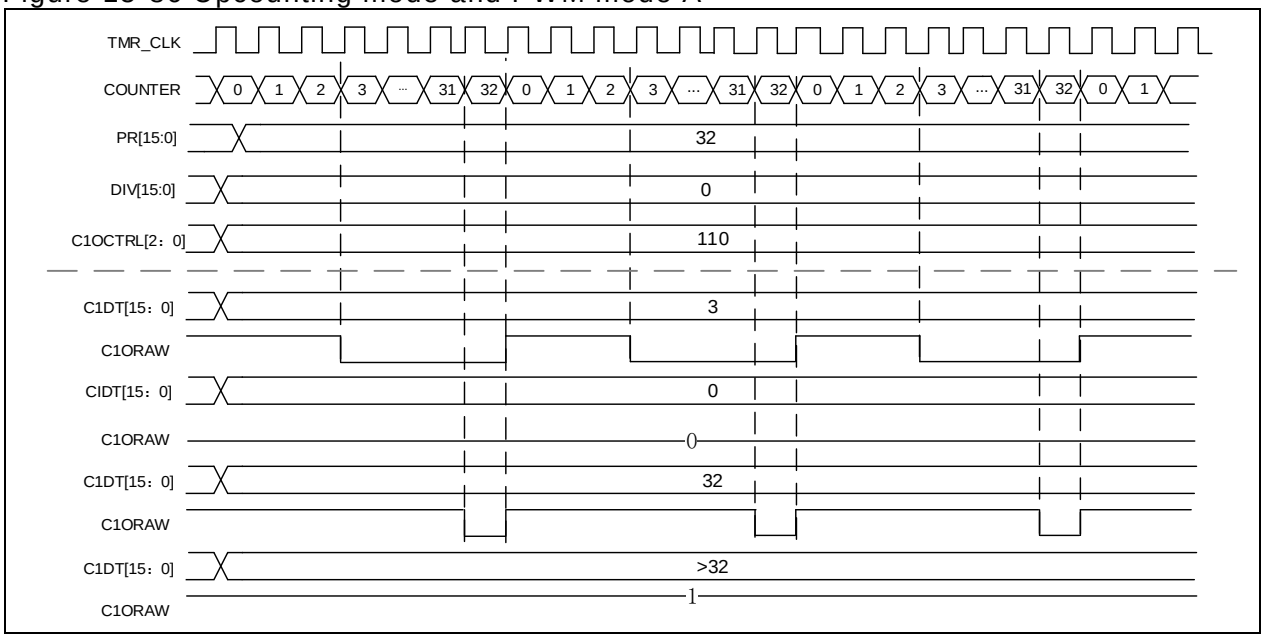
Output mode

Write CxC[1: 0]≠2'b00 to configure the channel as output to implement multiple output modes. In this case, the counter value is compared with the value in the TMRx_CxDT register, and the intermediate signal CxORAW is generated according to the output mode selected by CxOCTRL[2: 0], which is sent to IO after being processed by the output control circuit. The period of the output signal is configured by the TMRx_PR register, while the duty cycle by the TMRx_CxDT register.

Output compare modes include:

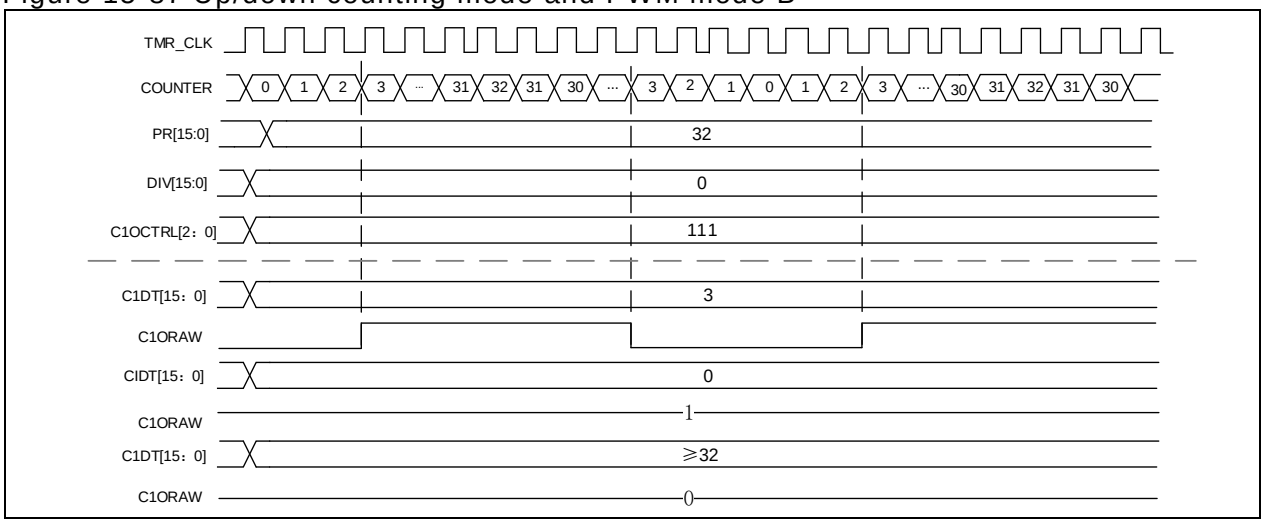
- **PWM mode A:** Set CxOCTRL=3'b110 to enable PWM mode A. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. Figure 15-86 shows the example of PWM mode A in upcounting mode, with PR=0x32 and CxDT with different values.
  To set PWM mode A, the following process is recommended:
  - Set the TMRx_PR register to set PWM period;
  - Set the TMRx_CxDT register to set PWM duty cycle;
  - Set CxOCTRL=3'b110 in the TMRx_CM1/CM2 register to set output mode as PWM mode A;
  - Set the TMRx_DIV register and set the counting frequency;
  - Set the TWCMSEL[1:0] bit in the TMRx_CTRL1 register to set the count mode;
  - Set CxP bit and CxCP bit in the TMRx_CCTRL register to set output polarity;
  - Set CxEN bit and CxCEN bit in the TMRx_CCTRL register to enable channel output;
  - Set the OEN bit in the TMRx_BRK register to enable TMRx output;
  - Set the corresponding GPIO of TMR output channel as the multiplexed mode;
  - Set the TMREN bit in the TMRx_CTRL1 register to enable TMRx counter.

Figure 15-86 Upcounting mode and PWM mode A



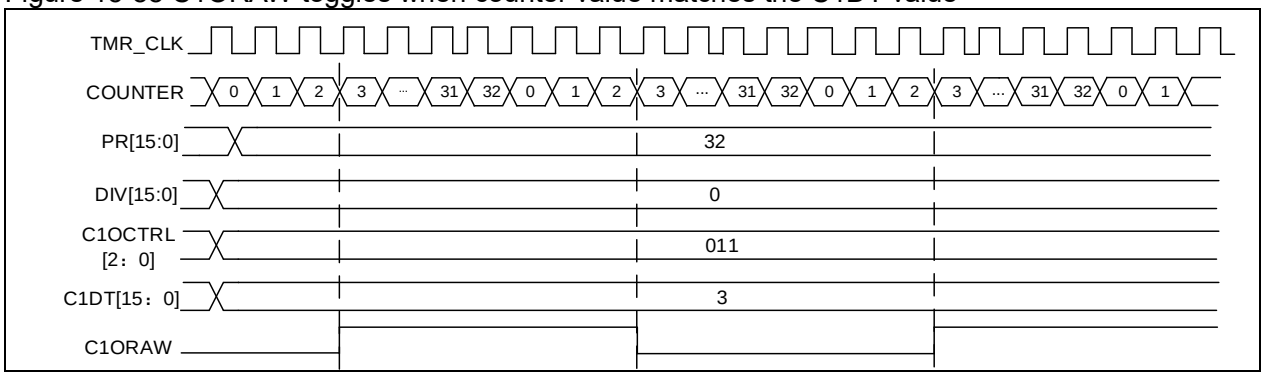
- **PWM mode B:** Set CxOCTRL=3'b111 to enable PWM mode B. In upcounting, when TMRx_C1DT>TMRx_CVAL, C1ORAW outputs low; otherwise, outputs high. In downcounting, when TMRx_C1DT<TMRx_CVAL, C1ORAW outputs high; otherwise, outputs low. Figure 15-87 gives an example of the combination between up/down counting mode and PWM mode B. The output signal behaves when PR=0x32 but CxDT is configured with a different value.

Figure 15-87 Up/down counting mode and PWM mode B



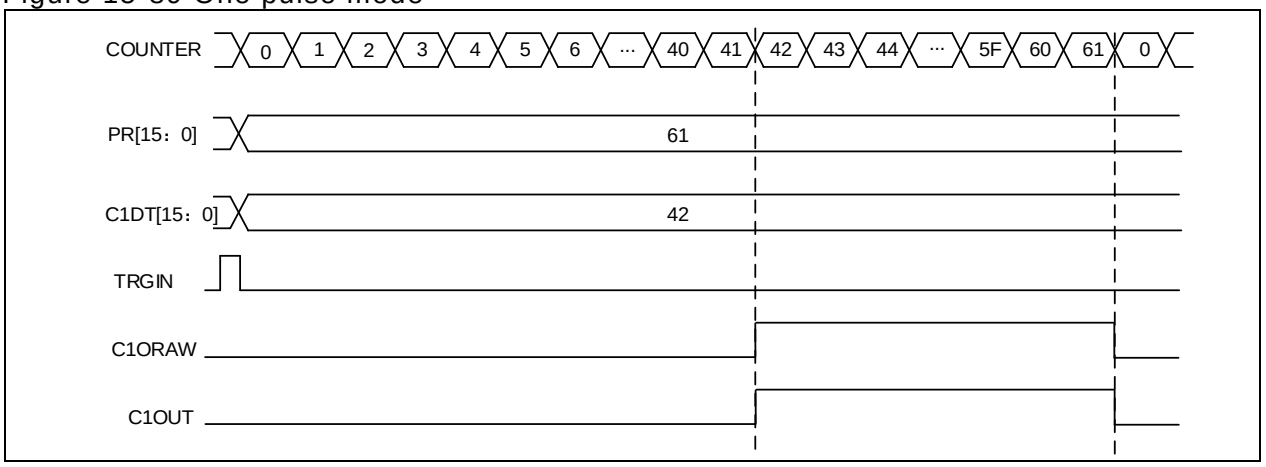
- **Forced output mode:** Set CxOCTRL=3’b100/101 to enable forced output mode. In this case, the CxORAW is forced to be the programmed level, irrespective of the counter value. Despite this, the channel flag bit and DMA request still depend on the compare result.
- **Output compare mode:** Set CxOCTRL=3’b001/010/011 to enable output compare mode. In this case, when the counter value matches the value of the CxDT register, the CxORAW is forced high (CxOCTRL=3’b001), low (CxOCTRL=3’b010) or toggling (CxOCTRL=3’b011). Figure 15-88 gives an example of output compare mode (toggle) with C1DT=0x3. When the counter value is equal to 0x3, C1OUT toggles.

Figure 15-88 C1ORAW toggles when counter value matches the C1DT value



- **One-pulse mode:** This is a particular case of PWM mode. Set OCMEN=1 in the TMRx_CTRL1 register to enable one-pulse mode. In this mode, the comparison match is performed in the current counting period. The TMREN bit is cleared as soon as the current counting is completed. Therefore, only one pulse is output. When configured as in upcounting mode, the configuration must follow the rule: CVAL<CxDT≤PR; in downcounting mode, CVAL>CxDT is required. Figure 15-89 gives an example of the combination between upcounting mode and one-pulse PWM mode B. The counter only counts only one cycle, and the output signal sends only one pulse.

Figure 15-89 One pulse mode



- **Fast output mode:** Set CxOIEN=1 in the TMRx_CM1/CM2 register to enable this mode. If enabled, the CxORAW signal will not change when the counter value matches the CxDT, but at the beginning of the current counting period. In other words, the comparison result is advanced, so the comparison result between the counter value and the TMRx_CxDT register will determine the level of CxORAW in advance.

Master timer event output

When TMR is selected as the master timer, the following signal sources can be selected as TRGOUT signal to output to the slave timer, by setting the PTOS bit in the TMRx_CTRL2 register.

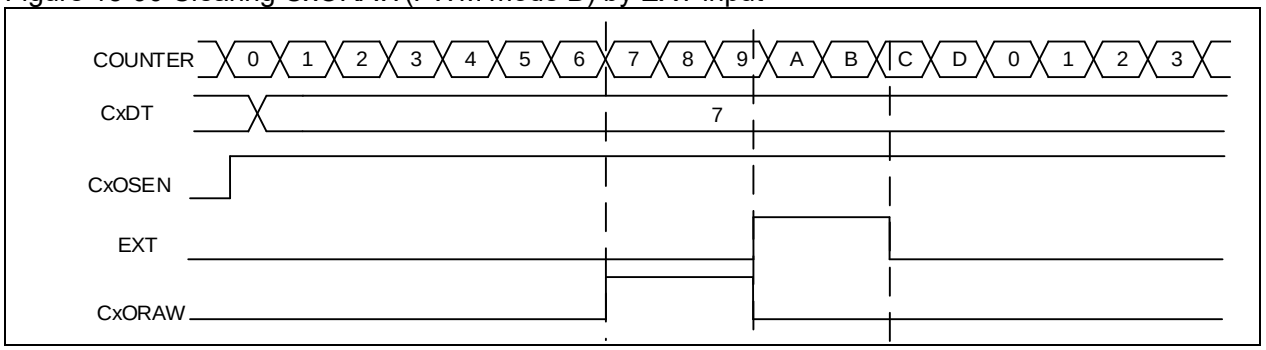
- PTOS=3'b000, TRGOUT outputs software overflow event (OVFSWTR bit in the TMRx_SWEVT register).
- PTOS=3'b001, TRGOUT outputs counter enable signal.
- PTOS=3'b010, TRGOUT outputs counter overflow event.
- PTOS=3'b011, TRGOUT outputs capture and compare event.
- PTOS=3'b100, TRGOUT outputs C1ORAW signal.
- PTOS=3'b101, TRGOUT outputs C2ORAW signal.
- PTOS=3'b110, TRGOUT outputs C3ORAW signal.
- PTOS=3'b111, TRGOUT outputs C4ORAW signal.

CxORAW clear

When the CxOSEN bit is set to 1 in the TMRx_CM1/CM2 register, the CxORAW signal for a given channel is cleared by applying a high level to the EXT input. The CxORAW signal remains unchanged until the next overflow event.

This function can only be used in output capture or PWM modes, and does not work in forced mode. *Figure 15-90* shows the example of clearing CxORAW. When the EXT input is high, the CxORAW signal, which was originally high, is driven low; when the EXT is low, the CxORAW signal outputs the corresponding level according to the comparison result between the counter value and CxDT value.

Figure 15-90 Clearing CxORAW(PWM mode B) by EXT input



Dead-time insertion

The channel 1 to 4 of the advanced-control timers contains a set of reverse channel output. This function is enabled by the CxCEN bit and its polarity is defined by CxCP.

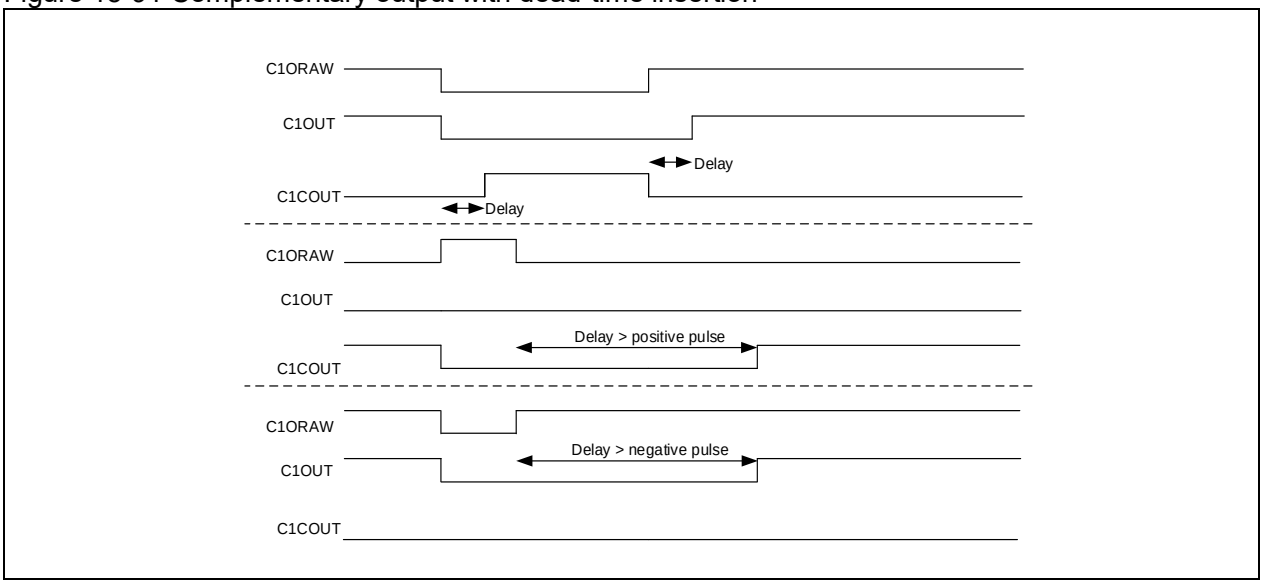
The dead-time is activated when switching to IDLEF state (OEN falling down to 0).

Setting both CxEN and CxCEN bits, and using DTC[7:0] bit to insert dead-time of different durations. After the dead-time insertion, the rising edge of the CxOUT is delayed compared to the rising edge of the reference signal; the rising edge of the CxCOU is delayed compared to the falling edge of the reference signal.

If the delay is greater than the width of the active output, and if C1OUT and C1COUT do not generate corresponding pulses, the dead-time should be less than the width of the active output.

Figure 15-91 gives an example of dead-time insertion when CxP=0, CxCP=0, OEN=1, CxEN=1 and CxCEN=1.

Figure 15-91 Complementary output with dead-time insertion



15.4.3.5 TMR break function

When the brakaaeunction is enabled (BRKEN=1 in the TMRx_BRK register), the CxOUT and CxCOUT are jointly controlled by OEN, FCSODIS, FCSOEN, CxIOS and CxCIOS. But, CxOUT and CxCOUT cannot be set both to active level at the same time. Please refer to 15-15 for more details.

The brake source can be the brake input pin or a clock failure event. The polarity is controlled by the BRKV bit.

When a brake event occurs, there are the following actions:

- The OEN bit is cleared asynchronously, and the channel output state is selected by setting the FCSODIS bit. This function works even if the MCU oscillator is off.
- Once OEN=0, the channel output level is defined by the CxIOS bit. If FCSODIS=0, the timer output is disabled; otherwise, the output enable remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state, that is, 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 active, then the dead-time generator is activated. The CxIOS and CxCIOS bits are used to program the level after dead-time. Even in this case, the CxIOS and CxCIOS cannot be driven to their active level at the same time. It should be note that because of synchronization on OEN, the dead-time duration is usually longer than usual (around 2 clk_tmr clock cycles)
  - If FCSODIS=0, the timer releases the enable output; otherwise, it keeps the enable output; the enable output becomes high as soon as one of the CxEN and CxCEN bits becomes high.
- If the breake interrupt or DMA request is enabled, the breake statue flag is set, and a breake interrupt or DMA request can be generated.
- If AOEN=1, the OEN bit is automatically set again at the next overflow event.

Note: When the brake input is active, the OEN cannot be set, nor the status flag, BRKIF can be cleared.

Figure 15-92 TMR output control

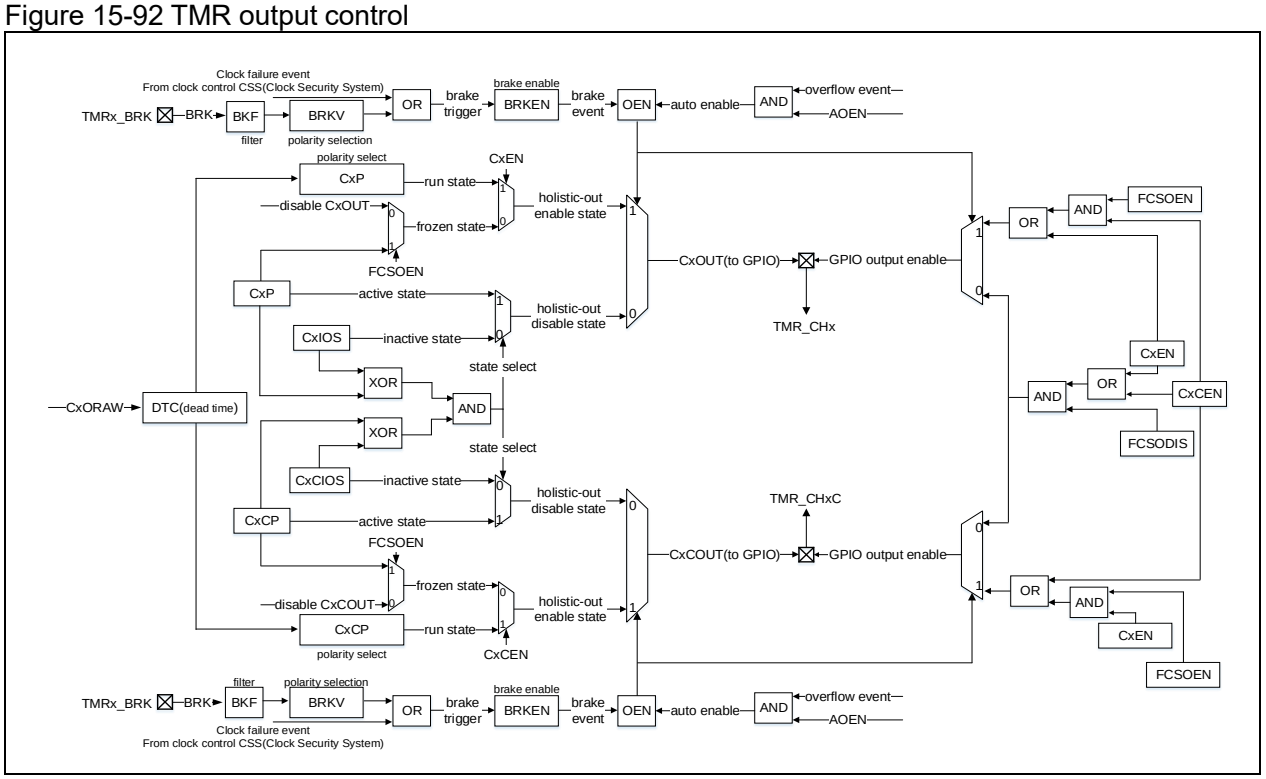
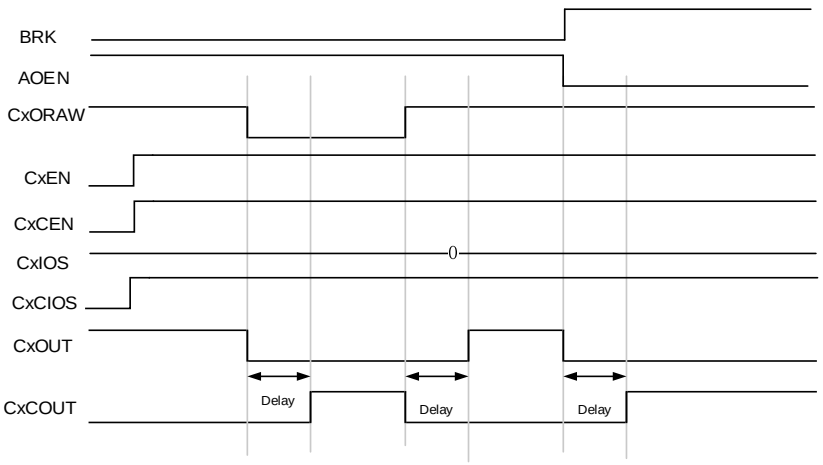


Figure 15-93 Example of TMR breake function



15.4.3.6 TMR synchronization

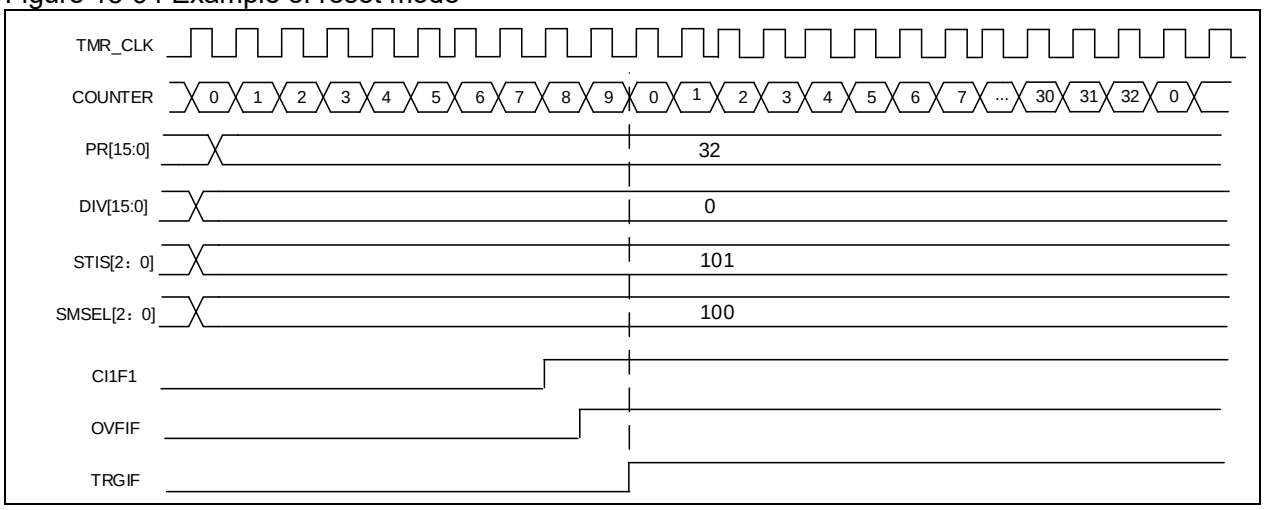
The timers are linked together internally for timer synchronization. Master timer is selected by setting the PTOS[2: 0] bit in the TMRx_CTRL2 register; Slave timer is selected by setting the SMSEL[2: 0] bit in the TMRx_STCTRL register.

Slave modes include:

Slave mode: Reset mode

The counter and its prescaler can be reset by a selected trigger signal. An overflow event can be generated when OVFS=0 in the TMRx_CTRL1 register.

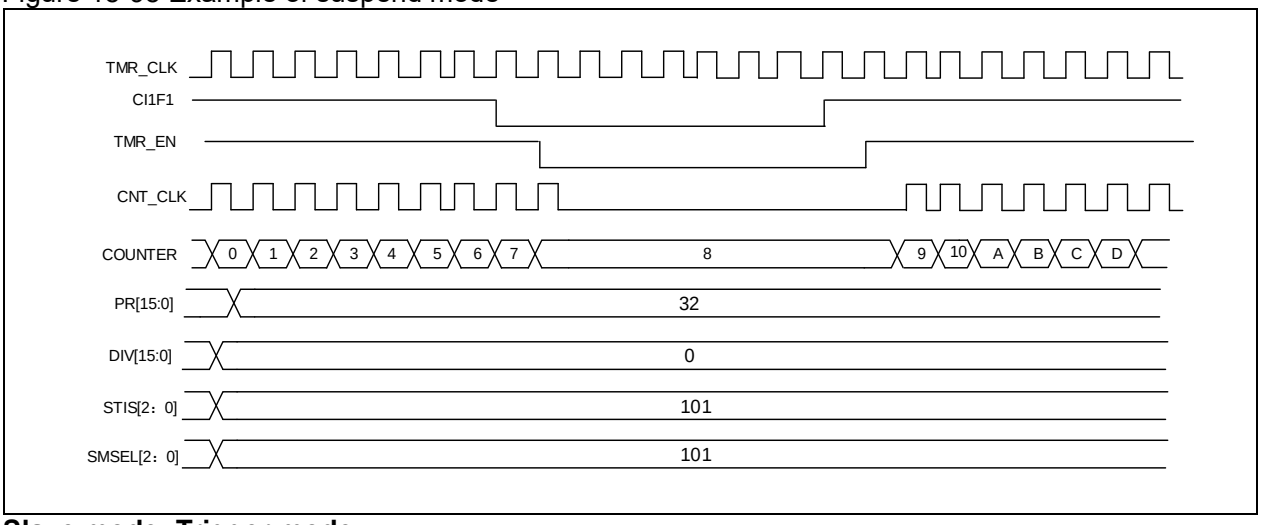
Figure 15-94 Example of reset mode



Slave mode: Suspend mode

In this mode, the counter is controlled by a selected trigger input. The counter starts counting when the trigger input is high and stops as soon as the trigger input is low.

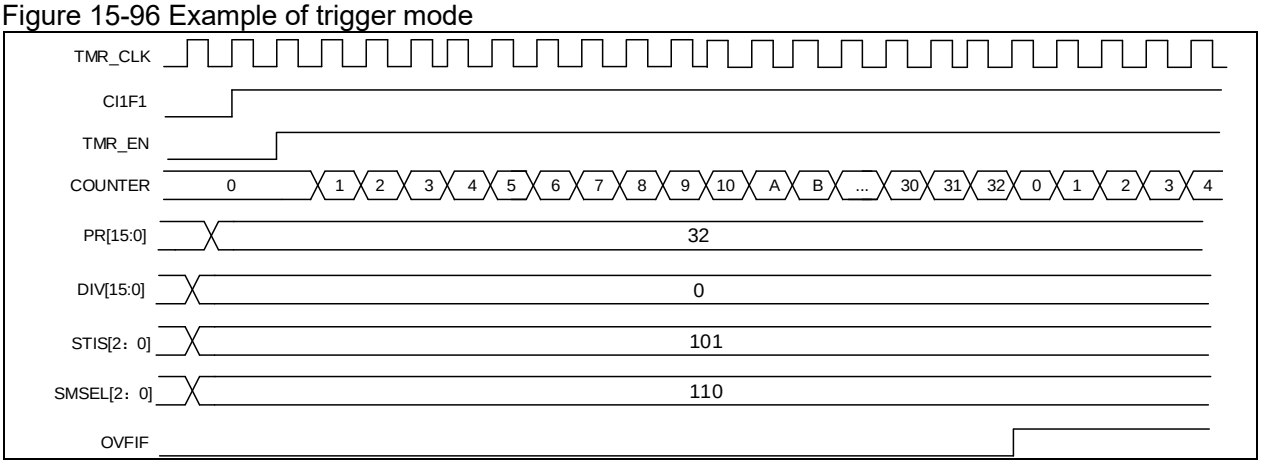
Figure 15-95 Example of suspend mode



Slave mode: Trigger mode

The counter can start counting on the rising edge of a selected trigger input (TMR_EN=1).

Figure 15-96 Example of trigger mode



15.4.3.7 TMR DMA

TMR has the following events for DMA transfer request: overflow event DMA request, trigger event DMA rquest, Hall sensor DMA request and chanel event DMA request. It is possible to enable DMA request by setting the TMRx_IDEN register. Once enabled, upon an event generated, a DMA request is triggered and ouput to the DMA peripheral.

TMR DMA Burst feature

TMR also supports TMR DMA Burst feature. Thanks to this feature, DMA can be triggered to rewrite multiple TMR consecutive registers by enabling a certain DMA request through the TMRx_IDEN register.

As an overflow event to trigger TMR DMA Burst as an example:

- Enable overflow event to trigger DMA request using the OVFDEN bit in the TMRx_IDEN reigster
- Configure Burst transfer times using the DTB bit in the TMRx_DMACTRL register
- Configure the start address of Burst transfer using the ADDR bit in the TMRx_DMACTRL reigster
- Enable counter

After the above-mentioned configurations, here is the whole process of TMR DMA Burst transfer:

Upon an overflow event, the TMR will send a DMA request to the DMA peripheral. Then the DMA writes the data into the TMRx_DMADT register as requested. Inside TMR, the DMADT bit data is written into the start address register of Burst thransfer and an ACK single is sent to TMR. After receiving this ACK, the TMR clears the current DMA request; when the Burst transfer is detected by the TMR not to complete fully, a new overflow event for DMA request is released to the DMA so tthat the DMA writes the requested dta into the TMRx_DMADT again. In this case, inside tmr, the DMADT register data is written into the Burst transfer start address + 0x4 address register, and a new ACK is sent to TMR, and so on, and so on, until the last operation of Burst transfer. When a full Burst transfer is complete, the REQ for overflow event DMA request would not be set any more until the next overflow event.

Note: when using TMR DMA Burst feature, an empty register is prohibited and the TMRx_DMACTRL as well as TMRx_DMADT registers should not be available during the period from the start address to the end address.

15.4.3.8 Debug mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter stops counting by setting the TMRx_PAUSE in the DEBUG module.

15.4.4 TMR1 and TMR8 registers

These peripheral registers must be accessed by words (32 bits).

Table 15-14 TMR1 and TMR8 register map and reset value

Register	Offset	Reset value
TMRx_CTRL1	0x00	0x0000 0000
TMRx_CTRL2	0x04	0x0000 0000
TMRx_STCTRL	0x08	0x0000 0000
TMRx_IDEN	0x0C	0x0000 0000
TMRx_ISTS	0x10	0x0000 0000
TMRx_SWEVT	0x14	0x0000 0000
TMRx_CM1	0x18	0x0000 0000
TMRx_CM2	0x1C	0x0000 0000
TMRx_CCTRL	0x20	0x0000 0000
TMRx_CVAL	0x24	0x0000 0000
TMRx_DIV	0x28	0x0000 0000
TMRx_PR	0x2C	0x0000 FFFF
TMRx_RPR	0x30	0x0000 0000
TMRx_C1DT	0x34	0x0000 0000
TMRx_C2DT	0x38	0x0000 0000
TMRx_C3DT	0x3C	0x0000 0000
TMRx_C4DT	0x40	0x0000 0000
TMRx_BRK	0x44	0x0000 0000

TMRx_DMACTRL	0x48	0x0000 0000
TMRx_DMADT	0x4C	0x0000 0000
TMRx_CM3	0x70	0x0000 0000
TMRx_C5DT	0x74	0x0000 0000

15.4.4.1 TMR1 and TMR8 control register 1 (TMRx_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x00 0000	resd	Kept at its default value.
Bit 9: 8	CLKDIV	0x0	rw	Clock division This field is used to define the division ratio between digital filter sampling frequency (f_{DTS}) and timer clock frequency (f_{CK_INT}). it is also used to set the ratio relationship between dead time base (T_{DTS}) and timer clock period (T_{CK_INT}) 00: No division, $f_{DTS}=f_{CK\_INT}$ 01: Divided by 2, $f_{DTS}=f_{CK\_INT}/2$ 10: Divided by 4, $f_{DTS}=f_{CK\_INT}/4$ 11: Reserved
Bit 7	PRBEN	0x0	rw	Period buffer enable 0: Period buffer is disabled 1: Period buffer is enabled
Bit 6: 5	TWCMSEL	0x0	rw	Two-way counting mode selection 00: One-way counting mode, depending on the OWCDIR bit 01: Two-way up/downcounting mode 1, count up and down alternately, the CxIF bit is set only when the counter counts down 10: Two-way up/downcounting mode 2, count up and down alternately, the CxIF bit is set only when the counter counts up 11: Two-way up/downcounting mode 3, count up and down alternately, the CxIF bit is set when the counter counts up / down
Bit 4	OWCDIR	0x0	rw	One-way count direction 0: Up 1: Down
Bit 3	OCMEN	0x0	rw	One cycle mode enable This bit is use to select whether to stop counting at an update event 0: The counter does not stop at an update event 1: The counter stops at an update event
Bit 2	OVFS	0x0	rw	Overflow event source This bit is used to select overflow event or DMA request sources. 0: Counter overflow, setting the OVFSWTR bit or overflow event generated by slave timer controller 1: Only counter overflow generates an overflow event
Bit 1	OVFEN	0x0	rw	Overflow event enable 0: Enabled 1: Disabled
Bit 0	TMREN	0x0	rw	TMR enable 0: Disabled 1: Enabled

15.4.4.2 TMR1 and TMR8 control register 2 (TMRx_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31	TRGOUT2EN	0x0	rw	TRGOUT2 enable 0: TRGOUT2 remains low 1: TRGOUT2 outputs C4ORAW rising edge or C5ORAW falling edge

Bit 30: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15	C4CIOS	0x0	rw	Channel 4 complementary idle output state
Bit 14	C4IOS	0x0	rw	Channel 4 idle output state
Bit 13	C3CIOS	0x0	rw	Channel 3 complementary idle output state
Bit 12	C3IOS	0x0	rw	Channel 3 idle output state
Bit 11	C2CIOS	0x0	rw	Channel 2 complementary idle output state
Bit 10	C2IOS	0x0	rw	Channel 2 idle output state
Bit 9	C1CIOS	0x0	rw	Channel 1 complementary idle output state OEN = 0 after dead-time: 0: C1OUTL=0 1: C1OUTL=1
Bit 8	C1IOS	0x0	rw	Channel 1 idle output state OEN = 0 after dead-time: 0: C1OUT=0 1: C1OUT=1
Bit 7	C1INSEL	0x0	rw	C1IN selection 0: CH1 pin is connected to C1IRAW input 1: The XOR-ed result of CH1, CH2 and CH3 pins is connected to C1IRAW input
Bit 6: 4	PTOS	0x0	rw	Master TMR output selection This field is used to select the TMRx signal sent to the slave timer. 000: Reset 001: Enable 010: Update 011: Compare pulse (C1DT) 100: C1ORAW signal 101: C2ORAW signal 110: C3ORAW signal 111: C4ORAW signal
Bit 3	DRS	0x0	rw	DMA request source 0: Capture/compare event 1: Overflow event
Bit 2	CCFS	0x0	rw	Channel control bit flash selection This bit only acts on channels that have complementary output. If the channel control bits are buffered: 0: Control bits are updated by setting the HALLSWTR bit 1: Control bits are updated by setting the HALLSWTR bit or a rising edge on TRGIN.
Bit 1	Reserved	0x0	resd	Kept at its default value.
Bit 0	CBCTRL	0x0	rw	Channel buffer control This bit acts on channels that have complementary output. 0: CxEN, CxCEN and CxOCTRL bits are not buffered. 1: CxEN, CxCEN and CxOCTRL bits are buffered. Note: when CBCTRL="1", CxEN, CxCEN and CxOCTRL bits are updated only upon a HALL event.

15.4.4.3 TMR1 and TMR8 slave timer control register (TMRx_STCTRL)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at its default value.
Bit 15	ESP	0x0	rw	External signal polarity 0: High or rising edge 1: Low or falling edge
Bit 14	ECMBEN	0x0	rw	External clock mode B enable This bit is used to enable external clock mode B 0: Disabled 1: Enabled
Bit 13: 12	ESDIV	0x0	rw	External signal divide This field is used to select the frequency division of an external trigger 00: Normal 01: Divided by 2

				10: Divided by 4 11: Divided by 8
Bit 11: 8	ESF	0x0	rw	External signal filter This field is used to filter an external signal. The external signal can be sampled only after it has been generated N times 0000: No filter, sampling by 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	STS	0x0	rw	Subordinate TMR synchronization If enabled, master and slave timer can be synchronized. 0: Disabled 1: Enabled
Bit 6: 4	STIS	0x0	rw	Subordinate TMR input selection This field is used to select the subordinate TMR input. 000: Internal selection 0 (IS0) 001: Internal selection 1 (IS1) 010: Internal selection 2 (IS2) 011: Internal selection 3 (IS3) 100: C1IRAW input detector (C1INC) 101: Filtered input 1 (C1IFP1) 110: Filtered input 2 (C1IFP2) 111: External input (EXT) Please refer to Table 15-2 for more information on ISx for each timer.
Bit 3	Reserved	0x0	resd	Kept at its default value.
Bit 2: 0	SMSEL	0x0	rw	Subordinate TMR mode selection 000: Slave mode is disabled 001: Encoder mode A 010: Encoder mode B 011: Encoder mode C 100: Reset mode — Rising edge of the TRGIN input reinitializes the counter 101: Suspend mode — The counter starts counting when the TRGIN is high 110: Trigger mode — A trigger event is generated at the rising edge of the TRGIN input 111: External clock mode A — Rising edge of the TRGIN input clocks the counter Note: Please refer to count mode section for the details on encoder mode A/B/C.

15.4.4.4 TMR1 and TMR8 DMA/interrupt enable register (TMRx_IDEN)

Bit	Name	Reset value	Type	Description
Bit 31:15	Reserved	0x0 0000	resd	Kept at its default value.
Bit 14	TDEN	0x0	rw	Trigger DMA request enable 0: Disabled 1: Enabled
Bit 13	HALLDE	0x0	rw	HALL DMA request enable 0: Disabled

				1: Enabled
Bit 12	C4DEN	0x0	rw	Channel 4 DMA request enable 0: Disabled 1: Enabled
Bit 11	C3DEN	0x0	rw	Channel 3 DMA request enable 0: Disabled 1: Enabled
Bit 10	C2DEN	0x0	rw	Channel 2 DMA request enable 0: Disabled 1: Enabled
Bit 9	C1DEN	0x0	rw	Channel 1 DMA request enable 0: Disabled 1: Enabled
Bit 8	OVFDEN	0x0	rw	Overflow event DMA request enable 0: Disabled 1: Enabled
Bit 7	BRKIE	0x0	rw	Brake interrupt enable 0: Disabled 1: Enabled
Bit 6	TIEN	0x0	rw	Trigger interrupt enable 0: Disabled 1: Enabled
Bit 5	HALLIEN	0x0	rw	HALL interrupt enable 0: Disabled 1: Enabled
Bit 4	C4IEN	0x0	rw	Channel 4 interrupt enable 0: Disabled 1: Enabled
Bit 3	C3IEN	0x0	rw	Channel 3 interrupt enable 0: Disabled 1: Enabled
Bit 2	C2IEN	0x0	rw	Channel 2 interrupt enable 0: Disabled 1: Enabled
Bit 1	C1IEN	0x0	rw	Channel 1 interrupt enable 0: Disabled 1: Enabled
Bit 0	OVFIEN	0x0	rw	Overflow interrupt enable 0: Disabled 1: Enabled

15.4.4.5 TMR1 and TMR8 interrupt status register (TMRx_ISTS)

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	C5IF	0x0	rw0c	Channel 5 interrupt flag This bit is set by hardware and cleared by software upon compare event. 0: No compare event 1: Compare event
Bit 15: 13	Reserved	0x0	resd	Kept at its default value.
Bit 12	C4RF	0x0	rw0c	Channel 4 recapture flag Please refer to C1RF description.
Bit 11	C3RF	0x0	rw0c	Channel 3 recapture flag Please refer to C1RF description.
Bit 10	C2RF	0x0	rw0c	Channel 2 recapture flag Please refer to C1RF description.
Bit 9	C1RF	0x0	rw0c	Channel 1 recapture flag This bit indicates whether a recapture is detected when C1IF=1. This bit is set by hardware, and cleared by writing “0”. 0: No capture is detected 1: Capture is detected.
Bit 8	Reserved	0x0	resd	Default value

Bit 7	BRKIF	0x0	rw0c	Brake interrupt flag This bit indicates whether the brake input is active or not. It is set by hardware and cleared by writing “0” 0: Inactive level 1: Active level
Bit 6	TRGIF	0x0	rw0c	Trigger interrupt flag This bit is set by hardware on a trigger event. It is cleared by writing “0”. 0: No trigger event occurs 1: Trigger event is generated. Trigger event: an active edge is detected on TRGIN input, or any edge in suspend mode.
Bit 5	HALLIF	0x0	rw0c	HALL interrupt flag This bit is set by hardware on HALL event. It is cleared by writing “0”. 0: No Hall event occurs. 1: Hall event is detected. HALL even: CxEN, CxCEN and CxOCTRL are updated.
Bit 4	C4IF	0x0	rw0c	Channel 4 interrupt flag Please refer to C1IF description.
Bit 3	C3IF	0x0	rw0c	Channel 3 interrupt flag Please refer to C1IF description.
Bit 2	C2IF	0x0	rw0c	Channel 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF	0x0	rw0c	Channel 1 interrupt flag If the channel 1 is configured as input mode: This bit is set by hardware on a capture event. It is cleared by software or read access to the TMRx_C1DT 0: No capture event occurs 1: Capture event is generated If the channel 1 is configured as output mode: This bit is set by hardware on a compare event. It is cleared by software. 0: No compare event occurs 1: Compare event is generated
Bit 0	OVFIF	0x0	rw0c	Overflow interrupt flag This bit is set by hardware on an overflow event. It is cleared by software. 0: No overflow event occurs 1: Overflow event is generated. If OVFEN=0 and OVFS=0 in the TMRx_CTRL1 register: − An overflow event is generated when OVFSWTR= 1 in the TMRx_SWEVT register; − An overflow event is generated when the counter CVAL is reinitialized by a trigger event.

15.4.4.6 TMR1 and TMR8 software event register (TMRx_SWEVT)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	BRKSWTR	0x0	wo	Brake event triggered by software This bit is set by software to generate a brake event. 0: No effect 1: Generate a brake event.
Bit 6	TRGSWTR	0x0	rw	Trigger event triggered by software This bit is set by software to generate a trigger event. 0: No effect 1: Generate a trigger event.
Bit 5	HALLSWTR	0x0	wo	HALL event triggered by software This bit is set by software to generate a HALL event. 0: No effect 1: Generate a HALL event. Note: This bit acts only on channels that have complementary output.
Bit 4	C4SWTR	0x0	wo	Channel 4 event triggered by software

				Please refer to C1SWTR description.
Bit 3	C3SWTR	0x0	wo	Channel 3 event triggered by software Please refer to C1SWTR description.
Bit 2	C2SWTR	0x0	wo	Channel 2 event triggered by software Please refer to C1SWTR description
Bit 1	C1SWTR	0x0	wo	Channel 1 event triggered by software This bit is set by software to generate a channel 1 event. 0: No effect 1: Generate a channel 1 event.
Bit 0	OVFSWTR	0x0	wo	Overflow event triggered by software This bit is set by software to generate an overflow event. 0: No effect 1: Generate an overflow event.

15.4.4.7 TMR1 and TMR8 channel mode register 1 (TMRx_CM1)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	rw	Channel 2 output switch enable
Bit 15	C2OSEN	0x0	rw	Channel 2 output switch enable
Bit 14: 12	C2OCTRL	0x0	rw	Channel 2 output control
Bit 11	C2OBEN	0x0	rw	Channel 2 output buffer enable
Bit 10	C2OIEN	0x0	rw	Channel 2 output enable immediately
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS register.
Bit 7	C1OSEN	0x0	rw	Channel 1 output switch enable 0: C1ORAW is not affected by EXT input. 1: Once a high level is detect on EXT input, clear C1ORAW.
Bit 6: 4	C1OCTRL	0x0	rw	Channel 1 output control This field defines the behavior of the original signal C1ORAW. 000: Disconnected. C1ORAW is disconnected from C1OUT; 001: C1ORAW is high when TMRx_CVAL=TMRx_C1DT 010: C1ORAW is low when TMRx_CVAL=TMRx_C1DT 011: Switch C1ORAW level when TMRx_CVAL=TMRx_C1DT 100: C1ORAW is forced low 101: C1ORAW is forced high. 110: PWM mode A – OWCDIR=0, C1ORAW is high once TMRx_C1DT>TMRx_CVAL, else low; – OWCDIR=1, C1ORAW is low once TMRx_ C1DT <TMRx_CVAL, else high; 111: PWM mode B – OWCDIR=0, C1ORAW is low once TMRx_ C1DT >TMRx_CVAL, else high; – OWCDIR=1, C1ORAW is high once TMRx_ C1DT <TMRx_CVAL, else low.

				Note: In the configurations other than 000', the C1OUT is connected to C1ORAW. The C1OUT output level is not only subject to the changes of C1ORAW, but also the output polarity set by CCTRL.
Bit 3	C1OBEN	0x0	rw	Channel 1 output buffer enable 0: Buffer function of TMRx_C1DT is disabled. The new value written to the TMRx_C1DT takes effect immediately. 1: Buffer function of TMRx_C1DT is enabled. The value to be written to the TMRx_C1DT is stored in the buffer register, and can be sent to the TMRx_C1DT register only on an overflow event.
Bit 2	C1OIEN	0x0	rw	Channel 1 output enable immediately In PWM mode A or B, this bit is used to accelerate the channel 1 output's response to the trigger event. 0: Need to compare the CVAL with C1DT before generating an output 1: No need to compare the CVAL and C1DT. An output is generated immediately when a trigger event occurs.
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Channel 2 digital filter
Bit 15: 12	C2DF	0x0	rw	Channel 2 digital filter
Bit 11: 10	C2IDIV	0x0	rw	Channel 2 input divider
Bit 9: 8	C2C	0x0	rw	Channel 2 configuration This field is used to define the direction of the channel 2 (input or output), and the selection of input pin when C2EN='0': 00: Output 01: Input, C2IN is mapped on C2IFP2 10: Input, C2IN is mapped on C1IFP2 11: Input, C2IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7: 4	C1DF	0x0	rw	Channel 1 digital filter This field defines the digital filter of the channel 1. "N" refers to the number of filtering, meaning that 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: $f_{SMPLING}=f_{DTS}/4$, 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 3: 2	C1IDIV	0x0	rw	Channel 1 input divider This field defines Channel 1 input divider. 00: No divider. An input capture is generated at each active edge. 01: An input compare is generated every 2 active edges 10: An input compare is generated every 4 active edges 11: An input compare is generated every 8 active edges Note: the divider is reset once C1EN='0'
Bit 1: 0	C1C	0x0	rw	Channel 1 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C1EN='0': 00: Output 01: Input, C1IN is mapped on C1IFP1 10: Input, C1IN is mapped on C2IFP1 11: Input, C1IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

15.4.4.8 TMR1 and TMR8 channel mode register 2 (TMRx_CM2)

The channel can be used in input (capture mode) or output (compare mode). The direction of a channel is defined by the corresponding CxC bits. All the other bits of this register have different functions in input and output modes. The CxOx describes its function in output mode when the channel is in output mode, while the CxIx describes its function in output mode when the channel is in input mode. Attention must be given to the fact that the same bit can have different functions in input mode and output mode.

Output compare mode:

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Channel 4 output switch enable
Bit 15	C4OSEN	0x0	rw	Channel 4 output switch enable
Bit 14: 12	C4OCTRL	0x0	rw	Channel 4 output control
Bit 11	C4OBEN	0x0	rw	Channel 4 output buffer enable
Bit 10	C4OIEN	0x0	rw	Channel 4 output enable immediately
Bit 9: 8	C4C	0x0	rw	Channel 4 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN='0': 00: Output 01: Input, C4IN is mapped on C4IFP4 10: Input, C4IN is mapped on C3IFP4 11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7	C3OSEN	0x0	rw	Channel 3 output switch enable
Bit 6: 4	C3OCTRL	0x0	rw	Channel 3 output control
Bit 3	C3OBEN	0x0	rw	Channel 3 output buffer enable
Bit 2	C3OIEN	0x0	rw	Channel 3 output enable immediately
Bit 1: 0	C3C	0x0	rw	Channel 3 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN='0': 00: Output 01: Input, C3IN is mapped on C3IFP3 10: Input, C3IN is mapped on C4IFP3 11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

Input capture mode:

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Channel 4 digital filter
Bit 15: 12	C4DF	0x0	rw	Channel 4 digital filter
Bit 11: 10	C4IDIV	0x0	rw	Channel 4 input divider

Bit 9: 8	C4C	0x0	rw	Channel 4 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C4EN='0': 00: Output 01: Input, C4IN is mapped on C4IFP4 10: Input, C4IN is mapped on C3IFP4 11: Input, C4IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.
Bit 7: 4	C3DF	0x0	rw	Channel 3 digital filter
Bit 3: 2	C3IDIV	0x0	rw	Channel 3 input divider
Bit 1:0	C3C	0x0	rw	Channel 3 configuration This field is used to define the direction of the channel 1 (input or output), and the selection of input pin when C3EN='0': 00: Output 01: Input, C3IN is mapped on C3IFP3 10: Input, C3IN is mapped on C4IFP3 11: Input, C3IN is mapped on STCI. This mode works only when the internal trigger input is selected by STIS.

15.4.4.9 TMR1 and TMR8 channel control register (TMRx_CCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept its default value.
Bit 15	C4CP	0x0	rw	Channel 4 complementary polarity Please refer to C1CP description.
Bit 14	C4CEN	0x0	rw	Channel 4 complementary enable Please refer to C1CEN description.
Bit 13	C4P	0x0	rw	Channel 4 polarity Please refer to C1P description.
Bit 12	C4EN	0x0	rw	Channel 4 enable Please refer to C1EN description.
Bit 11	C3CP	0x0	rw	Channel 3 complementary polarity Please refer to C1CP description.
Bit 10	C3CEN	0x0	rw	Channel 3 complementary enable Please refer to C1CEN description.
Bit 9	C3P	0x0	rw	Channel 3 polarity Please refer to C1P description.
Bit 8	C3EN	0x0	rw	Channel 3 enable Please refer to C1EN description.
Bit 7	C2CP	0x0	rw	Channel 2 complementary polarity Please refer to C1CP description.
Bit 6	C2CEN	0x0	rw	Channel 2 complementary enable Please refer to C1CEN description.
Bit 5	C2P	0x0	rw	Channel 2 polarity Please refer to C1P description.
Bit 4	C2EN	0x0	rw	Channel 2 enable Please refer to C1EN description.
Bit 3	C1CP	0x0	rw	Channel 1 complementary polarity 0: C1COUT is active high. 1: C1COUT is active low.
Bit 2	C1CEN	0x0	rw	Channel 1 complementary enable 0: Output is disabled. 1: Output is enabled.
Bit 1	C1P	0x0	rw	Channel 1 polarity When the channel 1 is configured as output mode: 0: C1OUT is active high 1: C1OUT is active low When the channel 1 is configured as input mode: 00: C1IN active edge is on its rising edge. When used as external trigger, C1IN is not inverted. 01: C1IN active edge is on its falling edge. When used as external trigger, C1IN is inverted.

				10: Reserved 11:C1IN active edge on rising and falling edge. When used as external trigger, C1IN is not inverted.
Bit0	C1EN	0x0	rw	Channel 1 enable 0: Input or output is disabled 1: Input or output is enabled

Table 15-15 Complementary output channel CxOUT and CxCOUT control bits with break feature

Control bit					Output state (1)	
OEN bit	FCSODIS bit	FCSOEN bit	CxEN bit	CxCEN bit	CxOUT output state	CxCOUT output state
1	X	0	0	0	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	0	1	Output disabled (no driven by the timer) CxOUT=0, Cx_EN=0	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		0	1	0	CxORAW+ polarity CxOUT= CxORAW xor CxP, Cx_EN=1	Output disabled (no driven by the timer) CxCOUT=0, CxCEN=0
		0	1	1	CxORAW+polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
		1	0	0	Output disabled (no driven by the timer) CxOUT=CxP, Cx_EN=0	Output disabled (no driven by the timer) CxCOUT=CxCP, CxCEN=0
		1	0	1	Off-state (Output enabled with inactive level) CxOUT=CxP, Cx_EN=1	CxORAW + polarity, CxCOUT= CxORAW xor CxCP, CxCEN=1
		1	1	0	CxORAW + polarity, CxOUT= CxORAW xor CxP, Cx_EN=1	Off-state (Output enabled with inactive level) CxCOUT=CxCP, CxCEN=1
		1	1	1	CxORAW+ polarity+dead-time, Cx_EN=1	CxORAW inverted+polarity+dead-time, CxCEN=1
0	0	X	0	0	Output disabled (the corresponding IO disconnected from timer, IO floating) Asynchronously: CxOUT=CxP, Cx_EN=0, CxCOUT=CxCP, CxCEN=0; If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	
	0		0	1		
	0		1	0		
	0		1	1		
	1		0	0	CxEN=CxCEN=0: output disabled (the corresponding IO disconnected from timer, IO floating) Other: Off-state (the corresponding channel outputs inactive level) Asynchronously: CxOUT =CxP, Cx_EN=1, CxCOUT=CxCP, CxCEN=1; If the clock is present: after a dead-time, CxOUT=CxIOS, CxCOUT=CxCIOS, assuming that CxIOS and CxCIOS do not correspond to CxOUT and CxCOUT active level.	
	1		0	1		
	1		1	0		
	1		1	1		

Note: If the two outputs of a channel are not used (CxEN = CxCEN = 0), CxIOS, CxCIOS, CxP and CxCP must be cleared.

Note: The state of the external I/O pins connected to the complementary CxOUT and CxCOUT channels depends on the CxOUT and CxCOUT channel state and the GPIO and the IOMUX registers.

15.4.4.10 TMR1 and TMR8 counter value (TMRx_CVAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	CVAL	0x0000	rw	Counter value

15.4.4.11 TMR1 and TMR8 division value (TMRx_DIV)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DIV	0x0000	rw	Divider value The counter clock frequency $f_{CK\_CNT} = f_{TMR\_CLK}$ / (DIV[15: 0]+1). The value of this register is transferred to the actual prescaler register when an overflow event occurs.

15.4.4.12 TMR1 and TMR8 period register (TMRx_PR)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	PR	0xFFFF	rw	Period value This defines the period value of the TMRx counter. The timer stops working when the period value is 0.

15.4.4.13 TMR1 and TMR8 repetition period register (TMRx_RPR)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	RPR	0x0000	rw	Repetition of period value This field is used to reduce the generation rate of overflow events. An overflow event is generated when the repetition counter reaches 0.

15.4.4.14 TMR1 and TMR8 channel 1 data register (TMRx_C1DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	C1DT	0x0000	rw	Channel 1 data register When the channel 1 is configured as input mode: The C1DT is the CVAL value stored by the last channel 1 input event (C1IN). When the channel 1 is configured as output mode: C1DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C1OBEN bit, and the corresponding output is generated on C1OUT as configured.

15.4.4.15 TMR1 and TMR8 channel 2 data register (TMRx_C2DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	C2DT	0x0000	rw	Channel 2 data register When the channel 2 is configured as input mode: The C2DT is the CVAL value stored by the last channel 2 input event (C2IN). When the channel 2 is configured as output mode: C2DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C2OBEN bit, and the corresponding output is generated on C2OUT as configured.

15.4.4.16 TMR1 and TMR8 channel 3 data register (TMRx_C3DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	C3DT	0x0000	rw	Channel 3 data register When the channel 3 is configured as input mode: The C3DT is the CVAL value stored by the last channel 3 input event (C3IN). When the channel 3 is configured as output mode: C3DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C3OBEN bit, and the corresponding output is generated on C3OUT as configured.

15.4.4.17 TMR1 and TMR8 channel 4 data register (TMRx_C4DT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	C3DT	0x0000	rw	Channel 4 data register When the channel 4 is configured as input mode: The C4DT is the CVAL value stored by the last channel 4 input event (C4IN). When the channel 3 is configured as output mode: C4DT is the value to be compared with the CVAL value. Whether the written value takes effective immediately depends on the C4OBEN bit, and the corresponding output is generated on C4OUT as configured.

15.4.4.18 TMR1 and TMR8 brake register (TMRx_BRK)

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x0000	resd	Kept at default value.
Bit 19: 16	BRKF	0x0	rw	Brake input filter This field is used to set the filter for brake input. “N” refers to the number of filtering, meaning that 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: $f_{SMPLING}=f_{DTS}/4$, 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 Note: if there is no way to guarantee clean break input, it is recommended to enable break input filtering.
Bit 15	OEN	0x0	rw	Output enable This bit acts on the channels as output. It is used to enable CxOUT and CxCOUT outputs.

				0: Disabled 1: Enabled
Bit 14	AOEN	0x0	rw	Automatic output enable OEN is set automatically at an overflow event. 0: Disabled 1: Enabled
Bit 13	BRKV	0x0	rw	Brake input validity This bit is used to select the active level of a brake input. 0: Brake input is active low. 1: Brake input is active high.
Bit 12	BRKEN	0x0	rw	Brake enable This bit is used to enable brake input. 0: Brake input is disabled. 1: Brake input is enabled.
Bit 11	FCSOEN	0x0	rw	Frozen channel status when holistic output enable This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=1. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output inactive level.
Bit 10	FCSODIS	0x0	rw	Frozen channel status when holistic output disable This bit acts on the channels that have complementary output. It is used to set the channel state when the timer is inactive and OEN=0. 0: CxOUT/CxCOUT outputs are disabled. 1: CxOUT/CxCOUT outputs are enabled. Output idle level.
Bit 9: 8	WPC	0x0	rw	Write protection configuration This field is used to enable write protection. 00: Write protection is OFF. 01: Write protection level 3, and the following bits are write protected: TMRx_BRK: BRKF, DTC, BRKEN, BRKV and AOEN TMRx_CTRL2: CxIOS and CxCIOS 10: Write protection level 2. The following bits and all bits in level 3 are write protected: TMRx_CCTRL: CxP and CxCP TMRx_BRK: FCSODIS and FCSOEN 11: Write protection level 1. The following bits and all bits in level 2 are write protected: TMRx_CMx: C2OCTRL and C2OBEN Note: Once WPC>0, its content remains frozen until the next system reset.
Bit 7: 0	DTC	0x00	rw	Dead-time configuration This field defines the duration of the dead-time insertion. The 3-bit MSB of DTC[7: 0] is used for function selection: 0xx: DT = DTC [7: 0] * TDTS 10x: DT = (64+ DTC [5: 0]) * TDTS * 2 110: DT = (32+ DTC [4: 0]) * TDTS * 8 111: DT = (32+ DTC [4: 0]) * TDTS * 16

Note: Based on lock configuration, AOEN, BRKV, BRKEN, FCSODIS, FCSOEN and DTC[7:0] can all be write protected. Thus it is necessary to configure write protection when writing to the TMRx_BRK register for the first time.

15.4.4.19 TMR1 and TMR8 DMA control register (TMRx_DMACTRL)

Bit	Name	Reset value	Type	Description
Bit 31:13	Reserved	0x0 0000	resd	Kept at its default value.
Bit 12:8	DTB	0x00	rw	DMA transfer bytes This field defines the number of DMA transfers: 00000: 1 byte 00001: 2 bytes 00010: 3 bytes 00011: 4 bytes .... 10000: 17 bytes 10001: 18 bytes
Bit 7:5	Reserved	0x0	resd	Kept at its default value.
Bit 4: 0	ADDR	0x00	rw	DMA transfer address offset ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register: 00000: TMRx_CTRL1 00001: TMRx_CTRL2 00010: TMRx_STCTRL .....

15.4.4.20 TMR1 and TMR8 DMA data register (TMRx_DMADT)

Bit	Name	Reset value	Type	Description
Bit 31:16	Reserved	0x0000	resd	Kept at default value.
Bit 15: 0	DMADT	0x0000	rw	DMA data register A write/read operation to the DMADT register accesses any TMR register located at the following address: TMRx peripheral address + ADDR*4 to TMRx peripheral address + ADDR*4 + DTB*4

15.4.4.21 TMR1 and TMR8 channel mode register 3 (TMRx_ CM3)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x00 0000	resd	Kept at its default value.
Bit 7	C5OSEN	0x0	rw	Channel 5 output switch enable
Bit 6: 4	C5OCTRL	0x0	rw	Channel 5 output control
Bit 3	C5OBEN	0x0	rw	Channel 5 output buffer enable
Bit 2	C5OIEN	0x0	rw	Channel 5 output immediately enable
Bit 1: 0	Reserved	0x0	resd	Kept at its default value.

15.4.4.22 TMR1 and TMR8 channel 5 data register (TMRx_C5DT)

Bit	Register	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	C5DT	0x0000	rw	Channel 5 data register C5DT holds the value that is to be compared with the CVAL. Whether the written data will takes effect immediately depends on the C5OBEN bit, and the corresponding output generates on the C5OUT bit.

16 Window watchdog timer (WWDT)

16.1 WWDT introduction

The window watchdog downcounter must be reloaded in a limited time window to prevent the watchdog circuits from generating a system reset. The window watch dog is used to detect the occurrence of system malfunctions.

The window watchdog timer is clocked by a divided APB1_CLK. The precision of the APB1_CLK enables the window watchdog to take accurate control of the limited window.

16.2 WWDT main features

- 7-bit downcounter
- If the watchdog is enabled, a system reset is generated when the value of the downcounter is less than 0x40 or when the downcounter is reloaded outside the window.
- The downcounter can be reloaded by enabling the counter interrupt.

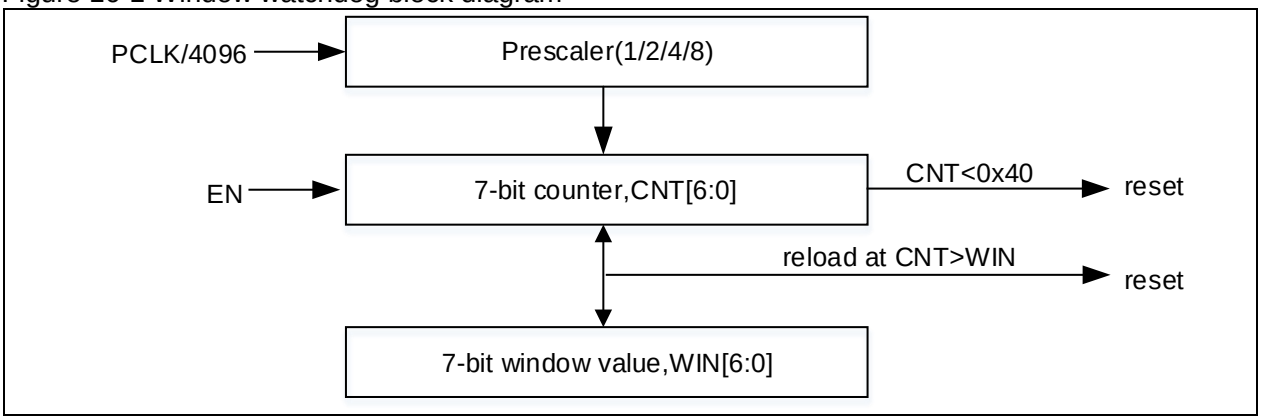
16.3 WWDT functional overview

If the watchdog is enabled, a system reset is generated at the following conditions:

When the 7-bit downcounter scrolls from 0x40 to 0x3F;

When the counter is reloaded while the 7-bit downcounter is greater than the value programmed in the window register.

Figure 16-1 Window watchdog block diagram



To prevent system reset, the counter must be reloaded only when its value is less than the value stored in the window register and greater than 0x40.

The WWDT counter is clocked by a divided APB1_CLK, with the division factor being defined by the DIV[1: 0] bit in the WWDT_CFG register. The counter value determines the maximum counter period before the watchdog generates a reset. The WIN[6: 0] bit can be used to configure the window value.

WWDT offers reload counter interrupt feature. If enabled, the WWDT will set the RLDF flag when the counter value reaches 0x40h, and an interrupt is generated accordingly. The interrupt service routine (ISTS) can be used to reload the counter to prevent a system reset. Note that if CNT[6]=0, setting the WWDTEN bit will generate a system reset, so the CNT[6] bit must be always set (CNT[6]=1) while writing to the WWDT_CTRL register to prevent the occurrence of an immediate reset once the window watchdog is enabled.

The formula to calculate the window watchdog time out:

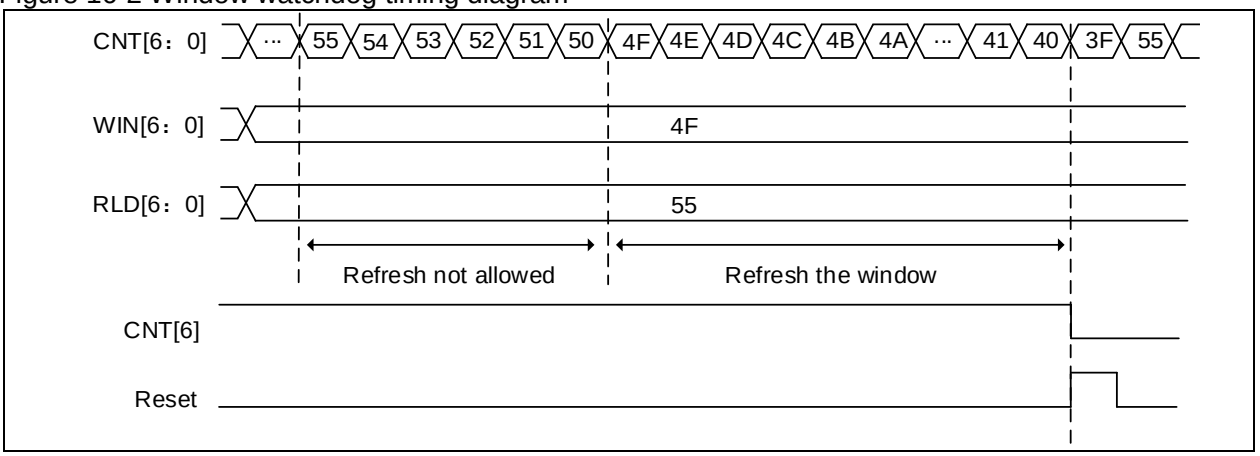
$$T_{WWDT} = T_{PCLK1} \times 4096 \times 2^{DIV[1:0]} \times (CNT[5:0] + 1); (ms)$$

Where: T_{PCLK1} refers to APB1 clock period, in ms.

Table 16-1 Minimum and maximum timeout value when PCLK1=72 MHz

Prescaler	Min. Timeout value	Max. Timeout value
0	56.5μs	3.64ms
1	113.5μs	7.28ms
2	227.5μs	14.56ms
3	455μs	29.12ms

Figure 16-2 Window watchdog timing diagram



16.4 WWDT registers

These peripheral registers must be accessed by words (32 bits).

Table 16-2 WWDT register map and reset value

Register	Offset	Reset value
WWDT_CTRL	0x00	0x0000 007F
WWDT_CFG	0x04	0x0000 007F
WWDT_STS	0x08	0x0000 0000

16.4.1 Control register (WWDT_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value.
Bit 7	WWDTEN	0x0	rw1s	Window watchdog enable 0: Disabled 1: Enabled This bit is set by software, but can be cleared only after reset.
Bit 6: 0	CNT	0x7F	rw	Downcounter When the counter counts down to 0x3F, a reset is generated.

16.4.2 Configuration register (WWDT_CFG)

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x000000	resd	Kept at its default value.
Bit 9	RLDIEN	0x0	rw1s	Reload counter interrupt 0: Disabled 1: Enabled
Bit 8: 7	DIV	0x0	rw	Clock division value 00: PCLK1 divided by 4096 01: PCLK1 divided by 8192 10: PCLK1 divided by 16384 11: PCLK1 divided by 32768
Bit 6: 0	WIN	0x7F	rw	Window value If the counter is reloaded while its value is greater than the window register value, a reset is generated. The counter must be reloaded between 0x40 and WIN[6: 0].

16.4.3 Status register (WWDT_STS)

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	RLDF	0x0	rw0c	Reload counter interrupt flag This flag is set when the downcounter reaches 0x40. 'This bit is set by hardware and cleared by software.

17 Watchdog timer (WDT)

17.1 WDT introduction

The WDT is driven by a dedicated low-speed clock (LICK). Due to the lower clock accuracy of LICK, the WDT is best suited to the applications that have lower timing accuracy and can run independently outside the main application.

17.2 WDT main features

- 12-bit downcounter
- The counter is clocked by LICK (can work in Stop and Standby modes)
- The counter can be configured to stop counting either in Deepsleep or Standby mode
- A system reset is generated under the following circumstances:
  - When the counter value is decremented to 0
  - When the counter is reloaded outside the window

17.3 WDT functional overview

WDT enable:

Both software and hardware operations can be used to enable WDT. In other words, the WDT can be enabled by writing 0xCCCC to the WDT_CMD register; or when the user enables the hardware watchdog through user system data area, the WDT will be automatically enabled after power-on reset.

WDT reset:

When the counter value of the WDT counts down to 0, a WDT reset be generated. Thus the WDT_CMD register must be written with the value 0xAAAA at regular intervals to reload the counter value to avoid the WDT reset.

WDT write-protected:

The WDT_DIV and WDT_RLD registers are write-protected. Writing the value 0x5555 to the WDT_CMD register will unlock write protection. The update status of these two registers are indicated by the DIVF and RLDF bits in the WDT_STS register. If a different value is written to the WDT_CMD register, these two registers will be re-protected. Writing the value 0xAAAA to the WDT_CMD register also enables write protection.

WDT clock:

The WDT counter is clocked by the LICK. The LICK is an internal RC clock with a typical value of 40kHz, with its range falling between 30kHz and 60kHz. The timeout period is also within a certain range, so a margin should be taken into account when configuring timeout period. The LICK can be calibrated to obtain the WDT timeout with a relatively accuracy. For more details, please refer to section 4.1.1.

WDT low power counting mode:

WDT can work in Sleep, Deepsleep and Standby modes. It is possible to stop counting in Deepsleep and Standby modes by setting the nWDT_DEPSLP and nWDT_STDBY bits in the User System Data area.

If the counter is disabled, it will stop decrementing as soon as the Deepsleep and Standby modes are entered. This means that the WDT would not perform a system reset in both low power modes. After waking up from these two modes, it continues downcounting from the value at the time of the entry of these modes.

Figure 17-1 WDT block diagram

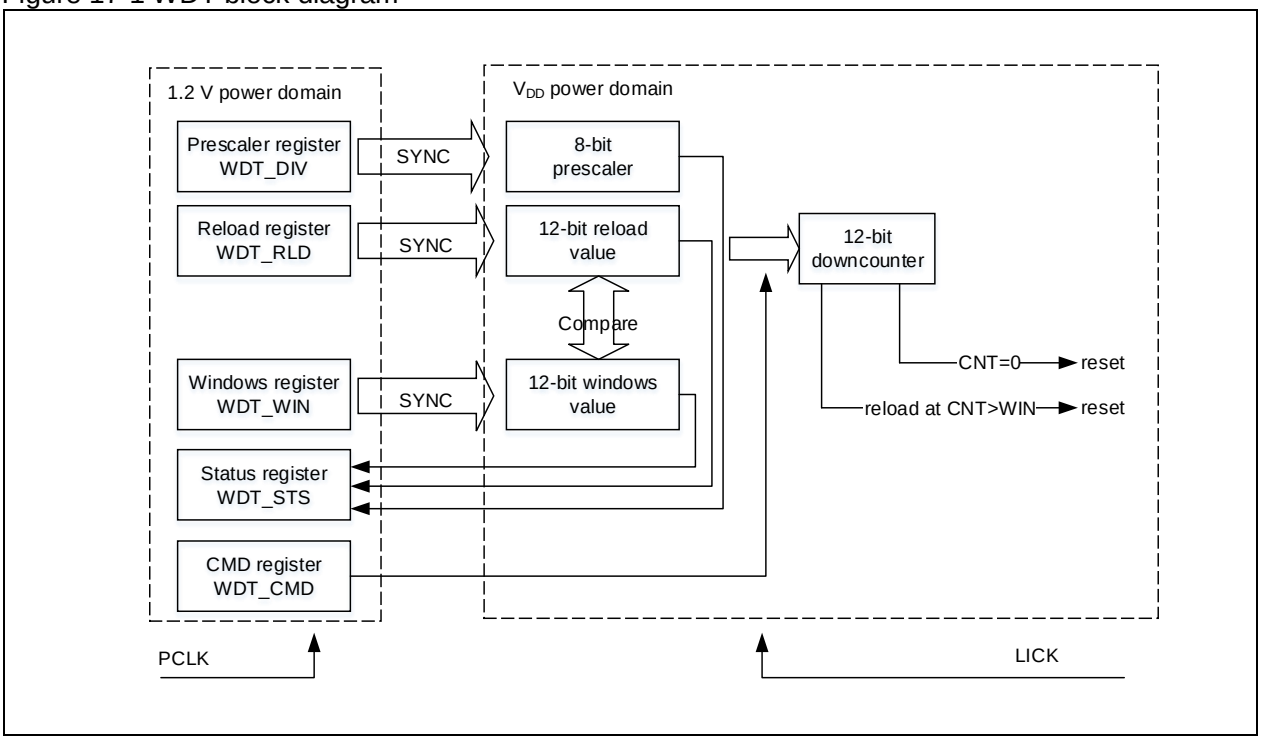


Table 17-1 WDT timeout period (LICK=40kHz)

Prescaler divider	DIV[2: 0] bits	Min.timeout (ms) RLD[11: 0] = 0x000	Max. timeout (ms) RLD[11: 0] = 0xFFF
/4	0	0.1	409.6
/8	1	0.2	819.2
/16	2	0.4	1638.4
/32	3	0.8	3276.8
/64	4	1.6	6553.6
/128	5	3.2	13107.2
/256	(6 or 7)	6.4	26214.4

17.4 Debug mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the WDT counter stops counting by setting the WDT_PAUSE in the DEBUG module. Refer to Chapter 31.2 for more information.

17.5 WDT registers

These peripheral registers must be accessed by words (32 bits).

Table 17-2 WDT register and reset value

Register	Offset	Reset value
WDT_CMD	0x00	0x0000 0000
WDT_DIV	0x04	0x0000 0000
WDT_RLD	0x08	0x0000 0FFF
WDT_STS	0x0C	0x0000 0000
WDT_WIN	0x10	0x0000 0FFF

17.5.1 Command register (WDT_CMD)

(Reset in Standby mode)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	CMD	0x0000	wo	Command register 0xAAAA: Reload counter 0x5555: Unlock write-protected WDT_DIV and WDT_RLD 0xCCCC: Enable WDT. If the hardware watchdog has been enabled, ignore this operation.

17.5.2 Divider register (WDT_DIV)

(Not reset in Standby mode).

Bit	Name	Reset value	Type	Description
Bit 31: 3	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 2: 0	DIV	0x0	rw	Clock division value 000: LICK divided by 4 001: LICK divided by 8 010: LICK divided by 16 011: LICK divided by 32 100: LICK divided by 64 101: LICK divided by 128 110: LICK divided by 256 111: LICK divided by 256 The write protection must be unlocked in order to enable write access to the register. The register can be read only when DIVF=0.

17.5.3 Reload register (WDT_RLD)

(Not reset in Standby mode)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value.
Bit 11: 0	RLD	0xFFF	rw	Reload value The write protection must be unlocked in order to enable write access to the register. The register can be read only when RLDF=0.

17.5.4 Status register (WDT_STS)

(Reset in Standby mode)

Bit	Name	Reset value	Type	Description
Bit 31: 3	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 2	WINF	0x0	ro	Window value update complete flag 0: Window value update complete 1: Window value update is in process. The WDT_WIN register can be written only when RLDF=0
Bit 1	RLDF	0x0	ro	Reload value update complete flag 0: Reload value update complete 1: Reload value update is in process. The reload register WDT_RLD can be written only when RLDF=0
Bit 0	DIVF	0x0	ro	Division value update complete flag 0: Division value update complete 1: Division value update is in process. The divider register WDT_DIV can be written only when DIVF=0

17.5.5 Window register (WDT_WIN)

(Not reset in Standby mode)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x000000	resd	Kept at its default value.
Bit 11 : 0	WIN	0xFFF	ro	Window value When the counter value is greater than the window value, the reload counter will perform a reset. The reload counter value falls between 0 and the window value.

18 Enhanced real-time clock (ERTC)

18.1 ERTC introduction

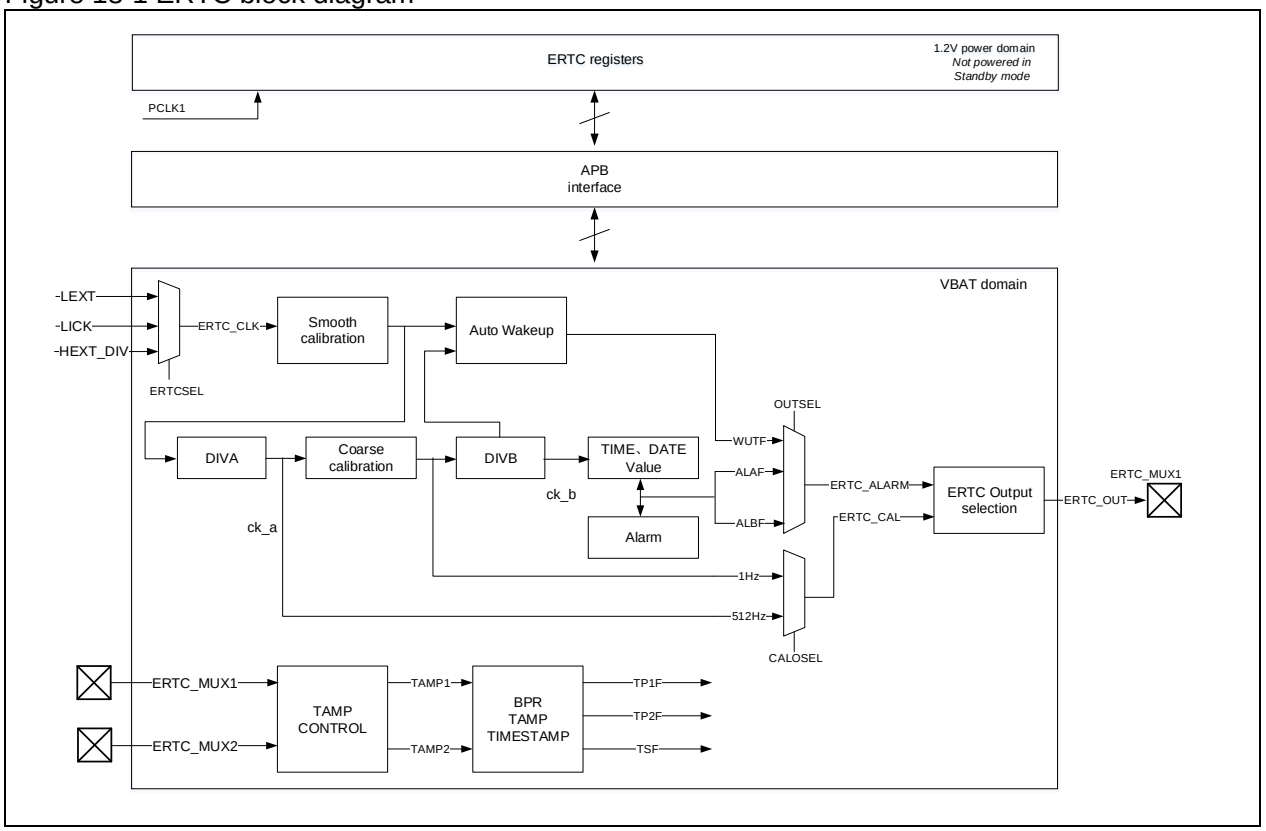
The real-time clock provides a calendar clock function. The time and date can be modified by modifying the ERTC_TIME and ERTC_DATE register.

The RTC module is in battery powered domain, which means that it keeps running and free from the influence of system reset and VDD power off as long as VBAT is powered.

18.2 ERTC main features

- Real-time calendar (automatic processing of month days, including 28 (February in a common year), 29 (February in a leap year), 30 (a lunar month of 30 days) and 31 (a solar month of 31 days), where the current register being a multiple of 4 indicates a leap year), two programmable alarms
- Periodic auto-wakeup
- Reference clock detection
- Two programmable tamper detection, supporting time stamp feature
- Supports fine calibration and coarse calibration
- 20 x battery powered registers
- 5 x interrupts: alarm A, alarm B, periodic auto-wakeup, tamper detection and time stamp
- Multiplexed function output, calibration clock output, alarm event or wakeup event
- Multiplexed function input, reference clock input, two-way tamper detection and time stamp

Figure 18-1 ERTC block diagram



18.3 ERTC function overview

18.3.1 ERTC clock

ERTC clock source (ERTC_CLK) is selected via clock controller from a LEXT, LICK, and a divided HEXT clock (By setting the ERTCSEL[1: 0] in the CRM_BPDC register). The HEXT divider value is configured through the ERTC_DIV[4: 0] bit in the CRM_CFG register.

The ERTC embeds two dividers: A and B, programmed by the DIVA[6: 0] and DIVB[14: 0] respectively. It is recommended that the DIVA is configured to a higher value in order to minimum power consumption. After being divided by prescaler A and B, the ERTC_CLK generates ck_a and ck_b clocks, respectively. The ck_a is used for subsecond update, while the ck_b is used for calendar update and periodic auto wakeup. The clock frequency of ck_a and ck_b can be obtained from the following equation:

$$F_{ck\_a} = \frac{f_{ERTC\_CLK}}{DIVA + 1}$$

$$F_{ck\_b} = \frac{f_{ERTC\_CLK}}{(DIVB + 1) \times (DIVA + 1)}$$

To obtain ck_b with frequency of 1 Hz, DIVA=127, DIVB=255, and 32.768 kHz LEXT should be used. This ck_b is then used for calendar update.

Note: To use a divided HEXT by the ERTC_CLK, it is necessary to configure a HEXT divider value before switching a clock source to the HEXT.

18.3.2 ERTC initialization

ERTC register write protection

After a power-on reset, all ERTC registers are write protected. Such protection mechanism is not affected by the system reset. Write access to the ERTC registers (except the ERTC_STS[14: 8], ERTC_TAMP and ERTC_BPRx registers) can be enabled by unlocking it.

To unlick the write protection of ERTC registers, the steps below should be respected:

1. Enable power interface clock by setting PWCEN=1 in the CRM_APB1EN register.
2. Unlock write protection of the battery powered domain by setting BPWEN=1 in the PWC_CTRL register.
3. Write 0xCA and 0x53 to the ERTC_WP register in sequence. Writing an incorrect key will activate the write protection again.

Table 18-1 lists the ERTC registers that can be configured only after the write protection is unlocked and when the initialization mode is entered.

Table 18-1 RTC register map and reset values

Register	ERTC_WP enabled	Whether to enter initialization mode	Others
ERTC_TIME	Y	Y	-
ERTC_DATE	Y	Y	-
ERTC_CTRL	Y	Bit 7, bit 6 and 4 only	-
ERTC_STS	Y, except [14: 8]	-	-
ERTC_DIV	Y	Y	-
ERTC_WAT	Y	N	Configurable when WATWF=1
ERTC_CCAL	Y	Y	-
ERTC_ALA	Y	N	Configurable when ALAWF =1
ERTC_ALB	Y	N	Configurable when ALAWF =1
ERTC_WP	-	-	-
ERTC_SBS	-	-	-
ERTC_TADJ	Y	N	Configurable when TADJF=0
ERTC_TSTM	-	-	-
ERTC_TSDT	-	-	-
ERTC_TSSBS	-	-	-
ERTC_SCAL	Y	N	Configurable when CALUPDF=0
ERTC_TAMP	N	N	-
ERTC_ALASBS	Y	N	Configurable when ALAWF =1
ERTC_ALBSBS	Y	N	Configurable when ALBWF=1
ERTC_BPRx	N	N	-

Clock and calendar initialization

After the register write protection is unlocked, follow the procedure below for clock and calendar initialization:

1. Set the IMEN bit to enter initialization mode
2. Wait until the initialization flag INITF bit is set
3. Configure DIVB and DIVA.
4. Configure the clock and calendar values.
5. Leave the initialization mode by clearing the IMEN bit. Wait until the UPDF bit is set, indicating the completion of the calendar update. The calendar starts counting.

The ERTC also allows the fine-tuning for daylight saving time and clock.

Daylight saving time feature: It is used to increase (ADD1H=1) or decrease (DEC1H=1) one hour in the calendar, without completing the whole initialization process.

Clock calibration: It is used for the fine calibration of the current clock. If only DECSBS[14: 0] is configured, the value will be added to the DIVB counter and a clock latency will be generated. If only ADD1S bit is set, the current clock will increase by one second. If both DECSBS[14: 0] and ADDIS bit are configured, the clock will increase by a fraction of a second.

Time latency (ADD1S=0): DECSBS/(DIVB+1)

Time advance (ADD1S=1): 1-(DECSBS/(DIVB+1)

Note: To avoid subsecond overflow, SBS[15]=0 must be asserted before setting the ERTC_TADJ register. Reference clock detection and coarse digital calibration cannot be used at the same time. Thus when RCDEN=1, coarse digital calibration is not supported.

Reading the calendar

The ERTC offers two different ways to read the calendar, that is, synchronous read (DREN=0) and asynchronous read (DREN=1).

In the case of DREN=0, the clock and calendar values can be obtained by reading a synchronous shadow register via the PCLK1. The UPDF bit is set each time the shadow register is synchronized with the ERTC calendar value located in the battery powered domain. The synchronization is performed every two ERTC_CLK. The shadow register is reset by a system reset. To ensure consistency between the 3 values (ERTC_SBS, ERTC_TIME and ERTC_DATE registers), reading lower-order registers will lock the values in the higher-order registers until the ERTC_DATE register is read. For example, reading the ERTC_SBS register will lock the values in the ERTC_TIME and ERTC_DATE registers.

In the case of DREN=1, the ERTC will perform direct read access to the ERTC clock and calendar located in the battery powered domain with the PCLK1, avoiding the occurrence of errors caused by time synchronization. In this mode, the UPDF flag is cleared by hardware. To ensure the data is correct when reading clock and calendar, the software must read the clock and calendar registers twice, and compare the results of two read operations. If the result is not aligned, read again until that the results of two read accesses are consistent. Besides, it is also possible to compare the least significant bits of the two read operations to determine their consistency.

Note: In Standby and Deepsleep modes, the current calendar values are not copied into the shadow registers. When waking up from these two modes, UPDF=0 must be asserted, and then wait until UPDF=1, to ensure that the latest calendar value can be read. In synchronous read (DREN=0) mode, the frequency of the PCLK1 must be at least seven times the ERTC_CLK frequency. In asynchronous read (DREN=1), an additional APB cycle is required to complete the read operations of the calendar register.

Alarm clock initialization

The ERTC contains two programmable alarm clocks: alarm clock A and alarm clock B, and their respective interrupts.

The alarm clock value is programmed with the ERTC_ALASBS/ERTC_ALA (ERTC_ALBSBS/ERTC_ALB). When the programmed alarm value matches the calendar value, an alarm event is generated if an alarm clock is enabled. The MASKx bit can be used to selectively mask calendar fields. The calendar fields, which are masked, are not allocated with an alarm clock.

To configure the alarm clocks, the following steps should be respected:

1. Disable alarm clock A or alarm clock B (by setting ALAEN=0 or ALBEN=0)
2. Wait until the ALAWF or ALBWF bit is set to enable write access to the alarm clock A or B
3. Configure alarm clock A or B registers (ERTC_ALA/ERTC_ALASBS and ERTC_ALB/ERTC_ALBSBS)
4. Enable alarm clock A or B by setting ALAEN=1 or ALBEN=1

Note: If MASK1=0 in the ERTC_ALA or ERTC_ALB, the alarm clock can work normally only when the DIVB value is at least equal to 3.

18.3.3 Periodic automatic wakeup

Periodic automatic wakeup unit is used to wake up ERTC from low power consumption modes automatically. The period is programmed with the VAL[15: 0] bi (When WATCLK[2]=1, it is extended to 17 bits, and the wakeup counter value is VAL+216). When the wakeup counter value drops from the VAL to zero, the WATF bit is set, and a wakeup event is generated, with the wakeup counter being reloaded with the VAL value. An interrupt is also generated if a periodic wakeup interrupt is enabled.

The WATCLK[2: 0] bit can be used to select a wakeup timer clock, including ERTC_CLK/16, ERTC_CLK/8, ERTC_CLK/4, ERTC_CLK/2 and ck_b (usually 1Hz). The cooperation between wakeup timer clocks and wakeup counter values enable users to adjust the wakeup period freely.

To enable a periodic automatic wakeup, the following steps should be respected:

1. Disable a periodic automatic wakeup by setting WATEN=0
2. Wait until WATWF=1 to enable write access to the wakeup reload timer and WATCLK[2: 0]
3. Configure the wakeup timer counter value and wakeup timer through VAL[15: 0] and WATCLK[2: 0] bits
4. Enable a timer by setting WATEN=1

Note: A wakeup timer is not affected by a system reset and low power consumption modes (Sleep, Deepsleep and Standby modes).

Note: In DEBUG mode, if ERTC_CLK is selected as the wakeup timer, the counter for periodic wakeup works normally..

18.3.4 ERTC calibration

Two calibration methods are available: coarse and fine calibration. But the two calibration methods cannot be used together.

Coarse digital calibration:

Coarse digital calibration can be used to advance or delay the calendar updates by increasing or decreasing ck_a cycles.

When positive calibration is enabled (CALDIR=0), 2 ck_a cycles are added every minute (around 15360 ck_a cycles) for the first 2xCALVAL minutes of the 64-minute cycle. This causes the calendar to be updated sooner.

When negative calibration is enabled (CALDIR=1), 1 ck_a cycle is ignored every minute (around ck_a cycles) for the first 2xCALVAL minutes of the 64-minute cycle. This causes the calendar to be updated later.

Note: Coarse digital calibration can work correctly only when the DIVA is 6 or above.

Smooth digital calibration:

Smooth digital calibration has a higher and well-distributed performance than the coarse digital calibration. The calibration is performed by increasing or decreasing ERTC_CLK in an evenly manner.

The smooth digital calibration period is around 2^{20} ERTC_CLK (32 seconds) when the ERTC_CLK is 32.768 kHz. The DEC[8: 0] bit specifies the number of pulses to be masked during the 2^{20} ERTC_CLK cycles. A maximum of 511 pulses can be removed. When the ADD is set, 512 pulses can be inserted during the 2^{20} ERTC_CLK cycles. When DEC[8: 0] and ADD are sued together, a deviation ranging from -511 to +512 ERTC_CLK cycles can be added during the 2^{20} ERTC_CLK cycles.

The effective calibrated frequency (F_{SCAL}):

$$F_{SCAL} = F_{ERTC\_CLK} \times [\, 1 + \frac{ADD \times 512 - DEC}{2^{20} + DEC - ADD \times 512} \,]$$

When the divider A is less than 3, the calibration operates as if ADD was equal to 0. The divider B value should be reduced so that each second is accelerated by 8 ERTC_CLK cycles, which means that 256 ERTC_CLK cycles are added every 32 seconds. When DEC[8: 0] and ADD are sued together, a deviation ranging from -255 to +256 ERTC_CLK cycles can be added during the 2^{20} ERTC_CLK cycles.

At this point, the effective calibrated frequency (F_{SCAL})

$$F_{SCAL} = F_{ERTC\_CLK} \times [\, 1 + \frac{256 - DEC}{2^{20} + DEC - 256} \,]$$

It is also possible to select 8 or 16-second digital calibration period through the CAL8 and CAL16 bits. The 8-second period takes priority over 16-second. In other words, when both 8-second and 16-second are enabled, 8-second calibration period prevails.

The CALUPDF flag in the ERTC indicates the calibration status. During the configuration of ERTC_SCAL registers, the CALUPDF bit is set, indicating that the calibration value is being updated; Once the calibration value is successfully applied, this bit is cleared automatically, indicating the completion of the calibration value update.

18.3.5 Reference clock detection

The calendar update can be synchronized (not used in low-power modes) to a reference clock (usually the mains 50 or 60 Hz) with a higher precision. This reference clock is used to calibrate the precision of the calendar update frequency (1 Hz)

When it is enabled, the reference clock edge detection is performed during the first 7 ck_a periods around each of the calendar updates. When detected, the edge is used to update calendar values, and 3 ck_a periods are used for subsequent calendar updates. Each time the reference clock edge is detected, the divider A value is forced to reload, making the reference clock and the 1 Hz clock are aligned. If the 1 Hz clock has a slight shift, a more accurate reference clock can be used to fine-tune the 1 Hz clock so that it is aligned with the reference clock. If no reference clock edge is detected, the calendar is updated based on ERTC's original clock source.

Note: Once the reference clock detection is enabled, the DIVA and DIVB must be kept at its respective reset value (0x7F and 0xFF respectively). The clock synchronization cannot be used in conjunction with the coarse digital calibration.

18.3.6 Time stamp function

When time stamp event is detected on the tamper pin (valid edge is detected), the current calendar value will be stored to the time stamp register.

When a time stamp event occurs, the time stamp flag bit (TSF) in the ERTC_STS register will be set. If a new time stamp event is detected when time stamp flag (TSF) is already set, then the time stamp overflow flag (TSOF) will be set, but the time stamp registers will remain the result of the last event. By setting the TSIEN bit, an interrupt can be generated when a time stamp event occurs.

Usage of time stamp:

1. How to enable time stamp when a valid edge is detected on a tamper pin
- Select a time stamp in by setting the TSPIN bit
- Select a rising edge or falling edge to trigger time stamp by setting the TSEDG bit
- Enable time stamp by setting TSEN=1

2. How to save time stamp on a tamper event
- Configure tamper detection registers
- Enable tamper detection time stamp by setting TPTSEN=1

Note: The TSF bit will be set after two ck_a cycles following a time stamp event. It is suggested that users poll TSOF bit when the TSF is set.

18.3.7 Tamper detection

The ERTC has two tamper detection modes: TAMP1 and TAMP2. They can be configured as a level detection with filter or edge detection respectively. TAMP1 can select either ERTC_MUX or ERTC_MUX2 through the TSPIN bit, while the TAMP2 can only select ERTC_MUX2.

The TP1F or TP2F will be set when a valid tamper event is detected. An interrupt will also be generated if a tamper detection interrupt is enabled. If the TPTSEN bit is already set, a time stamp event will be generated accordingly. Once a tamper event occurs, the battery powered registers will be reset so as to ensure data security in the battery powered domain.

How to configure edge detection

1. Select edge detection by setting TPFLT=00, and select a valid edge (either TP1EDG or TP2EDG)
2. According to your needs, configure whether to activate a time stamp on a tamer event (TPTSEN=1)
3. According to your needs, enable a tamper detection interrupt (TPIEN=1)

4. To use TAMP1 mode, select ERTC_MUX1 or ERTC_MUX2 (through the TP1PIN bit), and enable TAMP1 (setting TP1EN=1); To use TAMP2 mode, just need enable TAMP2 (TP2EN=1)

How to configure level detection with filter

1. Select level detection with filter, and valid level sampling times (TPFLT≠00)
2. Select tamper detection valid level (through TP1EDG or TP2EDG)
3. Select tamper detection sampling frequency (through the TPFREQ bit)
4. According to your needs, enable tamper detection pull-up (setting TPPU=1). When TPPU=1 is asserted, tamper detection pre-charge time must be configured through the TPPR bit
5. According to your needs, configure whether to activate a time stamp on a tamper event (TPTSEN=1)
6. According to your needs, enable a tamper interrupt (TPIEN=1)
7. To use TAMP1 mode, select ERTC_MUX1 or ERTC_MUX2 (through the TP1PIN bit), and enable TAMP1 (setting TP1EN=1); To use TAMP2 mode, just need enable TAMP2 (TP2EN=1)

In the case of edge detection mode, the following two points deserve our attention:

1. If a rising edge is configured to enable tamper detection, and the tamper detection pin turns to high level before tamper detection is enabled, then a tamper event will be detected right after the tamper detection is enabled;
2. If a falling edge is configured to enable tamper detection, and the tamper detection pin turns to low level before tamper detection is enabled, then a tamper event will be detected right after the tamper detection is enabled;

Note: Tamper detection is still active even when the VDD power is OFF.

Note: In Standby mode, TAMP2 (PA0 pin) can not use pre-charge function.

18.3.8 Multiplexed function output

ERTC provides a set of multiplexed function output for the following events:

1. Clocks calibrated (OUTSEL=0 and CALOEN=1)
- Output 512Hz (CALOSEL=0)
- Output 1Hz (CALOSEL=1)

2. Alarm clock A (OUTSEL=1)
3. Alarm clock B (OUTSEL=2)
4. Wakeup events (OUTSEL=3)

When alarm clock or wakeup events are selected (OUTSEL≠0), it is possible to select output type (open-drain or push-pull) with the OUTTYPE bit, and output polarity with the OUTP bit.

18.3.9 ERTC wakeup

ERTC can be woken up by alarm clocks, periodic auto wakeup feature, time stamps or tamper events. To enable an ERTC interrupt, configure as follows:

1. Configure the EXINT line corresponding to ERTC interrupts as an interrupt mode and enable it, and select a rising edge
2. Enable a NVIC channel corresponding to ERTC interrupts
3. Enable an ERTC interrupt

Table 18-2 lists the ERTC clock sources, events and interrupts that are able to wakeup low-power modes.

Table 18-2 ERTC low-power mode wakeup

Clock sources	Events	Wake up Sleep	Wake up Deepsleep	Wakeup Standby
HEXT	Alarm clock A	√	×	×
	Alarm clock B	√	×	×
	Periodic automatic wakeup	√	×	×
	Time stamp	√	×	×
	Tamper event	√	×	×
LICK	Alarm clock A	√	√	√
	Alarm clock B	√	√	√
	Periodic automatic wakeup	√	√	√
	Time stamp	√	√	√
	Tamper event	√	√	√
LEXT	Alarm clock A	√	√	√
	Alarm clock B	√	√	√
	Periodic automatic wakeup	√	√	√
	Time stamp	√	√	√
	Tamper event	√	√	√

Table 18-3 Interrupt control bits

Interrupt events	Event flag	Interrupt enable bit	EXINT line
Alarm clock A	ALAF	ALAIEN	17
Alarm clock B	ALBF	ALBIEN	17
Periodic automatic wakeup	WATF	WATIEN	22
Time stamp	TSF	TSIEN	21
Tamper event	TP1F/TP2F	TPIEN	21

18.4 ERTC registers

These peripheral registers must be accessed by words (32 bits).
ERTC registers are 32-bit addressable registers.

Table 18-4 ERTC register map and reset values

Register	Offset	Reset value
ERTC_TIME	0x00	0x0000 0000
ERTC_DATE	0x04	0x0000 2101
ERTC_CTRL	0x08	0x0000 0000
ERTC_STS	0x0C	0x0000 0007
ERTC_DIV	0x10	0x007F 00FF
ERTC_WAT	0x14	0x0000 FFFF
ERTC_CCAL	0x18	0x0000 0000
ERTC_ALA	0x1C	0x0000 0000
ERTC_ALB	0x20	0x0000 0000
ERTC_WP	0x24	0x0000 0000
ERTC_SBS	0x28	0x0000 0000
ERTC_TADJ	0x2C	0x0000 0000
ERTC_TSTM	0x30	0x0000 0000
ERTC_TSDT	0x34	0x0000 000D
ERTC_TSSBS	0x38	0x0000 0000
ERTC_SCAL	0x3C	0x0000 0000

ERTC_TAMP	0x40	0x0000 0000
ERTC_ALASBS	0x44	0x0000 0000
ERTC_ALBSBS	0x48	0x0000 0000
ERTC_BPRx	0x50-0x9C	0x0000 0000

18.4.1 ERTC time register (ERTC_TIME)

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value.
Bit 22	AMPM	0x0	rw	AM/PM 0: AM 1: PM Note: This bit is applicable for 12-hr format only. It is 0 for 24-hr format instead.
Bit 21: 20	HT	0x0	rw	Hour tens
Bit 19: 16	HU	0x0	rw	Hour units
Bit 15	Reserved	0x0	resd	Kept at its default value.
Bit 14: 12	MT	0x0	rw	Minute tens
Bit 11: 8	MU	0x0	rw	Minute units
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6: 4	ST	0x0	rw	Second tens
Bit 3: 0	SU	0x0	rw	Second units

18.4.2 ERTC date register (ERTC_DATE)

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x00	resd	Kept at its default value.
Bit 23: 20	YT	0x0	rw	Year tens
Bit 19: 16	YU	0x0	rw	Year units
Bit 15: 13	WK	0x1	rw	Week day 0: Forbidden 1: Monday 2: Tuesday 3: Wednesday 4: Thursday 5: Friday 6: Saturday 7: Sunday
Bit 12	MT	0x0	rw	Month tens
Bit 11: 8	MU	0x1	rw	Month units
Bit 7: 6	Reserved	0x0	resd	Kept at its default value.
Bit 5: 4	DT	0x0	rw	Date tens
Bit 3: 0	DU	0x1	rw	Date units

18.4.3 ERTC control register (ERTC_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x00	resd	Kept at its default value.
Bit 23	CALOEN	0x0	rw	Calibration output enable 0: Calibration output disabled 1: Calibration output enabled
Bit 22: 21	OUTSEL	0x0	rw	Output source selection 00: Output source disabled 01: Alarm clock A 10: Alarm clock B 11: Wakeup events

Bit 20	OUTP	0x0	rw	Output polarity 0: High 1: Low
Bit 19	CALOSEL	0x0	rw	Calibration output selection 0: 512Hz 1: 1Hz
Bit 18	BPR	0x0	rw	Battery powered domain data register This bit in the battery powered domain is not affected by a system reset. It is used to store the daylight saving time change or others that need to be saved permanently.
Bit 17	DEC1H	0x0	wo	Decrease 1 hour 0: No effect 1: Subtract 1 hour Note: This bit is applicable only when the current hour is not 0. The next second takes effect when this bit is set (don't set this bit when the hour is being incremented)
Bit 16	ADD1H	0x0	wo	Add 1 hour 0: No effect 1: Add 1 hour Note: The next second takes effect when this bit is set (don't set this bit when the hour is being incremented)
Bit 15	TSIEN	0x0	rw	Timestamp interrupt enable 0: Timestamp interrupt disabled 1: Timestamp interrupt enabled
Bit 14	WATIEN	0x0	rw	Wakeup timer interrupt enable 0: Wakeup timer interrupt disable 1: Wakeup timer interrupt enabled
Bit 13	ALBIEN	0x0	rw	Alarm B interrupt enable 0: Alarm B interrupt disabled 1: Alarm B interrupt enabled
Bit 12	ALAIEN	0x0	rw	Alarm A interrupt enable 0: Alarm A interrupt disabled 1: Alarm A interrupt enabled
Bit 11	TSEN	0x0	rw	Timestamp enable 0: Timestamp disabled 1: Timestamp enabled
Bit 10	WATEN	0x0	rw	Wakeup timer enable 0: Wakeup timer disabled 1: Wakeup timer enabled
Bit 9	ALBEN	0x0	rw	Alarm B enable 0: Alarm B disabled 1: Alarm B enabled
Bit 8	ALAEN	0x0	rw	Alarm A enable 0: Alarm A disabled 1: Alarm A enabled
Bit 7	CCALEN	0x0	rw	Coarse calibration enable 0: Coarse calibration disabled 1: Coarse calibration enabled
Bit 6	HM	0x0	rw	Hour mode 0: 24-hour format 1: 12-hour format
Bit 5	DREN	0x0	rw	Date/time register direct read enable 0: Date/time register direct read disabled. ERTC_TIME, ERTC_DATE and ERTC_SBS values are taken from the synchronized registers, which are updated once every two ERTC_CLK cycles 1: Date/time register direct read enabled. ERTC_TIME, ERTC_DATE and ERTC_SBS values are taken from the battery powered domain.
Bit 4	RCDEN	0x0	rw	Reference clock detection enable 0: Reference clock detection disabled 1: Reference clock detection enabled
Bit 3	TSEDG	0x0	rw	Timestamp trigger edge 0: Rising edge

				1: Falling edge
Bit 2: 0	WATCLK	0x0	rw	Wakeup timer clock selection 000: ERTC_CLK/16 001: ERTC_CLK/8 010: ERTC_CLK/4 011: ERTC_CLK/2 10x: ck_b 11x: ck_b is selected. 2^{16} is added to the wakeup counter value, and wakeup time =ERTC_WAT+2^{16}. *Note: The write access to this field is supported when WATEN=0 and WATWF=1.*

18.4.4 ERTC initialization and status register (ERTC_STS)

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	CALUPDF	0x0	ro	Calibration value update complete flag 0: Calibration value update is complete 1: Calibration value update is in progress This bit is automatically set when software writes to the ERTC_SCAL register. It is automatically cleared when a new calibration value is taking into account. When this bit is set, the write access to the ERTC_SCAL register is not allowed.
Bit 15	Reserved	0x0	resd	Kept at its default value.
Bit 14	TP2F	0x0	rw0c	Tamper detection 2 flag 0: No tamper event 1: Tamper event occurs
Bit 13	TP1F	0x0	rw0c	Tamper detection 1 flag 0: No tamper event 1: Tamper event occurs
Bit 12	TSOF	0x0	rw0c	Timestamp overflow flag 0: No timestamp overflow 1: Timestamp overflow occurs If a new time stamp event is detected when time stamp flag (TSF) is already set, this bit will be set by hardware.
Bit 11	TSF	0x0	rw0c	Timestamp flag 0: No timestamp event 1: Timestamp event occurs It is recommended to double check the TSOF flag after reading a timestamp and clearing the TSF. Otherwise, a new timestamp event may be detected while clearing the TSF. *Note: The clearing operation of this bit takes effect after two APB_CLK cycles.*
Bit 10	WATF	0x0	rw0c	Wakeup timer flag 0: No wakeup timer event 1: Wakeup timer event occurs *Note: The clearing operation of this bit takes effect after two APB_CLK cycles.*
Bit 9	ALBF	0x0	rw0c	Alarm clock B flag 0: No alarm clock event 1: Alarm clock event occurs *Note: The clearing operation of this bit takes effect after two APB_CLK cycles.*
Bit 8	ALAF	0x0	rw0c	Alarm clock A flag 0: No alarm clock event 1: Alarm clock event occurs *Note: The clearing operation of this bit takes effect after two APB_CLK cycles.*
Bit 7	IMEN	0x0	rw	Initialization mode enable 0: Initialization mode disabled 1: Initialization mode enabled When an initialization mode is entered, the calendar stops running.

Bit 6	IMF	0x0	ro	Enter initialization mode flag 0: Initialization mode is not entered 1: Initialization mode is entered The ERTC_TIME, ERTC_DATE and ERTC_DIV registers can be modified only when an initialization mode is enabled (INITEN=1) and entered (INITEF=1).
Bit 5	UPDF	0x0	rw0c	Calendar update flag 0: Calendar update is in progress 1: Calendar update is complete The UPDF bit is set each time the shadow register is synchronized with the ERTC calendar value located in the battery powered domain. The synchronization is performed every two ERTC_CLK.
Bit 4	INITF	0x0	ro	Calendar initialization flag 0: Calendar has not been initialized 1: Calendar has been initialized This bit is set when the calendar year filed (ERTC_DATE) is different from 0. It is cleared when the year is 0.
Bit 3	TADJF	0x0	ro	Time adjustment flag 0: No time adjustment 1: Time adjustment is in progress This bit is automatically set when a write access to the ERTC_TADJ register is performed. It is automatically cleared at the end of time adjustment.
Bit 2	WATWF	0x1	ro	Wakeup timer register allows write flag 0: Wakeup timer register configuration not allowed 1: Wakeup timer register configuration allowed
Bit 1	ALBWF	0x1	ro	Alarm B register allows write flag 0: Alarm B register write operation not allowed 1: Alarm B register write operation allowed
Bit 0	ALAWF	0x1	ro	Alarm A register allows write flag 0: Alarm A register write operation not allowed 1: Alarm A register write operation allowed

18.4.5 ERTC divider register (ERTC_DIV)

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value.
Bit 22: 16	DIVA	0x7F	rw	Divider A
Bit 15	Reserved	0x0	resd	Kept at its default value.
Bit 14: 0	DIVB	0x00FF	rw	Divider B Calendar clock = ERTC_CLK/((DIVA+1)x(DIVB+1))

18.4.6 ERTC wakeup timer register (ERTC_WAT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	VAL	0xFFFF	rw	Wakeup timer reload value

18.4.7 ERTC coarse calibration register (ERTC_CCAL)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value.
Bit 7	CALDIR	0x0	rw	Calibration direction 0: Positive calibration 1: Negative calibration
Bit 6: 5	Reserved	0x0	resd	Kept at its default value.
Bit 4: 0	CALVAL	0x00	rw	Calibration value Positive calibration 00000: +0 ppm 00001: +4 ppm 00010: +8 ppm

				11111: +126 ppm Negative calibration 00000: -0 ppm 00001: -2 ppm 00010: -4 ppm … 11111: - 63 ppm

18.4.8 ERTC alarm clock A register (ERTC_ALA)

Bit	Name	Reset value	Type	Description
Bit 31	MASK4	0x0	rw	Date/week day mask 0: Date/week day is not masked 1: Alarm clock doesn't care about date/week day
Bit 30	WKSEL	0x0	rw	Date/week day select 0: Date 1: Week day (DT[1: 0] is not used)
Bit 29: 28	DT	0x0	rw	Date tens
Bit 27: 24	DU	0x0	rw	Date/week day units
Bit 23	MASK3	0x0	rw	Hour mask 0: No hour mask 1: Alarm clock doesn't care about hours
Bit 22	AMPM	0x0	rw	AM/PM 0: AM 1: PM *Note: This bit is applicable for 12-hour format only. It is 0 for 24-hour format.*
Bit 21: 20	HT	0x0	rw	Hour tens
Bit 19: 16	HU	0x0	rw	Hour units
Bit 15	MASK2	0x0	rw	Minute mask 0: No minute mask 1: Alarm clock doesn't care about minutes
Bit 14: 12	MT	0x0	rw	Minute tens
Bit 11: 8	MU	0x0	rw	Minute units
Bit 7	MASK1	0x0	rw	Second mask 0: No second mask 1: Alarm clock doesn't care about seconds
Bit 6: 4	ST	0x0	rw	Second tens
Bit 3: 0	SU	0x0	rw	Second units

18.4.9 ERTC alarm clock B register (ERTC_ALB)

Bit	Name	Reset value	Type	Description
Bit 31	MASK4	0x0	rw	Date/week day mask 0: Date/week day is not masked 1: Alarm clock doesn't care about date/week day
Bit 30	WKSEL	0x0	rw	Date/week day select 0: Date 1: Week day (DT[1: 0] is not used)
Bit 29: 28	DT	0x0	rw	Date tens
Bit 27: 24	DU	0x0	rw	Date/week day units
Bit 23	MASK3	0x0	rw	Hour mask 0: No hour mask 1: Alarm clock doesn't care about hours
Bit 22	AMPM	0x0	rw	AM/PM 0: AM 1: PM *Note: This bit is applicable for 12-hour format only. It is 0 for 24-hour format.*
Bit 21: 20	HT	0x0	rw	Hour tens
Bit 19: 16	HU	0x0	rw	Hour units
Bit 15	MASK2	0x0	rw	Minute mask 0: No minute mask

				1: Alarm clock doesn't care about minutes
Bit 14: 12	MT	0x0	rw	Minute tens
Bit 11: 8	MU	0x0	rw	Minute units
Bit 7	MASK1	0x0	rw	Second mask 0: No second mask 1: Alarm clock doesn't care about seconds
Bit 6: 4	ST	0x0	rw	Second tens
Bit 3: 0	SU	0x0	rw	Second units

18.4.10 ERTC write protection register (ERTC_WP)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value
Bit 7: 0	CMD	0x00	wo	Command register All ERTC register write protection is unlocked by writing 0xCA and 0x53. Writing any other value will re-activate write protection.

18.4.11 ERTC subsecond register (ERTC_SBS)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 0	SBS	0x0000	ro	Sub-second value Subsecond is the value in the DIVB counter. Clock frequency = ERTC_CLK/(DIVA+1)

18.4.12 ERTC time adjustment register (ERTC_TADJ)

Bit	Name	Reset value	Type	Description
Bit 31	ADD1S	0x0	wo	Add 1 second 0: No effect 1: Add one second This bit can be written only when TADJF=0. It is intended to be used with DECSBS in order to fine-tune the time.
Bit 30: 15	Reserved	0x0000	resd	Kept at its default value
Bit 14: 0	DECSBS	0x0000	wo	DECSBS[14: 0]: Decrease sub-second value Delay (ADD1S=0): Delay = DECSBS/(DIVB+1) Advance (ADD1S=1) : Advance =1-(DECSBS/(DIVB+1))

18.4.13 ERTC time stamp time register (ERTC_TSTM)

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value
Bit 22	AMPM	0x0	ro	AM/PM 0: AM 1: PM *Note: This bit is applicable for 12-hour format only. It is 0 for 24-hour format.*
Bit 21: 20	HT	0x0	ro	Hour tens
Bit 19: 16	HU	0x0	ro	Hour units
Bit 15	Reserved	0x0	resd	Kept at its default value
Bit 14: 12	MT	0x0	ro	Minute tens
Bit 11: 8	MU	0x0	ro	Minute units
Bit 7	Reserved	0x0	resd	Kept at its default value
Bit 6: 4	ST	0x0	ro	Second tens
Bit 3: 0	SU	0x0	ro	Second units

Note: The content of this register is valid only when the TSF is set in the ERTC_STS register. It is cleared when TSF bit is reset.

18.4.14 ERTC time stamp date register (ERTC_TSDT)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 13	WK	0x0	ro	Week day
Bit 12	MT	0x0	ro	Month tens
Bit 11: 8	MU	0x0	ro	Month units
Bit 7: 6	Reserved	0x0	resd	Kept at its default value
Bit 5: 4	DT	0x0	ro	Date tens
Bit 3: 0	DU	0x0	ro	Date units

Note: The content of this register is valid only when the TSF is set in the ERTC_STS register. It is cleared when TSF bit is reset.

18.4.15 ERTC time stamp subsecond register (ERTC_TSSBS)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 0	SBS	0x0000	ro	Sub-second value

Note: The content of this register is valid only when the TSF is set in the ERTC_STS register. It is cleared when TSF bit is reset.

18.4.16 ERTC smooth calibration register (ERTC_SCAL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15	ADD	0x0	rw	Add ERTC clock 0: No ERTC clock added 1: One ERTC_CLK is inserted every 2^{11} ERTC_CLK cycles
Bit 14	CAL8	0x0	rw	8-second calibration period 0: No effect 1: 8-second calibration
Bit 13	CAL16	0x0	rw	16 second calibration period 0: No effect 1: 16-second calibration
Bit 12: 9	Reserved	0x0	resd	Kept at its default value
Bit 8: 0	DEC	0x000	rw	Decrease ERTC clock DEC out of ERTC_CLK cycles are masked during the 2^{20} ERTC_CLK periods. This bit is usually used with ADD. When the ADD is set, the actual number of ERTC_CLK is equal to 2^{20}+512-DEC during the 2^{20} ERTC_CLK periods.

18.4.17 ERTC tamper configuration register (ERTC_TAMP)

Bit	Name	Reset value	Type	Description
Bit 31: 19	Reserved	0x0000	resd	Kept at its default value
Bit 18	OUTTYPE	0x0	rw	Output type 0: Open-drain output 1: Push-pull output
Bit 17	TSPIN	0x0	rw	Time stamp detection pin selection 0: ERTC_MUX1 1: ERTC_MUX2
Bit 16	TP1PIN	0x0	rw	Tamper detection pin selection 0: ERTC_MUX1 1: ERTC_MUX2
Bit 15	TPPU	0x0	rw	Tamper detection pull-up 0: Tamper detection pull-up enabled

				1: Tamper detection pull-up disabled
Bit 14: 13	TPPR	0x0	rw	Tamper detection pre-charge time 0: 1 ERTC_CLK cycle 1: 2 ERTC_CLK cycles 2: 4 ERTC_CLK cycles 3: 8 ERTC_CLK cycles
Bit 12: 11	TPFLT	0x0	rw	Tamper detection filter time 0: No filter 1: Tamper is detected after 2 consecutive samples 2: Tamper is detected after 4 consecutive samples 3: Tamper is detected after 8 consecutive samples
Bit 10: 8	TPFREQ	0x0	rw	Tamper detection frequency 0: ERTC_CLK/32768 1: ERTC_CLK/16384 2: ERTC_CLK/8192 3: ERTC_CLK/4096 4: ERTC_CLK/2048 5: ERTC_CLK/1024 6: ERTC_CLK/512 7: ERTC_CLK/256
Bit 7	TPTSEN	0x0	rw	Tamper detection timestamp enable 0: Tamper detection timestamp disabled 1: Tamper detection timestamp enabled. Save timestamp on a tamper event.
Bit 6: 5	Reserved	0x0	resd	Kept at its default value
Bit 4	TP2EDG	0x0	rw	Tamper detection 2 valid edge If TPFLT=0: 0: Rising edge 1: Falling edge If TPFLT>0: 0: Low 1: High
Bit 3	TP2EN	0x0	rw	Tamper detection 2 enable 0: Tamper detection 2 disabled 1: Tamper detection 2 enabled
Bit 2	TPIEN	0x0	rw	Tamper detection interrupt enable 0: Tamper detection interrupt disabled 1: Tamper detection interrupt enabled
Bit 1	TP1EDG	0x0	rw	Tamper detection 1 valid edge If TPFLT=0: 0: Rising edge 1: Falling edge If TPFLT>0: 0: Low 1: High
Bit 0	TP1EN	0x0	rw	Tamper detection 1 enable 0: Tamper detection 1 disabled 1: Tamper detection 1 enabled

18.4.18 ERTC alarm clock A subsecond register (ERTC_ALASBS)

Bit	Name	Reset value	Type	Description
Bit 31: 28	Reserved	0x0	resd	Kept at its default value
Bit 27: 24	SBSMSK	0x0	rw	Sub-second mask 0: No comparison. Alarm A doesn't care about subseconds. 1: SBS[0] is compared 2: SBS[1: 0] are compared 3: SBS[2: 0] are compared ... 14: SBS[13: 0] are compared 15: SBS[14: 0] are compared
Bit 23: 15	Reserved	0x000	rw	Kept at its default value
Bit 14: 0	SBS	0x0000	rw	Sub-second value

18.4.19 ERTC alarm clock B subsecond register (ERTC_ALBSBS)

Bit	Name	Reset value	Type	Description
Bit 31: 28	Reserved	0x0	resd	Kept at its default value
Bit 27: 24	SBSMSK	0x0	rw	Sub-second mask 0: No comparison. Alarm B doesn't care about subseconds. 1: SBS[0] is compared 2: SBS[1: 0] are compared 3: SBS[2: 0] are compared ... 14: SBS[13: 0] are compared 15: SBS[14: 0] are compared
Bit 23: 15	Reserved	0x000	rw	Kept at its default value
Bit 14: 0	SBS	0x0000	rw	Sub-second value

18.4.20 ERTC battery powered domain data register (ERTC_BPRx)

Bit	Name	Reset value	Type	Description
Bit 31: 0	DT	0x0000 0000	rw	Battery powered domain data BPR_DTx registers are powered on by V_{BAT} so that they are not reset by a system reset. They are reset on a tamper event or when a battery powered domain is reset.

19 Analog-to-digital converter (ADC)

19.1 ADC introduction

The ADC is a peripheral that converts an analog input signal into a 12-bit/10-bit/8-bit/6-bit digital signal. Its sampling rate is as high as 5.33 MSPS. Each ADC has up to 19 channels (internal + external) for sampling and conversion, totaling 16 external channel sources for two ADCs.

19.2 ADC main features

In terms of analog:

- 12-bit, 10-bit, 8-bit or 6-bit configurable resolution
- Self-calibration time: 205 ADC clock cycles
- ADC conversion time
  - Fast channel: ADC conversion time is 0.1875 µs at 80 MHz in 12-bit resolution (5.33MSps)
  - Slow channel: ADC conversion time is 0.2375 µs at 80 MHz in 12-bit resolution (4.21MSps)
  - Fast channel: ADC conversion time is 0.1126 µs at 80 MHz in 6-bit resolution (8.88MSps)
  - Slow channel: ADC conversion time is 0.1626 µs at 80 MHz in 6-bit resolution (6.15MSps)
- ADC supply requirement: Refer to datasheet for more information
- ADC input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$

In terms of digital control:

- Regular channels and preempted channels with different priority
- Regular channels and preempted channels both have their own trigger detection circuit
- Each channel can independently define its own sampling time
- Conversion sequence supports various conversion modes
- Oversampling: hardware oversampling up to 16-bit resolution
- Optional data alignment mode
- Programmable voltage monitor threshold
- Regular channels with DMA transfers
- Interrupt generation at one of the following events:
  - Conversion overflow of regular channels
  - End of the conversion of preempted channels
  - End of the conversion of regular channels
  - Voltage outside the threshold programmed
  - Triggered conversion error
- ADC master/slave modes
- Master/slave shift mode with programmable timing shift length between ADCs
  - Master-slave mode with DMA is suited to high-performance requirements

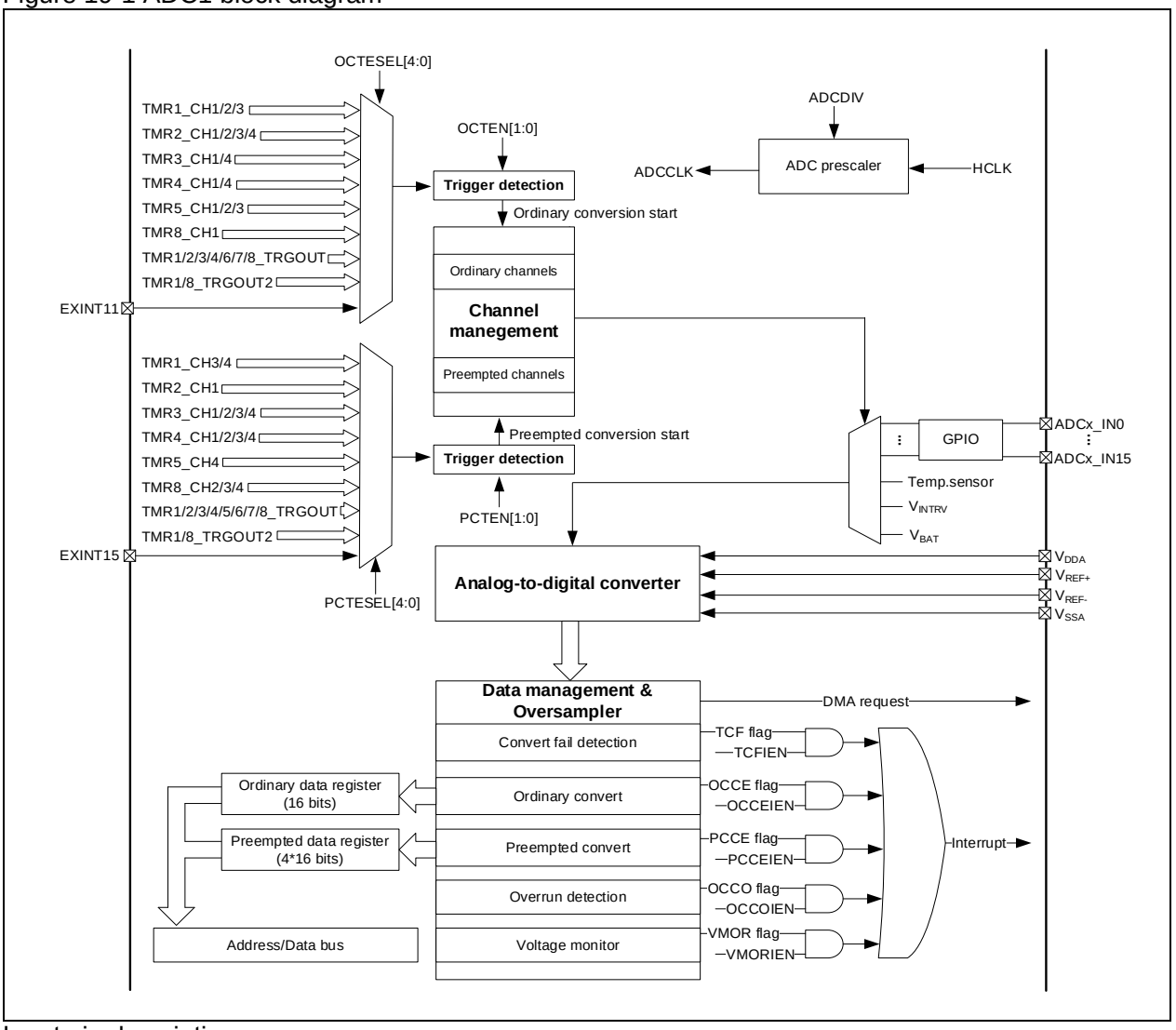
19.3 ADC structure

Figure 19-1 shows the block diagram of ADC1.

Differences between ADC2 and ADC1:

1. ADC2 is not connected to the internal temperature sensor (Temp. sensor) and internal reference voltage (V_{INTRV}) and battery voltage (V_{BAT}).

Figure 19-1 ADC1 block diagram



Input pin description:

- V_{DDA}: Analog supply, ADC analog supply
- V_{SSA}: Analog supply ground, ADC analog supply ground
- V_{REF+}: Analog reference positive, high/positive reference voltage for the ADC
- V_{REF-}: Analog reference negative, low/negative reference voltage for the ADC
- ADCx_IN: Analog input signal channels

Refer to the Datasheet for more information about the input pin connections and voltage ranges.

19.4 ADC functional overview

19.4.1 Channel management

Analog signal channel input:

There are 19 analog signal channel inputs for each of the ADCs, expressed by ADC_Inx (x=0 to 18).

- ADC1_IN0 to ADC1_IN15 are referred to as the external analog input, ADC1_IN16 as an internal temperature sensor, ADC1_IN17 as an internal reference voltage, and ADC1_IN18 as a battery voltage.
- ADC2_IN0 to ADC2_IN15 are referred to as the external analog input, and ADC2_IN16 and ADC2_IN17 as Vss, and ADC1_IN18 as V_{REF-}.

Channel conversion

The conversions are divided into two groups: ordinary and preempted channels. The preempted group has priority over the ordinary group.

If the preempted channel trigger occurs during the ordinary channel conversion, then the ordinary channel conversion is interrupted, giving the priority to the preempted channel, and the ordinary channel continues its conversion at the end of the preempted channel conversion. If the ordinary channel trigger occurs during the preempted channel conversion, the ordinary channel conversion won't start until the end of the preempted channel conversion.

Program the ADC_Inx into the ordinary channel sequence (ADC_OSQx) and the preempted channel sequence (ADC_PSQ), and the same channel can be repeated, the total number of sequences is determined by OCLEN and PCLEN, then it is ready to enable the ordinary channel or preempted channel conversion.

19.4.1.1 Internal temperature sensor

The temperature sensor is connected to ADC1_IN16. Before the temperature sensor channel conversion, it is required to enable the ITSRVEN bit in the ADC_CCTRL register and wait after power-on time.

The temperature sensor output voltage changes linearly with temperature. The offset of this linear function depends on each chip due to process variations from one chip to another. The internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatrues. Using the ADC to convert ADC_IN16, the current temperature sensor output voltage can be read. Then the following formulat can be used to calculate the current temperature.

Obtain the temperature using the following formula:

Temperature (in °C) = {(V 25 - V_{SENSE}) / Avg_Slope} + 25.

Where,

V_{25} = V_{SENSE} value for 25°C and

Avg_Slope = Average Slope for curve between Temperature vs. V_{SENSE} (given in mV/°C).

19.4.1.2 Internal reference voltage

The internal reference voltage of the typical value 1.2 V is connected to ADC1_IN17. It is required to enable the ITSRVEN bit in the ADC_CCTRL register before the internal reference channel conversion. The converted data of such channel can be used to calculate the external reference voltage.

19.4.1.3 Battery voltage

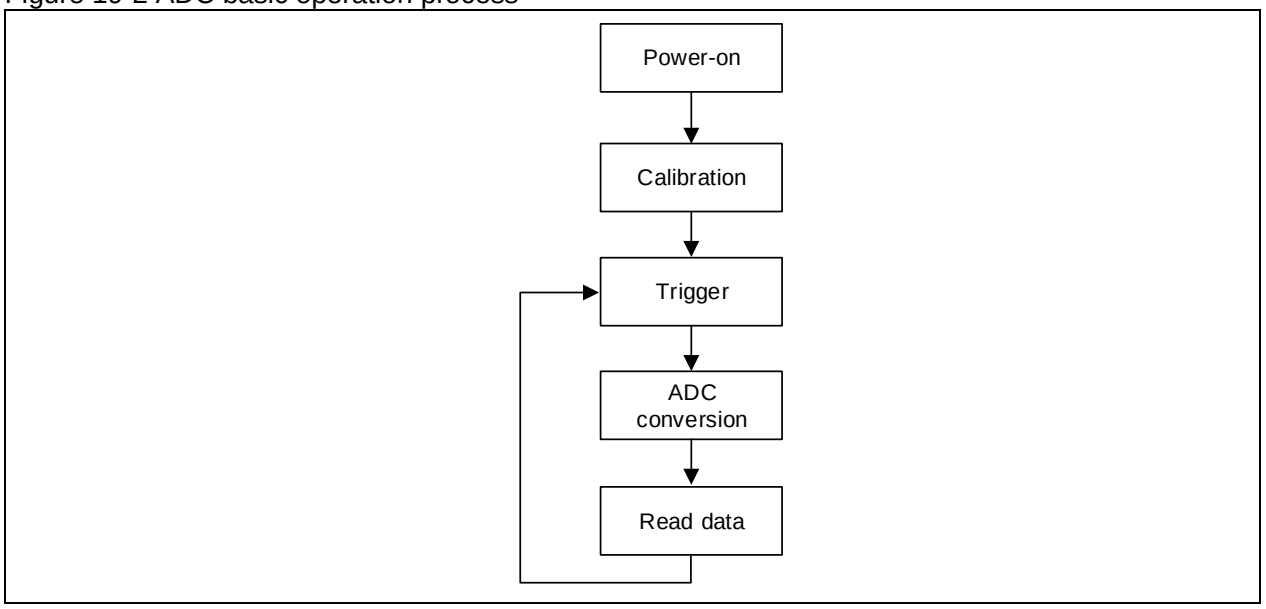
V_{BAT} is higher than V_{DDA}. Thus the V_{BAT} must be divided by 4 before being connected to the ADC1_IN18. The conversion of battery voltage channels starts only when the VBATEN bit is set in the ADC_CCTRL register in order to power the divided-by-4 circuit.

Note: When the ADC is used for sampling VBAT voltage, the VBAT keeps consuming as along as the VBATEN is set. Thus it is recommended to enable VBAT only during the sampling period, and disable it at the end of the sampling in order to lower power consumption.

19.4.2 ADC operation process

Figure 19-2 shows the basic operation process of the ADC. It is recommended to do the calibration after the initial power-on in order to improve the accuracy of sampling and conversion. After the calibration, trigger is used to enable ADC sampling and conversion. Read data at the end of the conversion.

Figure 19-2 ADC basic operation process



19.4.2.1 Power-on and calibration

Power-on

Set the ADCxEN bit in the CRM_APB2EN register to enable ADC clocks: PCLK2 and ADCCLK.

Program the desired ADCCLK frequency by setting the ADCDIV bit in the ADC_CCTRL register. The ADCCLK is derived from HCLK frequency division.

Note: ADCCLK must be less than 80 MHz, while the ADCCLK frequency must be lower than PCLK2.

Then set the ADCEN bit in the ADC_CTRL2 register to supply the ADC, and wait until the RDY flag is set before starting ADC conversion. Clear the ADCEN bit will stop the ADC conversion and result in a reset. In the meantime, the ADC is switched off to save power.

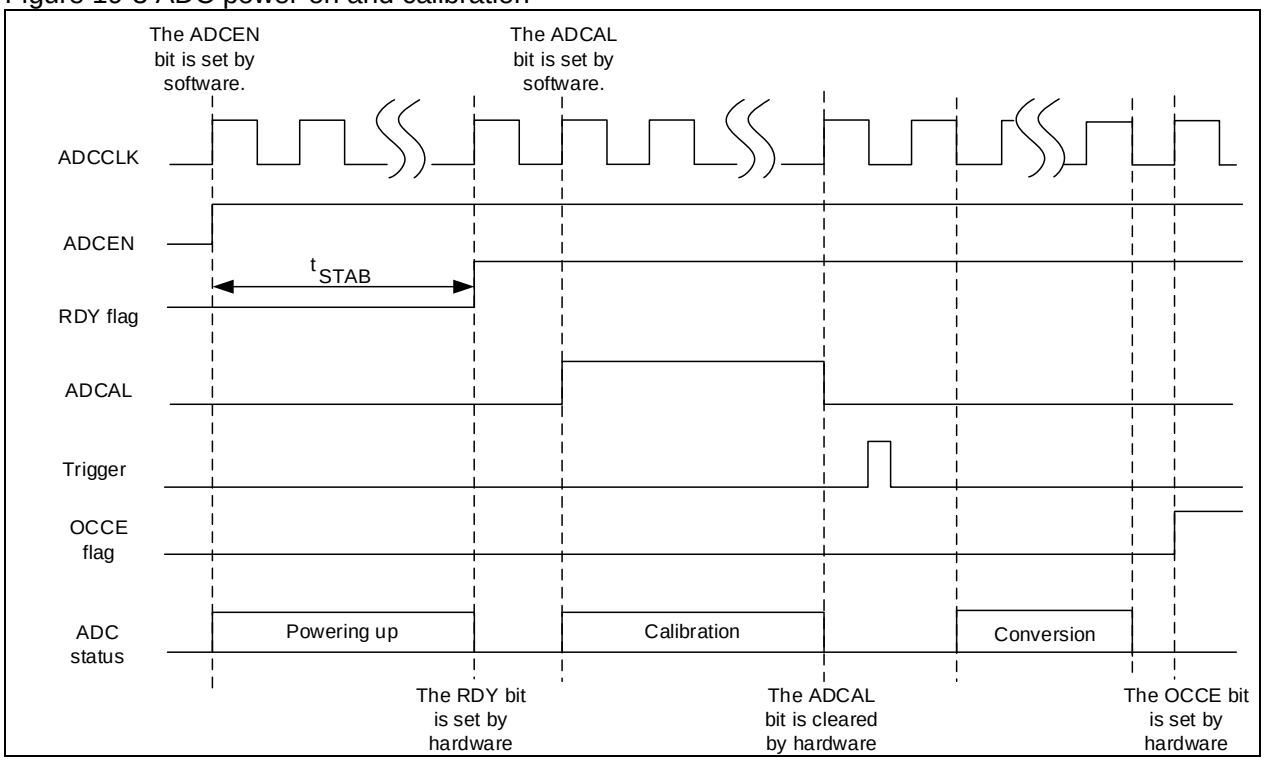
Calibration

After power-on, the calibration is enabled by setting the ADCAL bit in the ADC_CTRL2 register. When the calibration is complete, the ADCAL bit is cleared by hardware and the conversion is performed by software trigger.

After each calibration, the calibration value is stored in the least significant 7 bits of the ADC_CALVAL register, and then value is automatically sent back to the ADC so as to eliminate capacitance errors. The storage of the calibration value will not set the OCCE flag, or generate interrupts or DMA requests.

Note:Calibration is permitted only in 12-bit resolution mode. Switching resolution can be made only after the completion of calibration. Then, it is necessary to wait until the RDY flag is set before trigger operation.

Figure 19-3 ADC power-on and calibration



19.4.2.2 Trigger

The ADC triggers contain ordinary channel trigger and preempted channel trigger. The ordinary channel conversion is triggered by ordinary channel triggers while the preempted channel conversion is triggered by preempted ones. The valid polarity for external trigger sources can be selected by the OCETE and PCETE bits in the ADC_CTRL2 register. The ADC starts conversion after a trigger source is detected.

The conversion can be triggered by software write operation to the OCSWTRG and PCSWTRG bits in the ADC_CTRL2 register, or by an external event. The external events include timer and pin triggers. The OCTESEL and PCTESEL bits in the ADC_CTRL2 register are used to select specific trigger sources, as shown in Table 19-1 and Table 19-2.

Table 19-1 Trigger sources for ordinary channels

OCTESEL	Trigger source	OCTESEL	Trigger source
00000	TMR1_CH1 event	10000	Reserved
00001	TMR1_CH2 event	10001	Reserved
00010	TMR1_CH3 event	10010	Reserved
00011	TMR2_CH2 event	10011	Reserved
00100	TMR2_CH3 event	10100	Reserved
00101	TMR2_CH4 event	10101	TMR8_TRGOUT2 event
00110	TMR2_TRGOUT event	10110	TMR1_TRGOUT2 event
00111	TMR3_CH1 event	10111	TMR4_TRGOUT event
01000	TMR3_TRGOUT event	11000	TMR6_TRGOUT event
01001	TMR4_CH4 event	11001	TMR3_CH4 event
01010	TMR5_CH1 event	11010	TMR4_CH1 event
01011	TMR5_CH2 event	11011	TMR1_TRGOUT event
01100	TMR5_CH3 event	11100	TMR2_CH1 event
01101	TMR8_CH1 event	11101	Reserved
01110	TMR8_TRGOUT event	11110	TMR7_TRGOUT event
01111	EXINT line11 External pin	11111	Reserved

Table 19-2 Trigger sources for preempted channels

PCTESEL	Trigger source	PCTESEL	Trigger source
00000	TMR1_CH4 event	10000	Reserved
00001	TMR1_TRGOUT event	10001	Reserved
00010	TMR2_CH1 event	10010	Reserved
00011	TMR2_TRGOUT event	10011	TMR1_TRGOUT2 event
00100	TMR3_CH2 event	10100	TMR8_TRGOUT event
00101	TMR3_CH4 event	10101	TMR8_TRGOUT2 event
00110	TMR4_CH1 event	10110	TMR3_CH3 event
00111	TMR4_CH2 event	10111	TMR3_TRGOUT event
01000	TMR4_CH3 event	11000	TMR3_CH1 event
01001	TMR4_TRGOUT event	11001	TMR6_TRGOUT event
01010	TMR5_CH4 event	11010	TMR4_CH4 event
01011	TMR5_TRGOUT event	11011	TMR1_CH3 event
01100	TMR8_CH2 event	11100	Reserved
01101	TMR8_CH3 event	11101	Reserved
01110	TMR8_CH4 event	11110	TMR7_TRGOUT event
01111	EXINT line15 External pin	11111	v

19.4.2.3 Sampling and conversion sequence

The sampling period can be configured by setting the CSPTx bit in the ADC_SPT1 and ADC_SPT2 registers. Channels 0/1/8/9/12/13 are fast channels with a minimum of 2.5 sampling cycles, and others represent slow ones supporting a minimum of 6.5 sampling cycles.

The resolution can be programmed through the CRSEL bit in the ADC_CTRL1 register. A lower resolution has shorter conversion time. A single one conversion time is calculated with the following formula:

$$\text{A single one conversion time (ADCCLK period)} = \text{sampling time} + \text{resolution bits} + 0.5$$

Example:

If the CSPTx selects 2.5 period, and CRSEL select 12-bit resolution, then one conversion needs 2.5+12.5=15 ADCCLK periods

If the CSPTx selects 6.5 period, and CRSEL select 10-bit resolution, then one conversion needs 6.5+10.5=17 ADCCLK periods.

19.4.3 Conversion sequence management

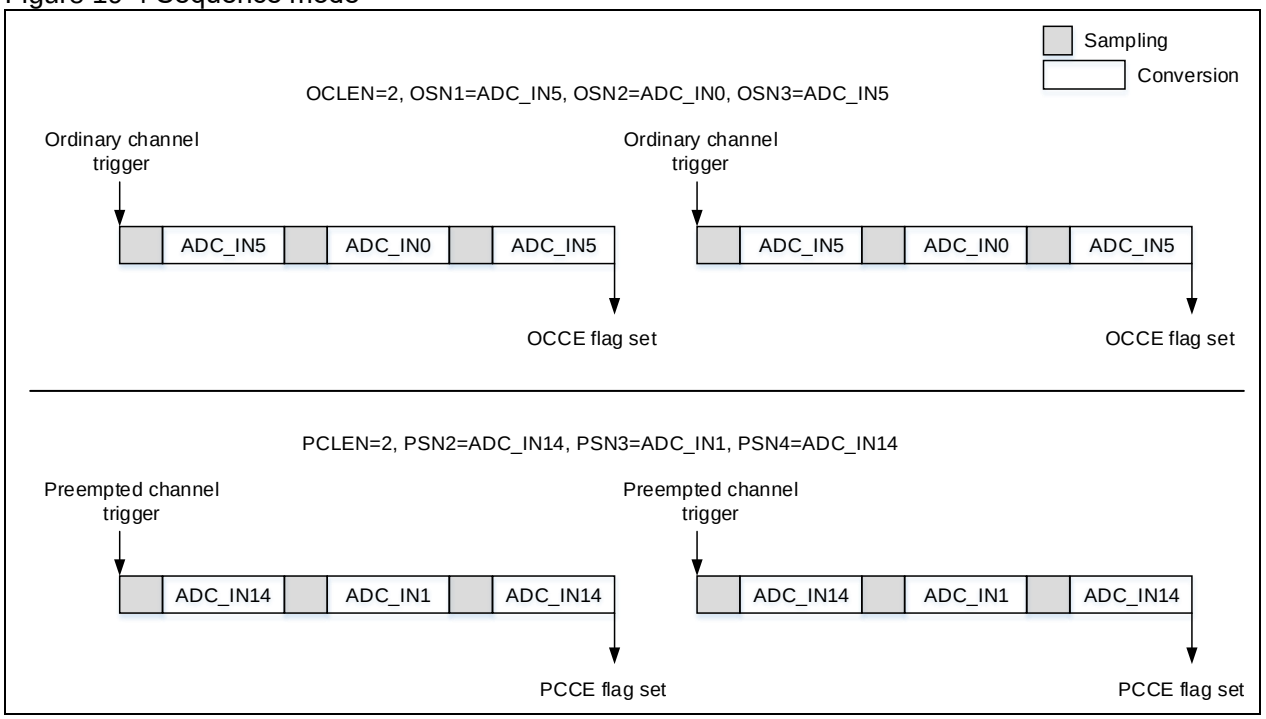
Only one channel is converted at each trigger event by default, that is, OSN1-defined channel or PSN4-defined channel.

The detailed conversion sequence modes are described in the following sections. With this, the channels can be converted in a specific order.

19.4.3.1 Sequence mode

The sequence mode is enabled by setting the SQEN bit in the ADC_CTRL1 register. The ADC_OSQx registers are used to configure the sequence and total number of the ordinary channels while the ADC_PSQ register is used to define the sequence and total number of the preempted channels. When the sequence mode is enabled, a single trigger event enables the conversion of a group of channels in order. The ordinary channels start converting from the QSN1 while the preempted channels starts from the PSNx, where x=4-PCLEN. Figure 19-4 shows an example of the behavior in sequence mode.

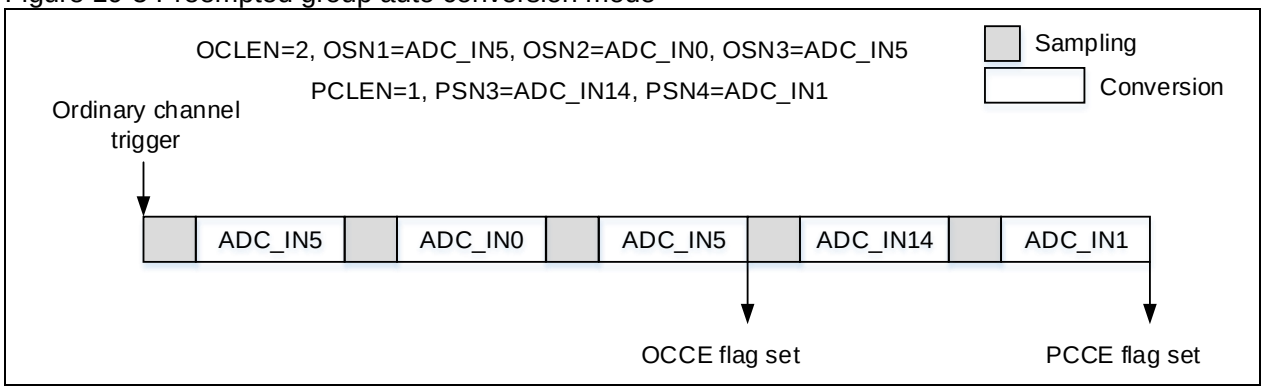
Figure 19-4 Sequence mode



19.4.3.2 Automatic preempted group conversion mode

The automatic preempted group conversion mode is enabled by setting the PCAUTOEN bit in the ADC_CTRL1 register. Once the ordinary channel conversion is over, the preempted group will automatically continue its conversion. This mode can work with the sequence mode. The preempted group conversion starts automatically at the end of the conversion of the ordinary group. Figure 19-5 shows an example of the behavior when the automatic preempted group conversion mode works with the ordinary group.

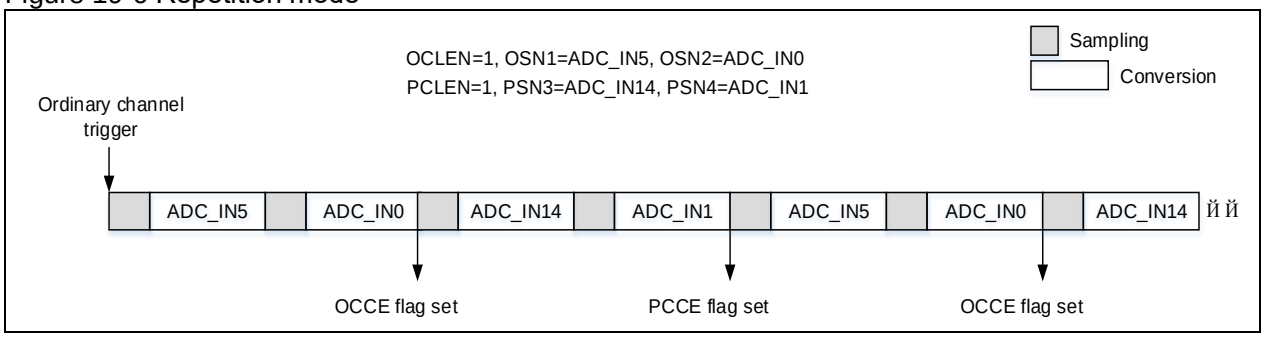
Figure 19-5 Preempted group auto conversion mode



19.4.3.3 Repetition mode

The repetition mode is enabled by setting the RPEN bit in the ADC_CTRL2 register. When a trigger signal is detected, the ordinary channels will be converted repeatedly. This mode can work with the ordinary channel conversion in the sequence mode to enable the repeated conversion of the ordinary group. Such mode can also work with the preempted group auto conversion mode to repeatedly convert the ordinary group and preempted group in sequence. Figure 19-6 shows an example of the behavior when the repetition mode works with the sequence mode and preempted group auto conversion mode.

Figure 19-6 Repetition mode



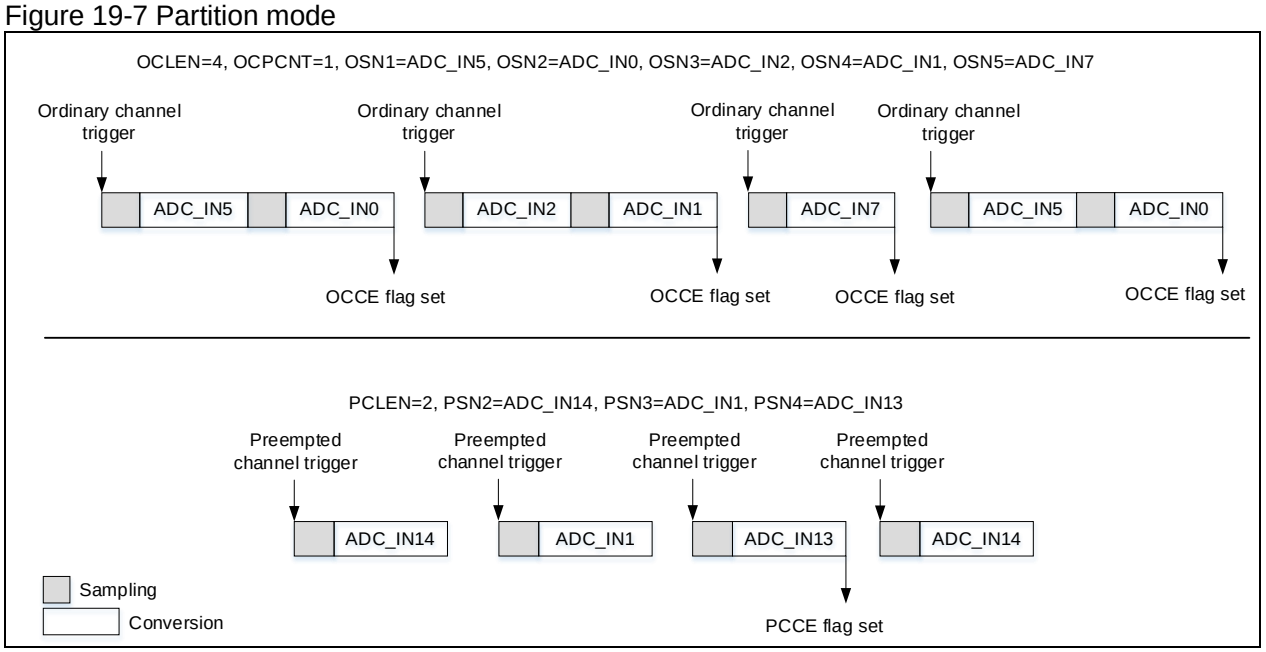
19.4.3.4 Partition mode

The partition mode of the ordinary group can be enabled by setting the OCPEN bit in the ADC_CTRL1 register. In this mode, the ordinary group conversion sequence length (OCLEN bit in the ADC_OSQ1 register) is divided into a smaller sub-group, in which the number of the channels is programmed with the OCPCNT bit in the ADC_CTRL1 register. A single trigger event will enable the conversion of all the channels in the sub-group. Each trigger event selects different sub-group in order.

Set the PCPEN bit in the ADC_CTRL1 register will enable the partition mode of the preempted group. In this mode, the preempted group conversion sequence length (PCLEN bit in the ADC_PSQ register) is divided into a sub-group with only one channel. A single one trigger event will enable the conversion of all the channels in the sub-group. Each trigger event selects different sub-group in order.

The partition mode cannot be used with the repetition mode at the same time. Figure 19-7 shows an example of the behavior in partition mode for ordinary group and preempted group.

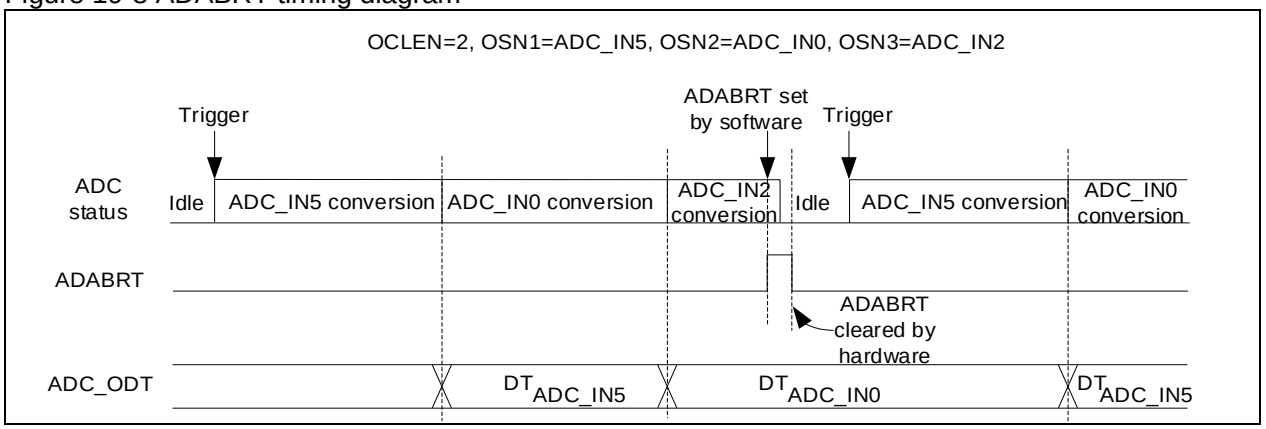
Figure 19-7 Partition mode



19.4.4 End of conversion

The ADABRT bit in the ADC_CTRL2 register can be used to stop ADC conversions. At the end of the conversions, the conversion sequence returns to the first channel. This allows the user to configure a new channel sequence, and the ADC starts conversions from the beginning based on the new order when a trigger event occurs. If the ADABRT is set, it is necessary to wait until it is cleared by hardware before starting other ADC operations. Figure 19-8 shows the timing diagram when the ADABRT bit is set.

Figure 19-8 ADABRT timing diagram



19.4.5 Trigger conversion failure

In case of ADC conversion failure, the TCF flag is set to indicate this failure, and the data in regular or preempted data registers become invlaid. By default, the ADC coversion would not stop due to conversion failure.

If there is a need to stop Adc conversion when coversion failed, it is possible to re-configure conversion sequence or channels in the trigger conversion failed interrupt (enabled through TCFIEN bit). Additionally, if the trigger conversion failed interrupt and auto stop conversion after trigger failure (using the TCFAC bit) are enabled at the same time, then when a conversion failed, the ADC will automatically stop the ongoing conversion sequence or channels and resets itselt to idle state. The subsequent valid trigger would re-trigger ADC to start conversion from the first channel.

19.4.6 Oversampling

A single oversampling converted data can be done through multiple conversions of the same channel in which the cumulative converted data is averaged.

- Oversampling ratio is selected through the OSRSEL bit in the ADC_OVSP register. This bit is used to specify the oversampling multiple, which is performed by converting the same channel several times
- Oversampling shift is selected through the OSSSEL bit in the ADC_OVSP register, which is performed by right shift

If the averaged data is greater than 16 bits, then only pick up the right-aligned 16-bit data and put them into a 16-bit data register, shown in Table 19-3.

Example:

If 4x oversampling is selected through the OSRSEL bit, then the same channel is converted by four times in a single oversampling conversion, and the converted data derived from 4 conversions is put together. If 6-bit resolution is selected through the OSSSEL bit, then the cumulative data is divided by 2^6 and rounded up.

Table 19-3 Correlation between maximum cumulative data, oversampling multiple and shift digits

Oversampling multiple	2x	4x	8x	16x	32x	64x	128x	256x
Max cumulative data	0x1FFE	0x3FFC	0x7FF8	0xFFF0	0x1FFE0	0x3FFC0	0x7FF80	0xFFF00
No shift	0x1FFE	0x3FFC	0x7FF8	0xFFF0	0xFFE0	0xFFC0	0xFF80	0xFF00
Shift 1 digit	0x0FFF	0x1FFE	0x3FFC	0x7FF8	0xFFF0	0xFFE0	0xFFC0	0xFF80
Shift 2 digits	0x0800	0x0FFF	0x1FFE	0x3FFC	0x7FF8	0xFFF0	0xFFE0	0xFFC0
Shift 3 digits	0x0400	0x0800	0x0FFF	0x1FFE	0x3FFC	0x7FF8	0xFFF0	0xFFE0
Shift 4 digits	0x0200	0x0400	0x0800	0x0FFF	0x1FFE	0x3FFC	0x7FF8	0xFFF0
Shift 5 digits	0x0100	0x0200	0x0400	0x0800	0x0FFF	0x1FFE	0x3FFC	0x7FF8
Shift 6 digits	0x0080	0x0100	0x0200	0x0400	0x0800	0x0FFF	0x1FFE	0x3FFC
Shift 7 digits	0x0040	0x0080	0x0100	0x0200	0x0400	0x0800	0x0FFF	0x1FFE
Shift 8 digits	0x020	0x0040	0x0080	0x0100	0x0200	0x0400	0x0800	0x0FFF

When using oversampling conversion mode, the DTALIGN and PCDTOx are ignored, and data must be right aligned. The CRSEL bit is used to select the desired oversampling resolution only, without changing the data operation mode.

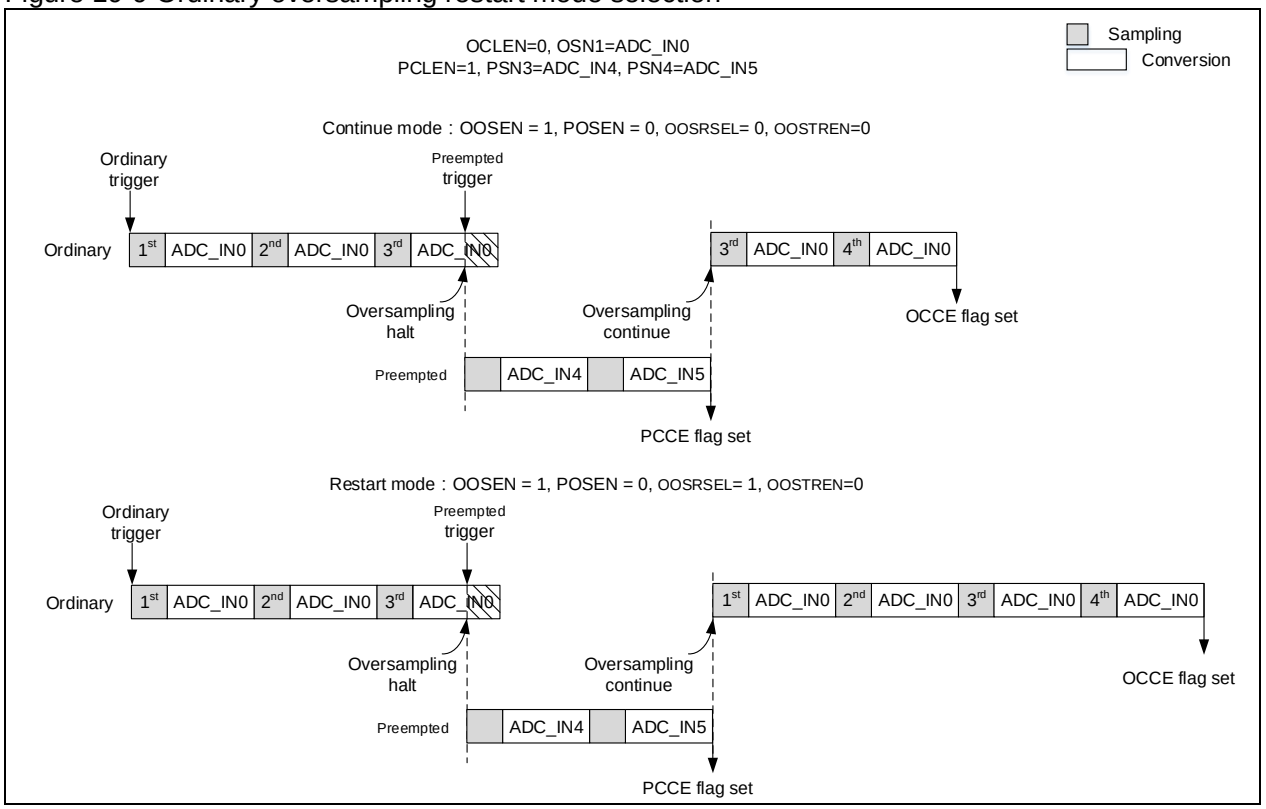
19.4.6.1 Oversampling of ordinary group of channels

The OOSRSEL bit in the ADC_OVSP register can be used to resume ordinary oversampling mode.

- OOSRSEL=0: continuous conversion mode. Ordinary group of channels, after being interrupted by preempted group of channels during oversampling, will retain the converted data and resume from the last interrupted ordinary conversion.
- OOSRSEL=1: restart mode. Ordinary group of channels, after being interrupted by preempted group of channels during oversampling, will be reset and restart the ordinary conversion.

Figure 19-9 shows the differences between ordinary continuous modes and restart mode in 4x oversampling rate and sequential mode.

Figure 19-9 Ordinary oversampling restart mode selection

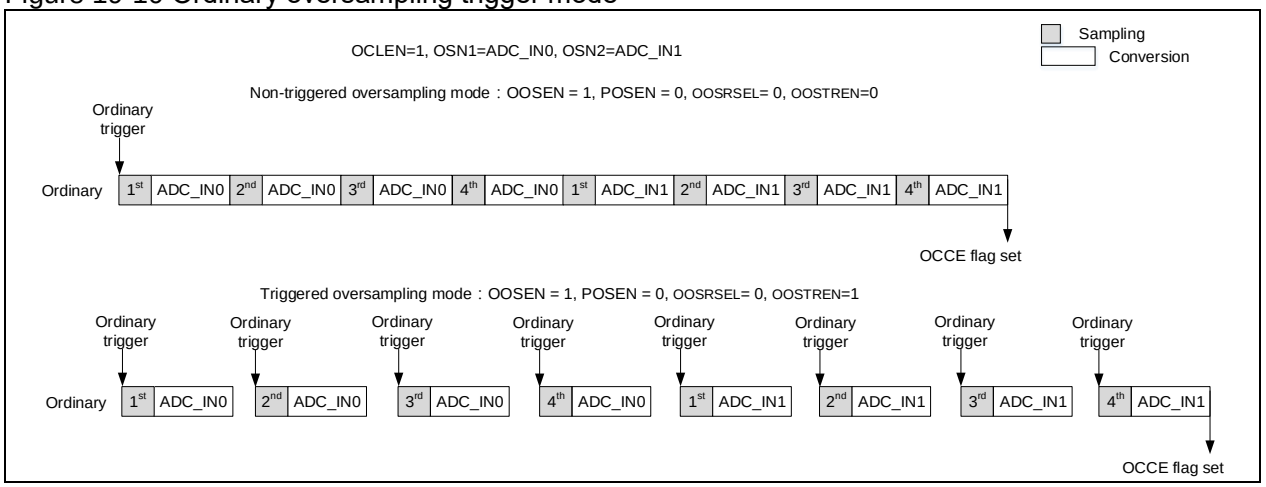


Trigger mode can be enabled by setting the OOSTREN bit in the ADC_OVSP register. The user must trigger each of the ordinary conversions. In this mode, once the ordinary conversion is interrupted by preempted group of channels, it is necessary to re-trigger ordinary group of channels before resuming the ordinary channels.

When the trigger mode works together with conversion sequence management mode, trigger mode is applied, and the conversion complete flag follows the conversion sequence management mode. Figure 19-10 shows the behavior when the ordinary trigger mode works together with resume mode in 4x oversampling rate and sequential mode.

Note: It is not possible to use both the trigger mode and repetition mode simultaneously.

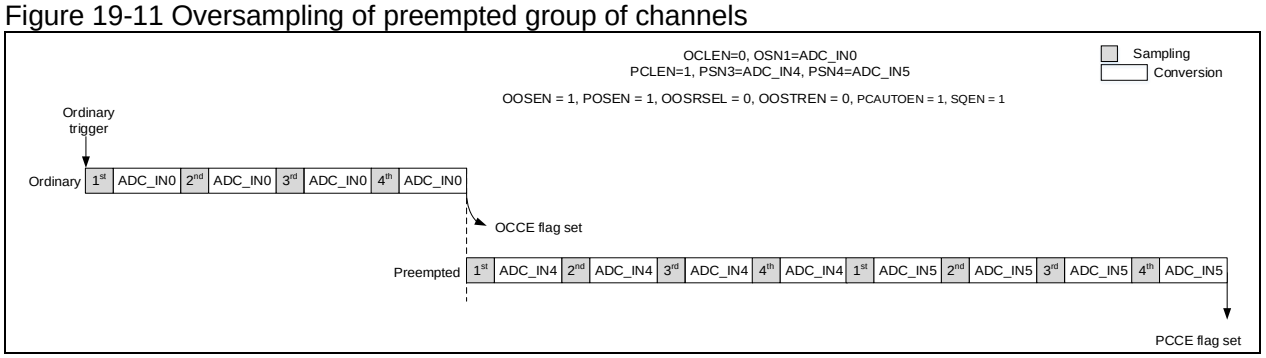
Figure 19-10 Ordinary oversampling trigger mode



19.4.6.2 Oversampling of preempted group of channels

It is possible to use both the preempted oversampling and ordinary oversampling simultaneously or individually. The oversampling of the preempted group of channels does not affect the ordinary oversampling modes. Figure 18-11 shows the behavior when the preempted oversampling and ordinary oversampling trigger mode are used simultaneously in 4x oversampling rate and auto-conversion of preempted group.

Figure 19-11 Oversampling of preempted group of channels



19.4.7 Data management

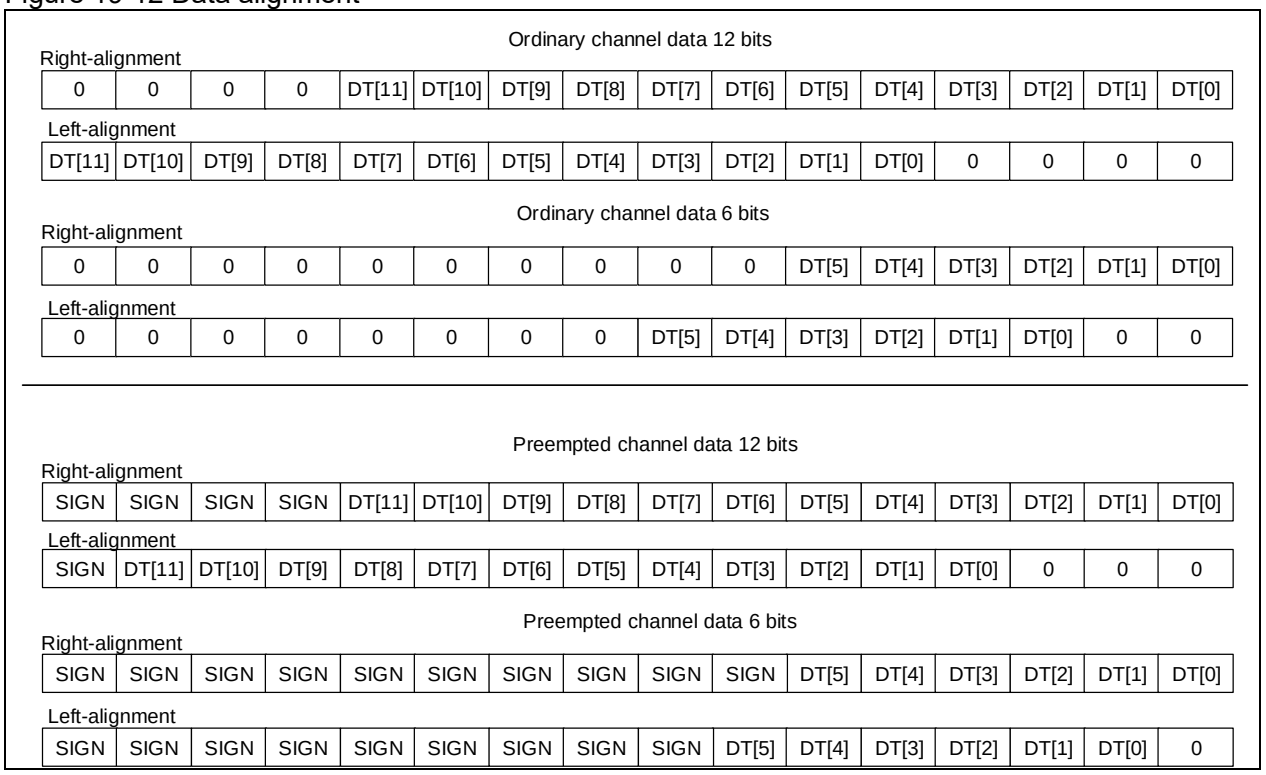
At the end of the conversion of the ordinary group, the converted value is stored in the ADC_ODT register. Once the preempted group conversion ends, the converted data of the preempted group is stored in the ADC_PDTx register.

19.4.7.1 Data alignment

DTALIGN bit in the ADC_CTRL2 register selects the alignment of data (right-aligned or left-aligned). Apart from this, the converted data of the preempted group is decreased by the offset written in the ADC_PCDTOx register. Thus the result may be a negative value, marked by SIGN.

The data are aligned on a half-word basis expect when the resolution is set to 6-bit. In this case, the data are aligned on a byte basis, as shown in Figure 19-12.

Figure 19-12 Data alignment



19.4.7.2 Data read

Read access to the ADC_ODT register using CPU or DMA gets the converted data of the ordinary group. Read access to the ADC_PDTx register using CPU gets the converted data of the preempted group.

The EOCSFEN bit in the ADC_CTRL2 register can be used to select when to set the ordinary group conversion complete flag, that is, at the end of sequence mode or each time the ordinary data register is updated.

When the OCDMAEN bit is set in the ADC_CTRL2 register, the ADC will issue DMA requests each time the ADC_OTD register is updated.

When the EOCSFEN or OCDMAEN is set, the ADC will automatically start overflow detection. If an overflow event occurs, the OCCO flag will be set, the ADC stops conversion, and the last valid data is stored in the data register. If the DMA is used, the DMA request remains set so that the DMA can read the last valid data. The OCCO flag is cleared by software, and the ADC is triggered again so that it starts conversion from the next channel of the valid data. In this case, even if an overflow event occurs halfway, all the data read are valid and in order.

The OCDRCEN bit in the ADC_CTRL2 register can be used to select whether to continually send DMA requests after the DMA transfer register is reset.

19.4.8 Voltage monitoring

The OCVMEN bit or PCVMEN bit in the ADC_CTRL1 register is used to enable voltage monitoring based on the converted data.

The VMOR bit will be set if the converted result is outside the high threshold (ADC_VMHB[11:0] register) or is less than the low threshold (ADC_VMLB[11:0]register).

In 10-bit resolution, ADC_VMHB[11:10] and ADC_VMLB[11:10] must be set to 2’b00;

In 8-bit resolution, ADC_VMHB[11:8] and ADC_VMLB[11:8] must be set to 4’b0000;

In 6-bit resolution, ADC_VMHB[11:6] and ADC_VMLB[11:6] must be set to 6’b000000;

The VMSGEN bit in the ADC_CTRL1 register is used to enable voltage monitor on either a single channel or all the channels. The VMCSEL bit is used to select a specific channel that requires voltage monitoring.

Voltage monitoring is based on the comparison result between the original converted data and the 12-bit voltage monitor boundary register, irrespective of the CRSEL, PCDTOx and DTALIGN bits.

When using an oversampler, voltage monitoring is based on the comparison result between the 16-bit registers (ADC_VMHB[15:0] and ADC_VMLB[15:0]) and the oversampled data.

19.4.8.1 Status flag and interrupts

Each of the ADCs has its dedicated ADCx_STS registers, that is, RDY flag, OCCO flag, OCCS (ordinary channel conversion start flag), PCCS (preempted channel conversion start flag), PCCE (preempted channel conversion end flag), OCCE (ordinary channel conversion end flag) and VMOR (voltage monitor out of range).

Three ADCx_STS registers are mapped onto the ADC_CSTS register, so it is possible to obtain the status of these tree registers by simply reading the ADC_CSTS register.

OCCO, PCCE, CCE and VMOR have their respective interrupt enable bits. Once the interrupt bits are enabled, the corresponding flag is set and an interrupt is sent to CPU. ADC1 shares an interrupt vector with ADC2..

19.5 Master/Slave mode

If Master/Slave mode is enabled, the master is triggered to work with the slave to do the channel conversion. The ADC_ODT register is used as a single interface obtaining the ordinary channel converted data of master/salve ADC.

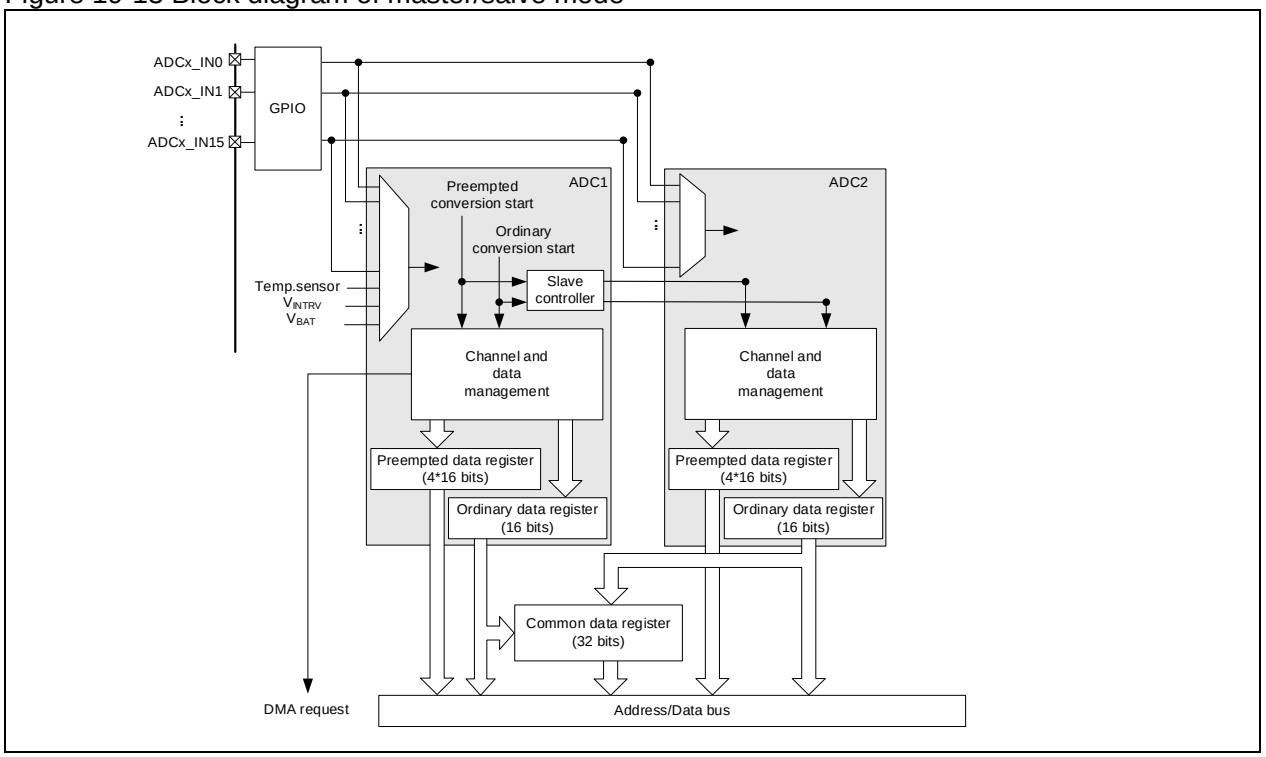
In a single master/slave mode, ADC1 acts as a master while ADC2 as a slave. In master/salve mode, it is necessary to enable the trigger mode of master and slave ADC.

Note: ADC conversion abort (ADABRT) cannot be used in master/slave mode. In this mode, each of the ADCEN bit of the ADCs must be cleared in order to stop ADC conversions.

Note: Both the master and slave must have the same resolution in order to avoid losing synchronization between master and slave.

Note: To enable several ADC conversions with low resolution simultaneously, it is recommended to use the master/slave mode together with DMA1 or DMA2.

Figure 19-13 Block diagram of master/salve mode



19.5.1 Data management

In Master/Slave mode, the converted data of ordinary channels is also stored in the ADC_CODT register. The MSDMASEL bit in the ADC_CCTRL register can be used to select from five DMA transfer modes, as shown in Table 19-4. As long as the MSDMASEL is set, the ADC1 DMA channel is used to generate a DMA request each time the data is ready, and overflow detection on master/salve is also enabled. Once an overflow event occurs, the ADC will stop conversion, the DMA request is halted, and loss of synchronization may happen between the master and slave, so it is recommended to re-initialize ADC before re-triggering on an overflow event.

Table 19-4 Master/slave DMA mode

MSDMASEL	Master/slave mode	DMA requests	ADC_CODT[31:0]
001	Single slave	1st	16 bit 0, ADC1_ODT[15:0]
		2nd	16 bit 0, ADC2_ODT [15:0]
		3rd	16 bit 0, ADC1_ODT [15:0]
010	Single slave	1st	ADC2_ODT[15:0], ADC1_ODT[15:0]
		2nd	ADC2_ODT[15:0], ADC1_ODT[15:0]
011	Single slave	1st	16 bit 0, ADC2_ODT[7:0], ADC1_ODT[7:0]
		2nd	16 bit 0, ADC2_ODT[7:0], ADC1_ODT[7:0]

MSDRCEN bit in the ADC_CCTRL register can be used to select when to stop DMA request, that is, when the DMA request remains set until the end of data transfer, or when the DMA transfer register is reset.

19.5.2 Simultaneous mode

Regular simultaneous mode

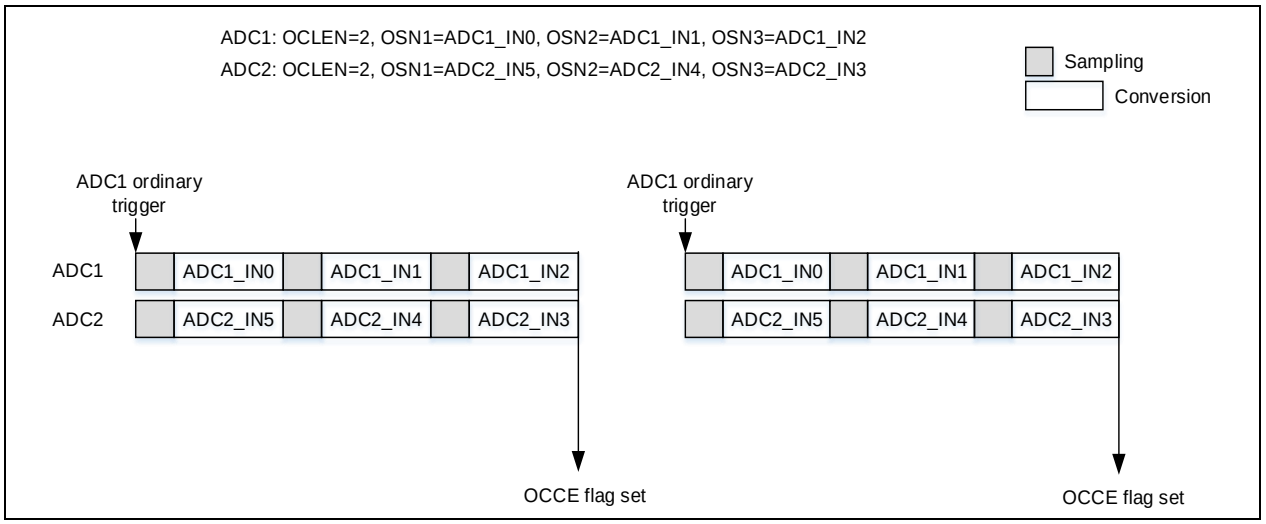
MSSEL bit in the ADC_CCTRL register is used to select regular simultaneous mode. If this mode is enabled, the regular channels of the master are triggered so that both the master and the slave convert the regular channels simultaneously. In this mode, it is required to configure the same sampling time and the same sequence length for the master and slave to avoid the loss of data due to the lack of synchronization.

Figure 19-14 shows an example of the regular simultaneous mode

The single slave mode can work with the mode 1/2/3 of the transfer mode (MSDMASEL).

Note: The same channel is not allowed to be sampled by several ADCs simultaneously. Do not put the same channel in the same sequence location of different ADCs.

Figure 19-14 Regular simultaneous mode

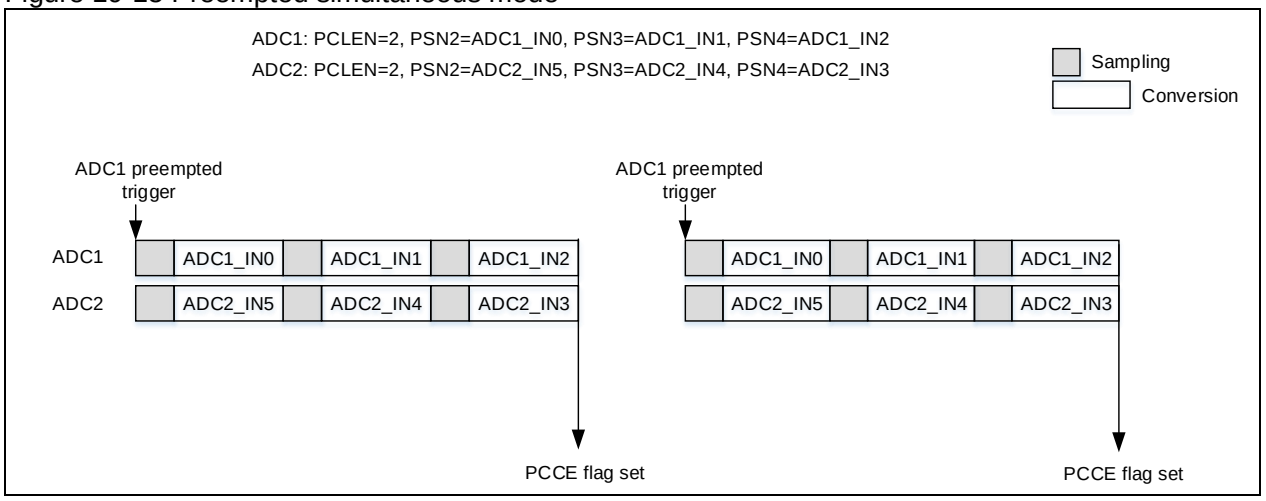


Preempted simultaneous mode

MSSEL bit in the ADC_CCTRL register is used to select preempted simultaneous mode. If this mode is enabled, the preempted channels of the master is triggered so that both the master and the slave convert the preempted channels simultaneously. Figure 19-15 shows an example of the preempted simultaneous mode

Note: The same channel is not allowed to be sampled by several ADCs simultaneously. Do not put the same channel in the same sequence location of different ADCs.

Figure 19-15 Preempted simultaneous mode



Combined regular/preempted simultaneous mode

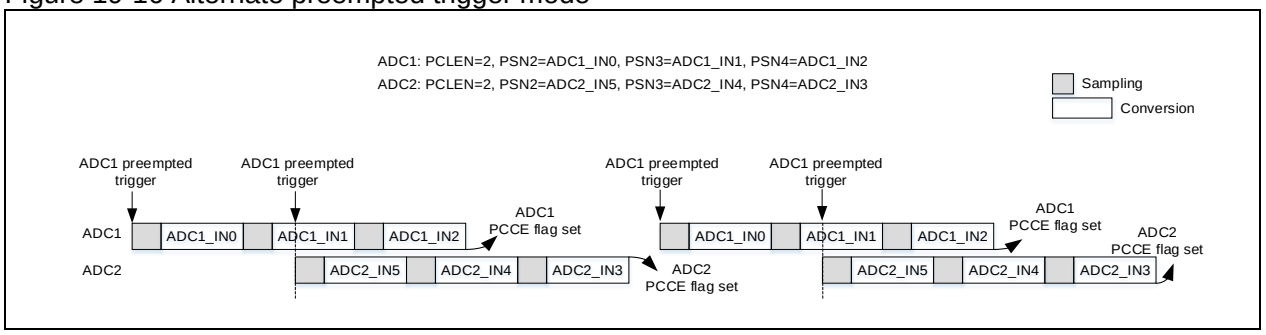
MSSEL bit in the ADC_CCTRL register is used to select combined regular/preempted simultaneous mode. If this mode is enabled, the regular channels of the master is triggered so that the master works with the slave to convert the regular channels simultaneously, or the preempted channels of the master is triggered to enable the master and slave to convert the preempted channels simultaneously.

19.5.3 Alternate preempted trigger mode

Alternate preempted trigger mode

MSSEL bit in the ADC_CCTRL register selects the alternate preempted trigger mode. If this mode is enabled, the preempted channels of the master are triggered continuously so that the master/slave ADCs convert the preempted channels alternately. Figure 19-16 shows an example of the alternate preempted trigger mode.

Figure 19-16 Alternate preempted trigger mode



Combined regular simultaneous + alternate preempted trigger mode

MSSEL bit in the ADC_CCTRL register is used to select combined regular simultaneous + alternate preempted trigger mode. In this mode, trigger the regular group of the master to start regular simultaneous conversion of master/slave, or trigger the preempted group of the master continuously to allow the master/slave ADCs to convert the preempted group alternately.

If the regular conversion is interrupted by the preempted trigger, the regular conversion of all ADCs is stopped, and one of the ADCs starts the preempted conversion. At this point, the master will ignore the preempted trigger until the regular conversion is resumed.

19.5.4 Regular shift mode

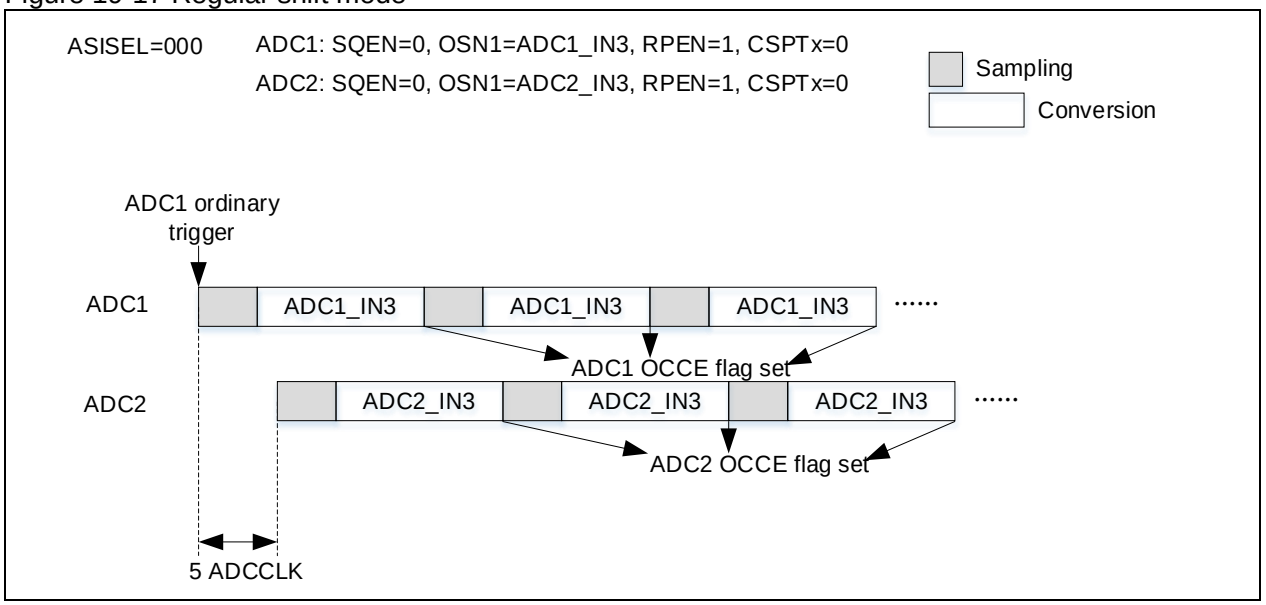
MSSEL bit in the ADC_CCTRL register is used to select regular shift mode on regular group. After a master trigger occurs, the conversion interval between ADCs is based on the programmed shift length (through the ASISEL bit in the ADC_CCTRL register), as shown in Figure 19-17.

In this mode, the sampling interval between ADCs is configured to be at least 2.5 ADCCLK cycles by hardware. Thus the ASISEL bit becomes invalid when it cannot meet such sampling interval. Because of this feature, it is possible to put the same channel in the same ADC location without causing the overlapped sampling time.

The single slave mode can work with the transfer mode 1/2/3 (MSDMASEL).

Note: The preempted trigger or slave regular trigger is not allowed in this mode.

Figure 19-17 Regular shift mode



19.6 ADC registers

Table 19-5 lists ADC register map and their reset values.

These peripheral registers must be accessed by words (32 bits).

Table 19-5 ADC register map and reset values

Register	Offset	Reset value
ADC1_STS	0x000	0x0000 0000
ADC1_CTRL1	0x004	0x0000 0000
ADC1_CTRL2	0x008	0x0000 0000
ADC1_SPT1	0x00C	0x0000 0000
ADC1_SPT2	0x010	0x0000 0000
ADC1_PCDTO1	0x014	0x0000 0000
ADC1_PCDTO2	0x018	0x0000 0000
ADC1_PCDTO3	0x01C	0x0000 0000
ADC1_PCDTO4	0x020	0x0000 0000
ADC1_VMHB	0x024	0x0000 FFFF
ADC1_VMLB	0x028	0x0000 0000
ADC1_OSQ1	0x02C	0x0000 0000
ADC1_OSQ2	0x030	0x0000 0000
ADC1_OSQ3	0x034	0x0000 0000
ADC1_PSQ	0x038	0x0000 0000
ADC1_PDT1	0x03C	0x0000 0000
ADC1_PDT2	0x040	0x0000 0000
ADC1_PDT3	0x044	0x0000 0000
ADC1_PDT4	0x048	0x0000 0000
ADC1_ODT	0x04C	0x0000 0000
ADC1_OVSP	0x080	0x0000 0000
ADC2_STS	0x100	0x0000 0000
ADC2_CTRL1	0x104	0x0000 0000
ADC2_CTRL2	0x108	0x0000 0000
ADC2_SPT1	0x10C	0x0000 0000
ADC2_SPT2	0x110	0x0000 0000
ADC2_PCDTO1	0x114	0x0000 0000
ADC2_PCDTO2	0x118	0x0000 0000
ADC2_PCDTO3	0x11C	0x0000 0000
ADC2_PCDTO4	0x120	0x0000 0000
ADC2_VMHB	0x124	0x0000 FFFF
ADC2_VMLB	0x128	0x0000 0000
ADC2_OSQ1	0x12C	0x0000 0000
ADC2_OSQ2	0x130	0x0000 0000
ADC2_OSQ3	0x134	0x0000 0000
ADC2_PSQ	0x138	0x0000 0000
ADC2_PDT1	0x13C	0x0000 0000
ADC2_PDT2	0x140	0x0000 0000
ADC2_PDT3	0x144	0x0000 0000
ADC2_PDT4	0x148	0x0000 0000
ADC2_ODT	0x14C	0x0000 0000
ADC2_OVSP	0x180	0x0000 0000
ADC_CSTS	0x300	0x0000 0000

ADC_CCTRL	0x304	0x0000 0000
ADC_CODT	0x308	0x0000 0000

19.6.1 ADC status register (ADC_STS)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x0000000	resd	Kept at its default value.
Bit 7	TCF	0x0	rw0c	Trigger convert fail flag When a trigger conversion failed, this bit is set by hardware. It is cleared by writing 1 to itself. 0: Trigger conversion success 1: Trigger conversion failed
Bit 6	RDY	0x0	rw0c	ADC conversion ready flag This bit is read only. It is set by hardware when the ADC is powered. 0: Not ready for ADC conversion 1: Ready for ADC conversion
Bit 5	OCCO	0x0	rw0c	Ordinary channel conversion overflow flag) This bit is set by hardware and cleared by software (writing 0). 0: No overflow occurred 1: Overflow occurred Overflow detection is applicable to the case of DMA transfer enable or EOCSFEN =1
Bit 4	OCCS	0x0	rw0c	Ordinary channel conversion start flag This bit is set by hardware and cleared by software (writing 0). 0: No ordinary channel conversion started 1: Ordinary channel conversion has started
Bit 3	PCCS	0x0	rw0c	Preempted channel conversion start flag This bit is set by hardware and cleared by software (writing 0). 0: No preempted channel conversion started 1: Preempted channel conversion has started
Bit 2	PCCE	0x0	rw0c	Preempted channel end of conversion flag This bit is set by hardware and cleared by software (writing 0). 0: Conversion is not complete 1: Conversion is complete
Bit 1	OCCE	0x0	rw0c	End of conversion flag This bit is set by hardware. It is cleared by software (writing 0) or by reading the ADC_ODT register. 0: Conversion is not complete 1: Conversion is complete
Bit 0	VMOR	0x0	rw0c	Voltage monitoring out of range flag This bit is set by hardware and cleared by software (writing 0). 0: Voltage is within the value programmed 1: Voltage is outside the value programmed

19.6.2 ADC control register 1 (ADC_CTRL1)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 29	Reserved	0x0	resd	Kept at its default value.
Bit 28	TCFACA	0x0	rw	Trigger conversion fail auto conversion abort 0: Disabled 1: Enabled Note: This feature can be enabled only in non-combined mode and when the TCFIEN is enabled at the same time.
Bit 27	TCFIEN	0x0	rw	Trigger conversion fail interrupt enable 0: Disabled 1: Enabled
Bit 26	OCCOIEN	0x0	rw	Ordinary channel conversion overflow interrupt enable 0: Ordinary channel conversion overflow interrupt disabled 1: Ordinary channel conversion overflow interrupt
Bit 25: 24	CRSEL	0x0	rw	Conversion resolution select 00: 12-bit 01: 10-bit 10: 8-bit 11: 6-bit
Bit 23	OCVMEN	0x0	rw	Voltage monitoring enable on ordinary channels 0: Voltage monitoring disabled on ordinary channels 1: Voltage monitoring enabled on ordinary channels
Bit 22	PCVMEN	0x0	rw	Voltage monitoring enable on preempted channels 0: Voltage monitoring disabled on preempted channels 1: Voltage monitoring enabled on preempted channels
Bit 21: 16	Reserved	0x0	resd	Kept at its default value.
Bit 15: 13	OCPCNT	0x0	rw	Partitioned mode conversion count of ordinary channels 000: 1 channel 001: 2 channels …… 111: 8 channels Note: In this mode, the preempted group converts only one channel at each trigger.
Bit 12	PCPEN	0x0	rw	Partitioned mode enable on preempted channels 0: Partitioned mode disabled on preempted channels 1: Partitioned mode enabled on preempted channels
Bit 11	OCPEN	0x0	rw	Partitioned mode enable on ordinary channels This is set and cleared by software to enable or disable partitioned mode on ordinary channels. 0: Partitioned mode disabled on ordinary channels 1: Partitioned mode enabled on ordinary channels
Bit 10	PCAUTOEN	0x0	rw	Preempted group automatic conversion enable after ordinary group 0: Preempted group automatic conversion disabled 1: Preempted group automatic conversion enabled
Bit 9	VMSGEN	0x0	rw	Voltage monitoring enable on a single channel 0: Disabled (Voltage monitoring enabled on all channels) 1: Enabled (Voltage monitoring enabled a single channel)

Bit 8	SQEN	0x0	rw	Sequence mode enable 0: Sequence mode disabled (a single channel is converted) 1: Sequence mode enabled (the selected multiple channels are converted) Note: If this mode is enabled and the OCCEIEN/PCCEIEN is set, an OCCE or PCCE interrupt is generated only at the end of conversion of the last channel.
Bit 7	PCCEIEN	0x0	rw	Conversion end interrupt enable on Preempted channels 0: Conversion end interrupt disabled on Preempted channels 1: Conversion end interrupt enabled on Preempted channels
Bit 6	VMORIEN	0x0	rw	Voltage monitoring out of range interrupt enable 0: Voltage monitoring out of range interrupt disabled 1: Voltage monitoring out of range interrupt enabled
Bit 5	CCEIEN	0x0	rw	Channel conversion end interrupt enable 0: Channel conversion end interrupt disabled 1: Channel conversion end interrupt enabled
Bit 4: 0	VMCSEL	0x00	rw	Voltage monitoring channel select This filed is valid only when the VMSGEN is enabled. 00000: ADC_IN0 channel 00001: ADC_IN1 channel …… 01111: ADC_IN15 channel 10000: ADC_IN16 channel 10001: ADC_IN17 channel 10010: ADC_IN18 channel 10010~11111: Unused, configuration is not allowed.

19.6.3 ADC control register 2 (ADC_CTRL2)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 30: 26	Reserved	0x00	resd	Kept at its default value
Bit 30	OCSWTRG	0x0	rw	Conversion of ordinary channels triggered by software 0: Conversion of ordinary channels not triggered 1: Conversion of ordinary channels triggered (This bit is cleared by software or by hardware as soon as the conversion starts)
Bit 29: 28	OCETE	0x0	rw	Ordinary channel external trigger edge select 00: Edge trigger forbidden 01: Rising edge 01: Falling edge 11: Any edge
Bit 31 Bit 27: 24	OCTESEL	0x0	rw	Ordinary channel conversion trigger event select Note: Refer to section 19.4.1.1 for details on bits.
Bit 22	PCSWTRG	0x0	rw	Conversion of preempted channels triggered by software 0: Conversion of preempted channels not triggered 1: Conversion of preempted channels triggered (This bit is cleared by software or by hardware as soon as the conversion starts)

Bit 21: 20	PCETE	0x0	rw	Preempted channel external trigger edge select 00: Edge trigger forbidden 01: Rising edge 01: Falling edge 11: Any edge
Bit 23 Bit 19: 16	PCTESEL	0x0	rw	Preempted channel conversion trigger event select Note: Refer to section 19.4.1.1 for details on bits.
Bit 15: 12	Reserved	0x0	resd	Kept at its default value.
Bit 11	DTALIGN	0x0	rw	Data alignment 0: Right alignment 1: Left alignment
Bit 10	EOCSFEN	0x0	rw	Each ordinary channel conversion OCCE flag enable) 0: Disabled 1: Enabled Note: Overflow detection is enabled automatically when this bit is set.
Bit 9	OCDRCEN	0x0	rw	Ordinary channel DMA request continue enable for independent mode) 0: Disabled (After the completion of the programmed number of DMA transfers, no DMA request generated at the end of ordinary conversion) 1: Enabled (Don't care about the programmed number of DMA transfers, Each ordinary channel sends DMA request at the end of ordinary conversion) Note: This bit is set only in non-master/slave mode with OCDMAEN = 1.
Bit 8	OCDMAEN	0x0	rw	DMA transfer enable of ordinary channels 0: Disabled 1: Enabled
Bit 7: 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	ADABRT	0x0	rw	ADC conversion abort 0: ADC conversion is not aborted 1: ADC conversion is aborted Note: This bit is cleared by hardware after ADC conversion is aborted. After this bit is cleared, re-triggering ADC conversion is allowed. If the ADABRT is set, it is necessary to wait until it is cleared before starting related ADC opertions.
Bit 3	ADCALINIT	0x0	rw	Initialize A/D calibration This bit is set by software and cleared by hardware. It is cleared after the calibration registers are initialized. 0: No initialization occurred or initialization completed 1: Enable initialization or initializations is ongoing
Bit 2	ADCAL	0x0	rw	A/D Calibration 0: No calibration occurred or calibration completed 1: Enable calibration or calibration is in process

Bit 1	RPEN	0x0	rw	Repetition mode enable 0: Repetition mode disabled When SQEN=0, a single conversion is done each time when a trigger event arrives; when SQEN=1, a group of repetition is done each timer when a trigger event arrives. 1: Repetition mode enabled When SQEN =0, continuous conversion mode on a single channel is enabled at each trigger event; when SQEN =1, continuous conversion mode on a group of channels is enabled at each trigger event.
Bit 0	ADCEN	0x0	rw	A/D converter enable 0: A/D converter disabled (ADC goes to power-down mode) 1: A/D converter enabled Note: When this bit is in OFF state, write an ON command can wake up The ADC from power-down mode. When this bit in ON state, write another ON command can start a regular group conversion. The application should pay attention to the fact that there is a delay of t_{STAB} between power on and start of conversion.

19.6.4 ADC sampling time register 1 (ADC_SPT1)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 27	Reserved	0x00	resd	Kept at its default value.
Bit 26: 24	CSPT18	0x0	rw	Sample time selection of channel ADC_IN18 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 23: 21	CSPT17	0x0	rw	Sample time selection of channel ADC_IN17 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 20: 18	CSPT16	0x0	rw	Sample time selection of channel ADC_IN16 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles

Bit 17: 15	CSPT15	0x0	rw	Sample time selection of channel ADC_IN15 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 14: 12	CSPT14	0x0	rw	Sample time selection of channel ADC_IN14 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 11: 9	CSPT13	0x0	rw	Sample time selection of channel ADC_IN13 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 8: 6	CSPT12	0x0	rw	Sample time selection of channel ADC_IN12 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 5: 3	CSPT11	0x0	rw	Sample time selection of channel ADC_IN11 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 2: 0	CSPT10	0x0	rw	Sample time selection of channel ADC_IN10 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles

19.6.5 ADC sampling time register 2 (ADC_SPT2)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value
Bit 29: 27	CSPT9	0x0	rw	Sample time selection of channel ADC_IN9 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 26: 24	CSPT8	0x0	rw	Sample time selection of channel ADC_IN8 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 23: 21	CSPT7	0x0	rw	Sample time selection of channel ADC_IN7 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 20: 18	CSPT6	0x0	rw	Sample time selection of channel ADC_IN6 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 17: 15	CSPT5	0x0	rw	Sample time selection of channel ADC_IN5 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles

Bit 14: 12	CSPT4	0x0	rw	Sample time selection of channel ADC_IN4 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 11: 9	CSPT3	0x0	rw	Sample time selection of channel ADC_IN3 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 8: 6	CSPT2	0x0	rw	Sample time selection of channel ADC_IN2 000: Reserved 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 5: 3	CSPT1	0x0	rw	Sample time selection of channel ADC_IN1 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles
Bit 2: 0	CSPT0	0x0	rw	Sample time selection of channel ADC_IN0 000: 2.5 cycles 001: 6.5 cycles 010: 12.5 cycles 011: 24.5 cycles 100: 47.5 cycles 101: 92.5 cycles 110: 247.5 cycles 111: 640.5 cycles

19.6.6 ADC preempted channel data offset register x (ADC_ PCDTOx) (x=1..4)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value
Bit 11: 0	PCDTOx	0x000	rw	Data offset for Preempted channel x Converted data stored in the ADC_PDTx = Raw converted data – ADC_PCDTOx

19.6.7 ADC voltage monitor high threshold register (ADC_VWHB)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x00000	resd	Kept at its default value
Bit 15: 0	VMHB	0xFFFF	rw	Voltage monitoring high boundary

19.6.8 ADC voltage monitor low threshold register (ADC_ VWLB)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x00000	resd	Kept at its default value
Bit 15: 0	VMLB	0x0000	rw	Voltage monitoring low boundary

19.6.9 ADC ordinary sequence register 1 (ADC_ OSQ1)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x00	resd	Kept at its default value
Bit 23: 20	OCLEN	0x0	rw	Ordinary conversion sequence length 0000: 1 conversion 0001: 2 conversions …… 1111: 16 conversions
Bit 19: 15	OSN16	0x00	rw	Number of 16th conversion in ordinary sequence
Bit 14: 10	OSN15	0x00	rw	Number of 15th conversion in ordinary sequence
Bit 9: 5	OSN14	0x00	rw	Number of 14th conversion in ordinary sequence
Bit 4: 0	OSN13	0x00	rw	Number of 13th conversion in ordinary sequence Note: The number can be from 0 to 17. For example, if the number is set to 3, it means that the 13th conversion is ADC_IN3 channel.

19.6.10 ADC ordinary sequence register 2 (ADC_ OSQ2)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value
Bit 29: 25	OSN12	0x00	rw	Number of 12th conversion in ordinary sequence
Bit 24: 20	OSN11	0x00	rw	Number of 11th conversion in ordinary sequence
Bit 19: 15	OSN10	0x00	rw	Number of 10th conversion in ordinary sequence
Bit 14: 10	OSN9	0x00	rw	Number of 9th conversion in ordinary sequence

Bit 9: 5	OSN8	0x00	rw	Number of 8th conversion in ordinary sequence
Bit 4: 0	OSN7	0x00	rw	Number of 7th conversion in ordinary sequence Note: The number can be from 0 to 17. For example, if the number is set to 8, it means that the 7th conversion is ADC_IN8 channel.

19.6.11 ADC ordinary sequence register 3 (ADC_ OSQ3)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value
Bit 29: 25	OSN6	0x00	rw	Number of 6th conversion in ordinary sequence
Bit 24: 20	OSN5	0x00	rw	Number of 5th conversion in ordinary sequence
Bit 19: 15	OSN4	0x00	rw	Number of 4th conversion in ordinary sequence
Bit 14: 10	OSN3	0x00	rw	Number of 3rd conversion in ordinary sequence
Bit 9: 5	OSN2	0x00	rw	Number of 2nd conversion in ordinary sequence
Bit 4: 0	OSN1	0x00	rw	Number of 1st conversion in ordinary sequence Note: The number can be from 0 to 18. For example, if the number is set to 8, it means that the 1st conversion is ADC_IN17 channel.

19.6.12 ADC preempted sequence register (ADC_ PSQ)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value
Bit 21: 20	PCLEN	0x0	rw	Preempted conversion sequence length 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions
Bit 19: 15	PSN4	0x00	rw	Number of 4th conversion in preempted sequence
Bit 14: 10	PSN3	0x00	rw	Number of 3rd conversion in preempted sequence
Bit 9: 5	PSN2	0x00	rw	Number of 2nd conversion in preempted sequence
Bit 4: 0	PSN1	0x00	rw	Number of 1st conversion in preempted sequence *Note: The number can be from 0 to 18. For example, if the number is set to 3, it refers to the ADC_IN3 channel.* *If PCLEN is less than 4, the conversion sequence starts from 4-PCLEN. For example, when ADC_PSQ ([21: 0]) =10 00110 00101 00100 00011, it indicates that the scan conversion follows the sequence: 4, 5, 6, not 3, 4,5.*

19.6.13 ADC preempted data register x (ADC_ PDTx) (x=1..4)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 0	PDTx	0x0000	ro	Conversion data from preempted channel

19.6.14 ADC ordinary data register (ADC_ ODT)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	ODT	0x0000	ro	Conversion data of ordinary channel

19.6.15 ADC oversampling register (ADC_ OVSP)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 11	Reserved	0x000000	resd	Kept at its default value.
Bit 10	OOSRSEL	0x0	rw	Ordinary oversampling restart mode select When the ordinary oversampling is interrupted by preempted conversions, this bit can be used to select where to resume ordinary conversions. 0: Continuous mode (ordinary oversampling buffer will be reserved) 1: Restart mode (ordinary oversampling buffer will be cleared, that is, the previously oversampled times are reset)
Bit 9	OOSTREN	0x0	rw	Ordinary oversampling trigger mode enable 0: Disabled (only one trigger is needed for all oversampling conversions) 1: Enabled (Each oversampling conversion needs a trigger)
Bit 8: 5	OSSSEL	0x0	rw	Oversampling shift select This field is used to define the number of right-shift used in the oversampling results. 0000: No shift 0001: 1 bit 0010: 2 bits 0011: 3 bits 0100: 4 bits 0101: 5 bits 0110: 6 bits 0111: 7 bits 1000: 8 bits 1001~1111: Unused. Do not configure.
Bit 4: 2	OSRSEL	0x0	rw	Oversampling ratio select 000: 2x 001: 4x 010: 8x 011: 16x 100: 32x 101: 64x 110: 128x 111: 256x
Bit 1	POSEN	0x0	rw	Preempted oversampling enable 0: Preempted oversampling disabled 1: Preempted oversampling enabled
Bit 0	OOSEN	0x0	rw	Ordinary oversampling enable 0:Ordinary oversampling disabled 1: Ordinary oversampling enabled

19.6.16 ADC common status register (ADC_CSTS)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15	TCF2	0x0	ro	ADC2 trigger convert fail flag This is the mapping bit of TCF bit in the ADC2_STS register.
Bit 14	RDY2	0x0	ro	ADC2 conversion ready flag This bit is the RDY mapping bit of the ADC2_STS register.
Bit 13	OCCO2	0x0	ro	Ordinary channel conversion overflow flag of ADC2 This bit is the OCCO mapping bit of the ADC2_STS register.
Bit 12	OCCS2	0x0	ro	Ordinary channel conversion start flag of ADC2 This bit is the OCCS mapping bit of the ADC2_STS register.
Bit 11	PCCS2	0x0	ro	Preempted channel conversion start flag of ADC2 This bit is the PCCS mapping bit of the ADC2_STS register.
Bit 10	PCCE2	0x0	ro	Preempted channels conversion end flag of ADC2 This bit is the PCCE mapping bit of the ADC2_STS register.
Bit 9	OCCE2	0x0	ro	Ordinary channels conversion end flag of ADC2 This bit is the OCCE mapping bit of the ADC2_STS register.
Bit 8	VMOR2	0x0	ro	Voltage monitoring out of range flag of ADC2 This bit is the VMOR mapping bit of the ADC2_STS register.
Bit 7	TCF1	0x0	ro	ADC1 trigger conversion fail flag This bit is the TCF mapping bit of the ADC1_STS register.
Bit 6	RDY1	0x0	ro	ADC1 conversion ready flag This bit is the RDY mapping bit of the ADC1_STS register.
Bit 5	OCCO1	0x0	ro	Ordinary channel conversion overflow flag of ADC1 This bit is the OCCO mapping bit of the ADC1_STS register.
Bit 4	OCCS1	0x0	ro	Ordinary channel conversion start flag of ADC1 This bit is the OCCS mapping bit of the ADC1_STS register.
Bit 3	PCCS1	0x0	ro	Preempted channel conversion start flag of ADC1 This bit is the PCCS mapping bit of the ADC1_STS register.
Bit 2	PCCE1	0x0	ro	Preempted channels conversion end flag of ADC1 This bit is the PCCE mapping bit of the ADC1_STS register.
Bit 1	OCCE1	0x0	ro	Ordinary channels conversion end flag of ADC1 This bit is the OCCE mapping bit of the ADC1_STS register.
Bit 0	VMOR1	0x0	ro	Voltage monitoring out of range flag of ADC1 This bit is the VMOR mapping bit of the ADC1_STS register.

19.6.17 ADC common control register (ADC_CCTRL)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x0000	resd	Kept at its default value.
Bit 23	ITSRVEN	0x0	rw	Internal temperature sensor and V_{INTRV} enable 0: Disabled 1: Enabled
Bit 22	VBATEN	0x0	rw	V_{BAT} enable 0: Disabled 1: Enabled
Bit 21: 20	Reserved	0x0	resd	Kept at its default value.
Bit 19: 16	ADCDIV	0x0	rw	ADC division 0000: HCLK/2 0001: HCLK/3 ...

				1111: HCLK/17 Note: The clock divided by this field are used by all ADCs. The maximum value of the ADCCLK is 80 MHz. After this division, the ADCCLK cannot be higher than PCLK2.
Bit 28 Bit 15: 14	MSDMASEL	0x0	rw	Ordinary channel DMA transfer mode select in master/ slave mode MSDMASEL[2] is the 28th bit in the ADC_CCTRL register. MSDMASEL[2: 0] is defined as follows: 000: No DMA transfer 001: DMA mode 1 010: DMA mode 2 011: DMA mode 3 110~111: Unused. Do not configure. Note: This field is applicable in master/slave mode. Refer to Section 19.5.1 for details on DMA mode x.
Bit 13	MSDRCEN	0x0	rw	Ordinary channel DMA request continuation enable in master/slave mode 0: Disabled (After the completion of the programmed number of DMA transfers, no DMA request generated at the end of ordinary conversion) 1: Enabled (Don't care about the programmed number of DMA transfers, Each ordinary channel sends DMA request at the end of ordinary conversion) Note: This bit is applicable in master/slave mode and when DMA mode x is selected through the MSDMASEL bit.
Bit 12	Reserved	0x0	resd	Kept at its default value.
Bit 11: 8	ASISEL	0x0	rw	Adjacent ADC sampling interval select in ordinary shift mode 0000: 5 * TADCCLK 0001: 6 * TADCCLK 0010: 7 * TADCCLK ... 1111: 20 * TADCCLK Note: This field is used to configure the conversion interval between adjacent ADCs in ordinary shift mode. Refer to Section 19.5.4 for details.
Bit 7: 5	Reserved	0x0	resd	Kept at its default value.
Bit 4: 0	MSSEL	0x00	rw	Combined master/slave mode select 00000: Non-combined mode 00001: Combined ordinary simultaneous+preempted simultaneous modes (single slave) 00010: Combined ordinary simultaneous+alternate preempted trigger modes (single slave) 00011~00100: Unused. Do not configure 00101: Preempted simultaneous mode (single slave) 00110: Ordinary simultaneous mode (single slave) 00111: Ordinary shift mode (single slave) 01000: Unused. Do not configure 01001: Alternate preempted trigger mode (single slave) 01010~11111: Unused. Do not configure Note: In combined master/slave mode, the change of configuration will cause loss of synchronization between master and slave. Thus it is recommended to disable combined master/slave mode before changing.

19.6.18 ADC common data register (ADC_CODT)

Accessed by words.

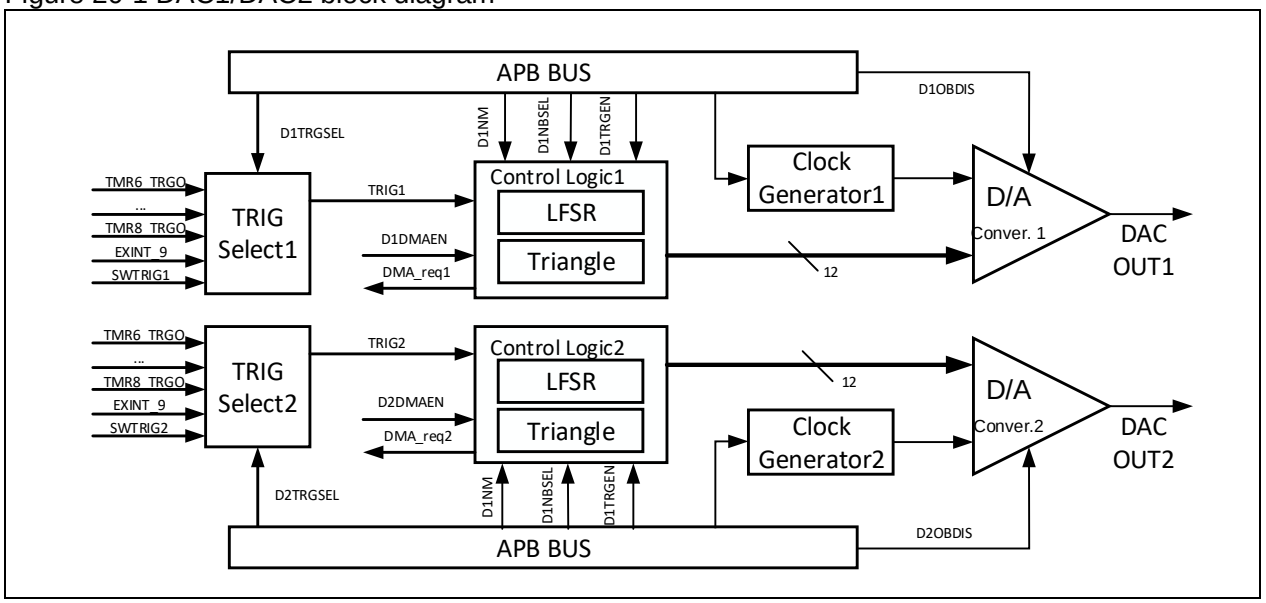
Bit	Name	Reset value	Type	Description
Bit 31: 16	CODTH	0x0000	rw	Ordinary conversion high halfword data in master slave mode Note: The meanings of data in this field vary from DMA mode to DMA mode. Refer to Section 19.5.1 for details.
Bit 15: 0	CODTL	0x0000	rw	Ordinary conversion low halfword data in master slave mode Note: The meanings of data in this field vary from DMA mode to DMA mode. Refer to Section 19.5.1 for details.

20 Digital-to-analog converter (DAC)

20.1 DAC introduction

The DAC uses a 12-bit digital input to generate an analog output between 0 and reference voltage. The digital part of the DAC can be configured in 8-bit or 12-bit mode and can be used in conjunction with the DMA. It supports left or right alignment in a single /dual DAC modes. It has two output channels, DAC1 and DCA2, with its own converter each. Each DAC1/DAC2 can be converted independently or simultaneously in dual DAC mode. The input reference voltage VINTRV makes conversion more accuracy.

Figure 20-1 DAC1/DAC2 block diagram



20.2 DAC main features

- A single/dual DAC 8-bit or 12-bit digital input
- Left or right data alignment
- Noise-wave/Triangular-wave generation
- Dual DAC or single DAC1/DAC2 independent conversions
- DMA mode for DAC1/DAC2
- Software or external triggers for conversion
- Input reference voltage VINTRV

20.3 Design tips

The following information can be used as DAC design reference:

- Analog module configuration
  The analog part of the DAC1/DAC2 can be enabled by setting the ENx bit in the DAC_CTRL register, but its digital part is not subject to this bit. The DAC integrates two output gains that can be used to reduce the output impedance, and to drive external loads directly without the need of an external operational amplifier. The DAC1/DAC2 output gain can be enabled and disabled through the DxOBDIS bit in the DAC_CTRL register.
- DMA capability
  The DAC1/DAC2 both have a DMA capability that can be enabled by setting the DxDMAEN bit in the DAC_CTRL register. One DMA request is generated when a trigger signal is active while the DxTRGEN bit is set. The DAC DMA request is not added up, meaning the new DAM request will be ignored and no error is reported.
  In dual DAC mode, the application can handle two channels (DAC1/DAC2) by using only one DMA request and a DMA channel.
- DMA underflow
  When the DAC DMA request is enabled, an overflow occurs if a second external trigger arrives before the acknowledgement for the first external trigger is received. In this case, no new external trigger is handled, or no new DMA request is issued, and the DxDMAUDRF bit in the DAC_SR register is set, reporting the error condition. An interrupt is generated if its corresponding DxDMAUDRIEN bit in the DAC_CTRL register is set.
  The software clears the DxDMAUDRF bit by writing 1. The DxDMAEN bit should be cleared in the DAC_CTRL register before re-starting a DMA transfer and re-initializing DMA and DAC.
- Input/output configuration
  The digital inputs are linearly converted to analog voltage outputs by the DAC, and it is between 0 and reference voltage. The analog DAC module is supplied by VDDA. The positive analog reference voltage input falls between 2.0 V and VDDA. To avoid parasitic interruption and excessive consumption, the PA4 or PA5 should be configured to analog input.
  DAC output = VINTRV x (DxODT[11: 0]/4095)

20.4 Functional overview

20.4.1 Trigger events

If the DxTRGEN bit in the DAC_CTRL register is set, the DAC conversion can then be triggered by an external event (timer counter, external interrupt line) or by software. The DxTRGSEL[2: 0] is used to select trigger sources.

Table 20-1 Trigger source selection

Source	DxTRGSEL [2:0]	Description
TMR6_TRGOUT	000	On-chip signals
TMR8_TRGOUT	001	
TMR7_TRGOUT	010	
TMR5_TRGOUT	011	
TMR2_TRGOUT	100	
TMR4_TRGOUT	101	
EXINT_9	110	External signals
DxSWTRG	111	Software trigger

When the DxTRGEN bit is set, the data stored into the HDRx register is transferred into the DAC_DxODT register each time a DAC detects an active trigger event,. If the software trigger is selected, the DxSWTRG flag is cleared by hardware after writing 1. The DAC output becomes active after a period of time once the data is loaded into the DAC_DxODT register.

When the DxTRGEN bit is cleared, each data written to the data register is immediately transferred into the DAC_DxODT register without the need of a trigger event.

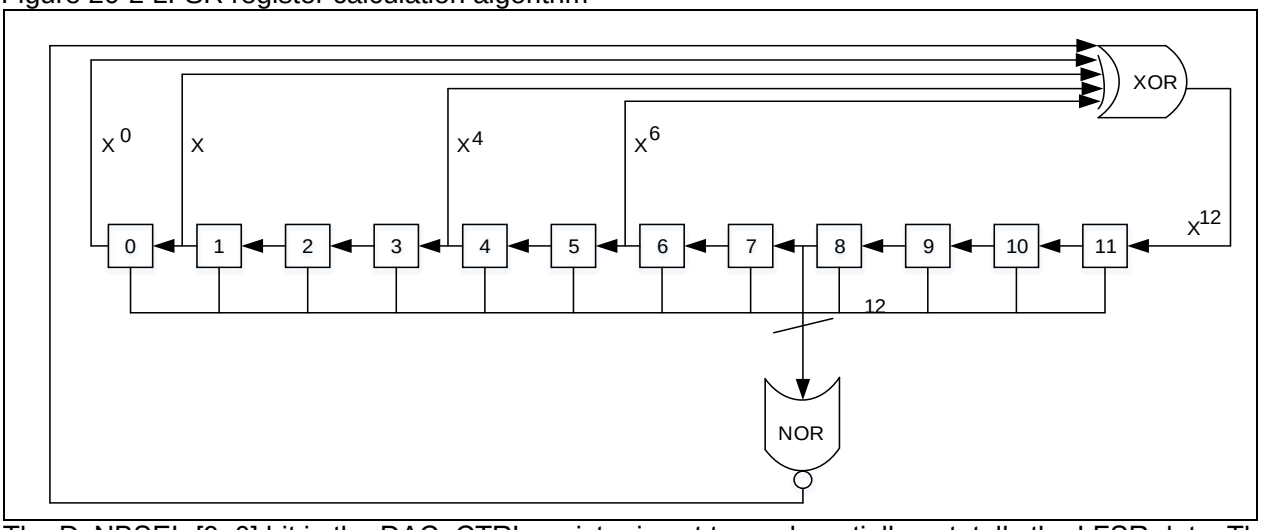
20.4.2 Noise/Triangular-wave generation

The DAC can output a variable-amplitude pseudo noise and a triangular wave, which is done by the LENinear Feedback Shift Register and triangle wave generator respectively. The DAC variable-amplitude pseudo noise generation is selected by setting DxNM[1:0]=01 in the LFSR, while the DAC triangular-wave generation is selected by setting the DxNM[1:0]=1x.

LFSR logic

The preloaded value in the LFSR is 0xAAA. This register is updated after each trigger event based on a specific calculation algorithm.

Figure 20-2 LFSR register calculation algorithm



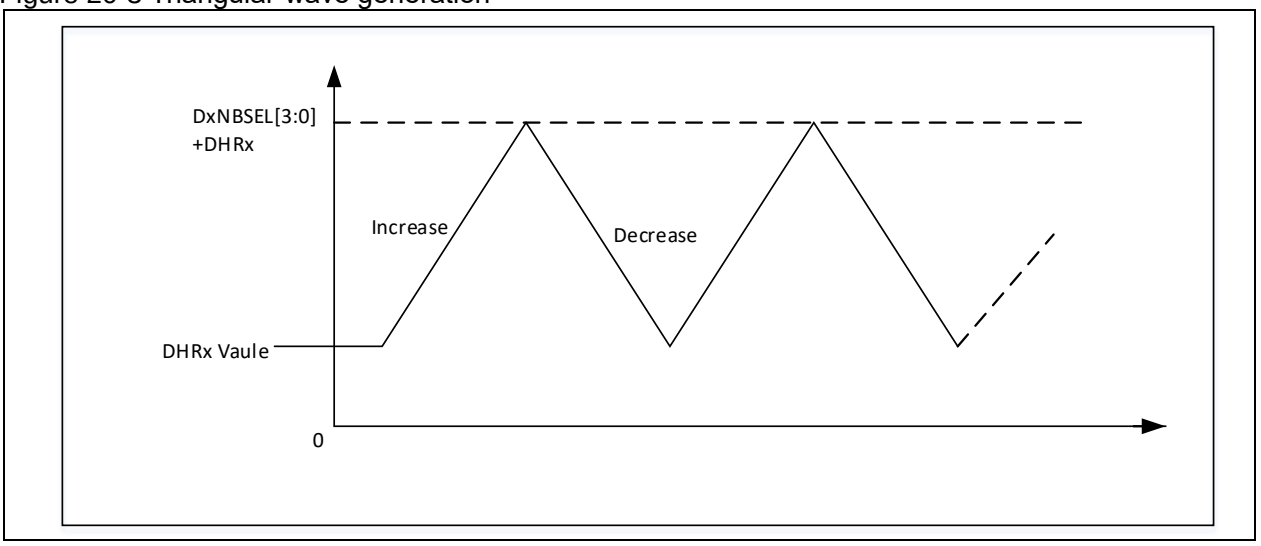
The DxNBSEL [3: 0] bit in the DAC_CTRL register is set to mark partially or totally the LFSR data. The resulting value is then added up to the DHRx value without overflow and this value is loaded into the DAC_DxODT register. It is possible to disable LFSR function and reset LFSR wave generation algorithm by setting DxNM[1: 0]=00.

Triangular wave logic

The DAC triangular-wave generation is selected by setting DxNM[1: 0]=1x. The amplitude is configured through the DxNBSEL [3: 0] bit in the DAC_CTRL register. An internal triangular-wave counter is incremented at each trigger event. Once the maximum amplitude programmed in the DxNBSEL [3: 0] is reached, the value of this counter is decremented down to 0, then incremented again, and so on.

Meanwhile, the value of this counter is then added up to the DHRx register without overflow and the resulting value is loaded into the DAC_DxODT register. It is possible to disable/reset the triangular-wave generation by setting DxNM[1: 0]=00.

Figure 20-3 Triangular-wave generation



20.4.3 DAC data alignment

The DAC supports a single DAC and dual DA mode. The data format is dependent on the selected configuration mode.

Single DAC data format:

8-bit right alignment: load data into the DAC_DxDTH8R [7:0]

12-bit left alignment: load data into the DAC_DxDTH12L [15: 4]

12-bit right alignment: load data in the DAC_DxDTH12R [11: 0]

Dual DAC data format:

8-bit right alignment: load data into the DAC_DDTH8R [7: 0] and DAC_DDTH8R [15: 8]

12-bit left alignment: load data into the DAC_DDTH12L [15: 4] and DAC_DDTH12L [31: 20]

12-bit right alignment: load data into the DAC_DDTH12R [11: 0] and DAC_DDTH12R [27:16]

The loaded 8-bit data corresponds to the DHRx[11:4] and the loaded 12-bit data corresponds to the DHRx[11: 0]

20.5 DAC registers

These peripheral registers must be accessed by words (32 bits).

Table 20-2 DAC register map and reset values

Register	Offset	Reset value
DAC_CTRL	000h	0x0000 0000
DAC_SWTRG	004h	0x0000 0000
DAC_D1DTH12R	008h	0x0000 0000
DAC_D1DTH12L	00Ch	0x0000 0000
DAC_D1DTH8R	010h	0x0000 0000
DAC_D2DTH12R	014h	0x0000 0000
DAC_D2DTH12L	018h	0x0000 0000
DAC_D2DTH8R	01Ch	0x0000 0000
DAC_DDTH12R	020h	0x0000 0000
DAC_DDTH12L	024h	0x0000 0000
DAC_DDTH8R	028h	0x0000 0000
DAC_D1ODT	02Ch	0x0000 0000
DAC_D2ODT	030h	0x0000 0000

DAC_STS	034h	0x0000 0000

20.5.1 DAC control register (DAC_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value
Bit 29	D2DMAUDRIEN	0x0	rw	DAC2 DMA transfer underrun interrupt enable This bit is set and cleared by software. 0: DAC2 DMA transfer underrun interrupt disabled 1: DAC2 DMA transfer underrun interrupt enabled
Bit 28	D2DMAEN	0x0	rw	DAC2 DMA transfer enable This bit is set and cleared by software. 0: DAC2 DMA mode disabled 1: DAC2 DMA mode enabled
Bit 27: 24	D2NBSEL	0x0	rw	DAC2 noise bit select These bits are used to select the mark bit in noise generation mode or amplitude in triangular-wave generation mode. 0000: Unmask LSFR bit0 /Triangle amplitude is equal to 1 0001: Unmask LSFR bit[1: 0] /Triangle amplitude is equal to 3 0010: Unmask LSFR bit[2: 0] /Triangle amplitude is equal to 7 0011: Unmask LSFR bit[3: 0] /Triangle amplitude is equal to 15 0100: Unmask LSFR bit[4: 0] /Triangle amplitude is equal to 31 0101: Unmask LSFR bit[5: 0] /Triangle amplitude is equal to 63 0110: Unmask LSFR bit[6: 0] /Triangle amplitude is equal to 127 0111: Unmask LSFR bit[7: 0] /Triangle amplitude is equal to 255 1000: Unmask LSFR bit[8: 0] /Triangle amplitude is equal to 511 1001: Unmask LSFR bit[9: 0] /Triangle amplitude is equal to 1023 1010: Unmask LSFR bit[10:0] /Triangle amplitude is equal to 2047 ≥1011: Unmask LSFR bit[11: 0] /Triangle amplitude is equal to 4095 Note: This bit field is effective only when D2TRGEN = 1.
Bit 23: 22	D2NM	0x0	rw	DAC2 noise mode 00: Wave generation disabled 01: Noise wave generation enabled 1x: Triangular wave generation enabled
Bit 21: 19	D2TRGSEL	0x0	rw	DAC2 trigger select 000: TMR6 TRGOUT event 001: TMR8 TRGOUT event 010: TMR7 TRGOUT event 011: TMR5 TRGOUT event 100: TMR2 TRGOUT event 101: TMR4 TRGOUT event 110: External interrupt line 9 111: Software trigger Note: These bit field is effective only when D2TRGEN = 1.
Bit 18	D2TRGEN	0x0	rw	DAC2 trigger enable 0: DAC2 trigger disabled 1: DAC2 trigger enabled Note: When the DAC2 trigger is disabled, the data written into the DAC_D2DTHx register is transferred into the DAC_D2ODT register after one APB1 clock cycle.

				When the DAC2 trigger is enabled, the data written into the DAC_D2DTHx register is transferred into the DAC_ D2ODT register after three APB1 clock cycles. If the software trigger is selected, it takes one APB1 clock cycle to have the data written into the DAC_D2DTHx register transferred into the DAC_D2ODT register.
Bit 17	D2OBDIS	0x0	rw	DAC2 output buffer disable 0: DAC2 output buffer enabled 1: DAC2 output buffer disabled
Bit 16	D2EN	0x0	rw	DAC2 enable 0: DAC2 disabled 1: DAC2 enabled
Bit 15: 14	Reserved	0x0	resd	Kept at its default value
Bit 13	D1DMAUDRIEN	0x0	rw	DAC1 DMA transfer underrun interrupt enable This bit is set and cleared by software. 0: DAC1 DMA transfer underrun interrupt disabled 1: DAC1 DMA transfer underrun interrupt enabled
Bit 12	D1DMAEN	0x0	rw	DAC1 DMA transfer enable 0: DAC1 DMA transfer disabled 1: DAC1 DMA transfer enabled
Bit 11: 8	D1NBSEL	0x0	rw	DAC1 noise bit select These bits are used to select the mark bit in noise generation mode or amplitude in triangular-wave generation mode. 0000: Unmask LSFR bit0/Triangle amplitude is equal to 1 0001: Unmask LSFR bit[1:0]/Triangle amplitude is equal to 3 0010: Unmask LSFR bit[2: 0]/Triangle amplitude is equal to 7 0011: Unmask LSFR bit[3: 0]/Triangle amplitude is equal to 15 0100: Unmask LSFR bit[4: 0]/Triangle amplitude is equal to 31 0101: Unmask LSFR bit[5: 0]/Triangle amplitude is equal to 63 0110: Unmask LSFR bit[6: 0]/Triangle amplitude is equal to 127 0111: Unmask LSFR bit[7: 0]/Triangle amplitude is equal to 255 1000: Unmask LSFR bit[8: 0]/Triangle amplitude is equal to 511 1001: Unmask LSFR bit[9: 0]/Triangle amplitude is equal to 1023 1010: Unmask LSFR bit[10: 0]/Triangle amplitude is equal to 2047 ≥1011: Unmask LSFR bit[11:0]/Triangle amplitude is equal to 4095 This bit field is effective when D1TRGEN= 1.
Bit 7: 6	D1NM	0x0	rw	DAC1 noise mode 00: Wave generation disabled 01: Noise wave generation enabled 1x: Triangular wave generation enabled
Bit 5: 3	D1TRGSEL	0x0	rw	DAC1 trigger select 000: TMR6 TRGOUT event 001: TMR8 TRGOUT event 010: TMR7 TRGOUT event 011: TMR5 TRGOUT event 100: TMR2 TRGOUT event 101: TMR4 TRGOUT event 110: External interrupt line 9 111: Software trigger Note: These bitfield is effective only when D1TRGEN = 1.
Bit 2	D1TRGEN	0x0	rw	DAC1 trigger enable 0: DAC1 trigger disabled

				1: DAC1 trigger enabled Note: When the DAC1 trigger is disabled, the data written into the DAC_D1DTHx register is transferred into the DAC_D1ODT register after one APB1 clock cycle. When the DAC1 trigger is enabled, the data written into the DAC_D1DTHx register is transferred into the DAC_ D1ODT register after three APB1 clock cycles If the software trigger is selected, it takes one APB1 clock cycle to have the data written into the DAC_D1DTHx register transferred into the DAC_ D1ODT register.
Bit 1	D1OBDIS	0x0	rw	DAC1 output buffer disable 0: DAC1 output buffer enabled 1: DAC1 output buffer disabled
Bit 0	D1EN	0x0	rw	DAC1 enable 0: DAC1 disabled 1: DAC1 enabled

20.5.2 DAC software trigger register (DAC_SWTRG)

Bit	Name	Reset value	Type	Description
Bit 31: 2	Reserved	0x0000 0000	resd	Kept at its default value
Bit 1	D2SWTRG	0x0	rw	DAC2 software trigger 0: DAC2 software trigger disabled 1: DAC2 software trigger enabled Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_D2DTH data is loaded into the DAC_D2ODT register.
Bit 0	D1SWTRG	0x0	rw	DAC1 software trigger 0: DAC1 software trigger disabled 1: DAC1 software trigger enabled Note: This bit is cleared by hardware (one APB1 clock cycle later) once the DAC_D1DTH data is loaded into the DAC_D1ODT register.

20.5.3 DAC1 12-bit right-aligned data holding register (DAC_ D1DTH12R)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value
Bit 11: 0	D1DT12R	0x000	rw	DAC1 12-bit right-aligned data

20.5.4 DAC1 12-bit left-aligned data holding register (DAC_ D1DTH12L)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 4	D1DT12L	0x000	rw	DAC1 12-bit left-aligned data
Bit 3: 0	Reserved	0x0	resd	Kept at its default value

20.5.5 DAC1 8-bit right-aligned data holding register (DAC_ D1DTH8R)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value
Bit 7: 0	D1DT8R	0x00	rw	DAC1 8-bit right-aligned data

20.5.6 DAC2 12-bit right-aligned data holding register (DAC_ D2DTH12R)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value
Bit 11: 0	D2DT12R	0x000	rw	DAC2 12-bit right-aligned data

20.5.7 DAC2 12-bit left-aligned data holding register (DAC_ D2DTH12L)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 4	D2DT12L	0x000	rw	DAC2 12-bit left-aligned data
Bit 3: 0	Reserved	0x0	resd	Kept at its default value

20.5.8 DAC2 8-bit right-aligned data holding register (DAC_ D2DTH8R)

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value
Bit 7: 0	D2DT8R	0x00	rw	DAC2 8-bit right-aligned data

20.5.9 Dual DAC 12-bit right-aligned data holding register (DAC_ DDTH12R)

Bit	Name	Reset value	Type	Description
Bit 31: 28	Reserved	0x0	resd	Kept at its default value
Bit 27: 16	DD2DT12R	0x000	rw	DAC2 12-bit right-aligned data
Bit 15: 12	Reserved	0x0	resd	Kept at its default value
Bit 11: 0	DD1DT12R	0x000	rw	DAC1 12-bit right-aligned data

20.5.10 Dual DAC 12-bit left-aligned data holding register (DAC_ DDTH12L)

Bit	Name	Reset value	Type	Description
Bit 31: 20	DD2DT12L	0x000	rw	DAC2 12-bit left-aligned data
Bit 19: 16	Reserved	0x0	resd	Kept at its default value
Bit 15: 4	DD1DT12L	0x000	rw	DAC1 12-bit left-aligned data
Bit 3: 0	Reserved	0x0	resd	Kept at its default value

20.5.11 Dual DAC 8-bit right-aligned data holding register (DAC_ DDTH8R)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value
Bit 15: 8	DD2DT8R	0x00	rw	DAC2 8-bit right-aligned data
Bit 7: 0	DD1DT8R	0x00	rw	DAC1 8-bit right-aligned data

20.5.12 DAC1 data output register (DAC_ D1ODT)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value
Bit 11: 0	D1ODT	0x000	rw	DAC1 output data

20.5.13 DAC2 data output register (DAC_ D2ODT)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value
Bit 11: 0	D2ODT	0x000	rw	DAC2 output data

20.5.14 DAC status register (DAC_STS)

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value
Bit 29	D2DMAUDRF	0x0	w1c	DAC2 DMA transfer underrun flag 0: No DAC2 DMA transfer underrun 1: DAC2 DMA transfer underrun occurs Note: This bit is cleared by writing 1.
Bit 28: 14	Reserved	0x0000	resd	Kept at its default value
Bit 13	D1DMAUDRF	0x0	w1c	DAC1 DMA transfer underrun flag 0: No DAC1 DMA transfer underrun 1: DAC1 DMA transfer underrun occurs Note: This bit is cleared by writing 1.
Bit 12: 0	Reserved	0x0000	resd	Kept at its default value

21 CAN

The AT32F456 and AT32F457 supports CANFD protocol but the AT32F455 not.

21.1 CAN overview

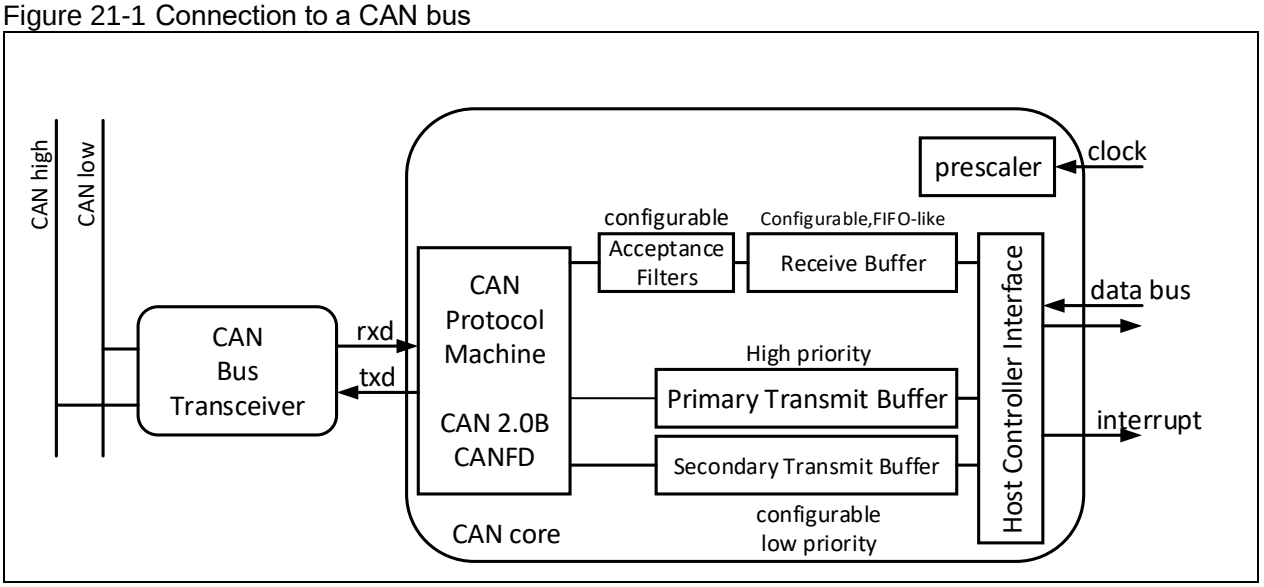
21.1.1 CAN core

The CAN controller core is a serial communications controller that performs serial communication according to the CAN protocol and meets all constrains of the CAN specificiations:

- Classic CAN2.0B
- CANFD (ISO 11898-1:2015)

The CAN protocol defines the transfer of frames (communication objects) between nodes of the network and the management of the error handling. The CAN controller core enables the user to set up economic and reliable links between various componets without any burden on the host controller for handling the CAN protocol. The CAN controller core appears to be a microcontroller as a memory-mapped I/O device. A CPU access CAN controller to control transmission or recpection of frames through a two wire CAN bus system. The connection to a CAN bus is illustrated in Figure 22-1.

Figure 21-1 Connection to a CAN bus



21.1.2 CAN protocol

CAN communication is organized in frames. Error frames and overload frames are handled fully automatically by CAN controller. Data frames are used to transmit data content from the host application. There are several types of data frames, see Figure 21-2 and Figure 21-3. Classic CAN2.0B is able to transmit up to 8 bytes of data payload, CAN FD up to 64 bytes using one data frame.

Data addressing is done using frame identifiers. In a CAN network, only one node shall transmit frames with a certain identifier. All notes receive all frames and the host controller of a node has to decide if a received frame was of interest. To reduce the load of a host controller, a CAN node may use acceptance filters. These filters compare all received frame identifiers to user-selectable bit patterns. Only if a frame passes an acceptance filter, it will be stored in the receive buffer and signaled to the host controller.

The identifiers of CAN frames are used for bus arbitration. A can protocol machine stos transmission of a frame with a low-priority identifier when a frame with a higher priority identifier is transmitted by another CAN node simultaneously. A CAN protocol machine automatically attempts to retransmit the stopped frame at the next possible transmit position.

CAN2.0B defnes data bit rates upt o 1 Mbit/s.For CAN FD, there is no limitation in theory, but a practical limitation by the electrical characteristics of the used transceivers and the bus topology. For CAN FD bit rate switching can be enabled. if enabled the transmission of the payload of frames can be done at higher speed while the frame header is transmitted at a lower speed.

21.1.3 Classic CAN2.0B and CANFD

The CAN protocol has a long history of protocol improvements over the years. The oldest version of the CAN protocol supported by the CAN is Classic CAN2.0B. The next step of evolution was CANFD (ISO11898-1:2015). This means that a CAN controller is able to receive and transmit all CAN FD and Classic CAN2.0B frames. Upward-compatibility is also supported by CAN controller.

The CAN controller can select a run-time for every frame if the frame will be transmitted as classic CAN2.0B or CAN FD frame. For received frames, the status bits in the receive buffer signal if a received frame was a classic CAN 2.0B (Figure 21-3) or CAN FD frame (Figure 21-2).

Figure 21-2 CANFD frame type

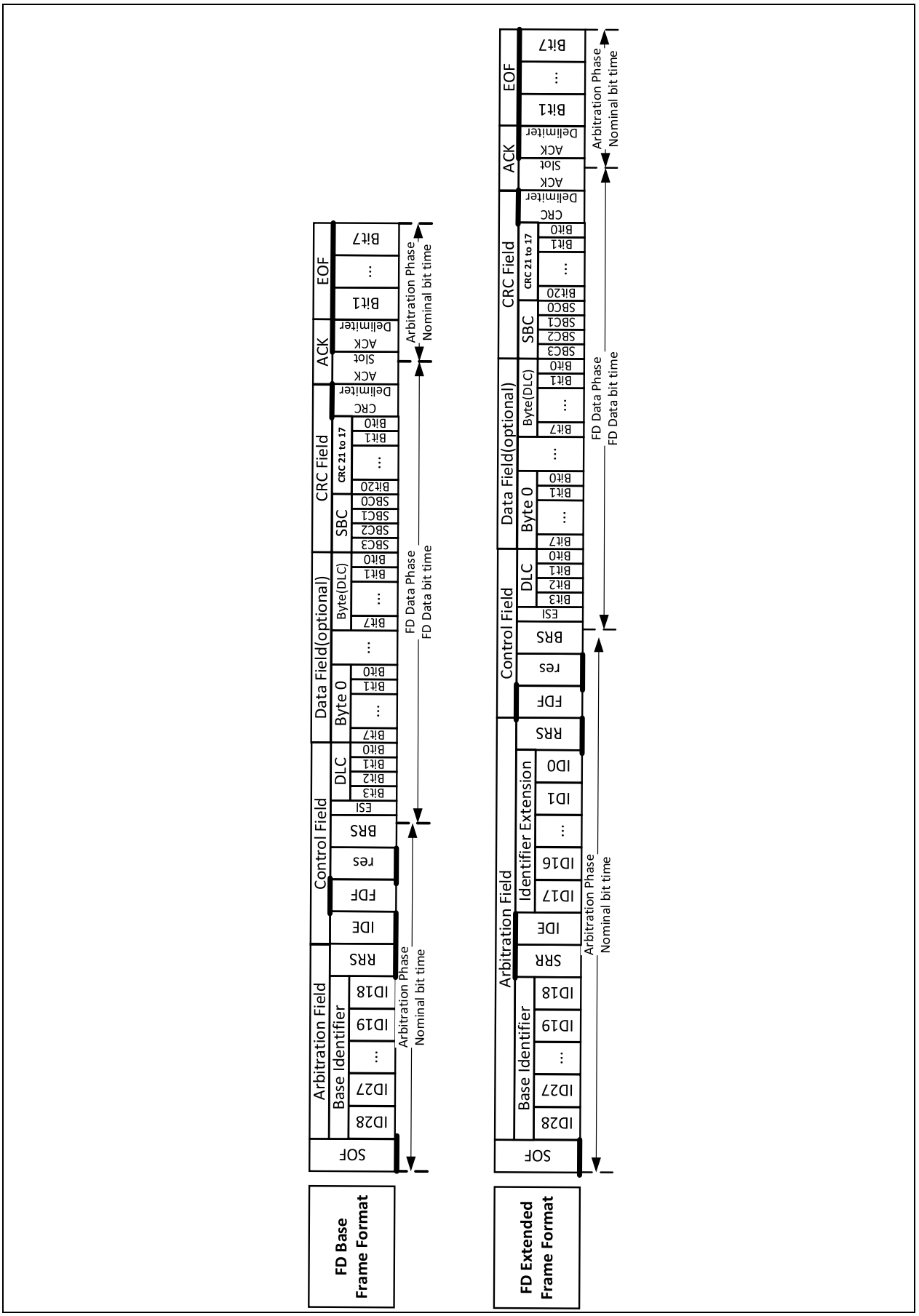
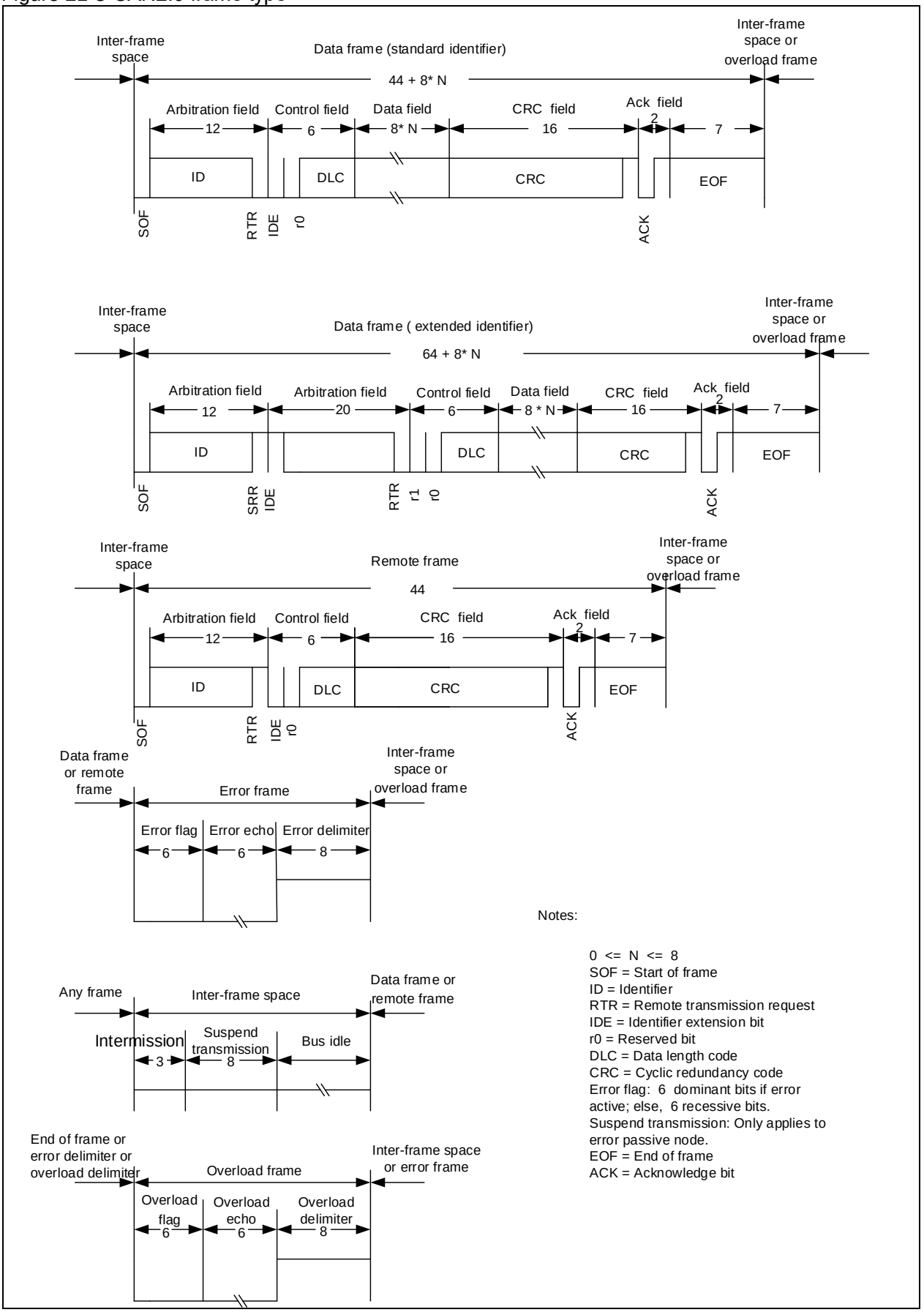

FD Base Frame Format
SOF
Arbitration Field
Base Identifier
ID28
ID27
...
ID19
ID18
RRS
IDE
FDF
res
BRS
Control Field
ESI
DLC
Bit3
Bit2
Bit1
Bit0
Data Field(optional)
Byte 0
Byte(DLC)
CRC Field
SBC
CRC 21 to 17
CRC
Delimiter
ACK
ACK Slot
ACK Delimiter
EOF
Arbitration Phase
Nominal bit time
FD Data Phase
FD Data bit time
FD Extended Frame Format
SOF
Arbitration Field
Base Identifier
ID28
ID27
...
ID19
ID18
SRR
IDE
Identifier Extension
ID17
ID16
...
ID1
ID0
RRS
FDF
res
BRS
Control Field
ESI
DLC
Bit3
Bit2
Bit1
Bit0
Data Field(optional)
Byte 0
Byte(DLC)
CRC Field
SBC
CRC 21 to 17
CRC
Delimiter
ACK
ACK Slot
ACK Delimiter
EOF
Arbitration Phase
Nominal bit time
FD Data Phase
FD Data bit time

Figure 21-3 CAN2.0 frame type



21.1.4 Upward compatibility and protocol exception event

The CAN specification includes reserved bits for protocol extension. This has been used to build the CAN FD specification on top of the CAN 2.0B specification.

Reserved bits are transmitted low (dominant) if not used. Unfortunately the CAN 2.0B specification defines the behavior for the case that a reserved bit is high (recessive) to accept this and proceed with the frame. Therefore if a CAN FD frame (which has a different and unexpected form compared to a CAN 2.0B frame) is received by a CAN 2.0B node that uses this behavior then this will result in an error frame by the CAN 2.0B node which destroys the frame. This behavior is called “CAN FD intolerant”.

To be upward compatible to new protocol specifications a so-called protocol exception event shall take place if a node detects a reserved bit high. This holds for CAN 2.0B as well as for CAN FD nodes. A protocol exception event results in no action for a receiver. The receiver just ignores this frame, does not generate an ACK, waits for bus idle and then may transmit or receive the next frame. For a CAN 2.0B node this is called “CAN FD tolerant” and enables coexistence of CAN 2.0B and FD frames within one network.

Older CAN 2.0B conformance tests check for the “CAN FD intolerant” behavior but it is recommended to use the new “CAN FD tolerant” behavior or in general the protocol exception event to be upward compatible to any new version of the CAN protocol.
Currently, we support new “CAN FD intolerant” behavior or protocol exception event only.

21.1.5 Time-triggered CAN

Optionally CAN controller can be used for time-triggered CAN communication (TTCAN) according to ISO11898-4. CAN controller offers partial hardware support and requires host software interactions in real time in this mode.

The basic concept of TTCAN is to have a timer for time-stamping of received frames and for triggering frames for transmission. One node in the CAN network is the time master. A time master transmit a reference message. With the reference message the cycle time starts. The time between two reference messages is a basic cycle. Inside the basic cycle messages can be transmitted in time windows. The TTCAN system administrator defines the start and duration of each time windown during an offline setup.

There are three time window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

If a frame is received, it gets the actual cycle time as time-stamp. For transmissions CAN controller offers a hardware trigger that starts transmission of a predefined frame at a predefined cycle time.

CAN controller automatically detects a reference message upon reception and starts the cycle time. The hardware time is a 16 bit time running at the CAN bit time as defined for ISO11898-4 Level1. CAN controller can be used as a time master.

Beside a hardware trigger for frame transmissions, CAN controller offers a watch trigger to detect a missing reference message.

Partial hardware support means that the host controller needs to prepare the actions of the node for each time window. E.g, the host needs to define the frame for the next transmission and define the trigger time for it.

21.1.6 CiA 603 time-stamping

CAN in automation (CiA) defines in specification CiA 603 a method for time-stamping with at least 16 bits, which is optionally supported by CAN controller. The CiA 603 time-stamping is independent from TTCAN. The basic concept of CiA 603 is to have a free0running timer which counts clock cycles and not CAN bit times. The precision shall be at least 10us (16 bit) or 1us (32 bit or more). Time-stamps can be acquired at the SOF or EOF of a CAN/CAN FD frame. CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR,one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master. CiA 603 defines rules to read out the time as well as to modify the timer. CAN controller does not include the timer, but uses an external timer.

the CAN controller only includes the mechanism of time-stamping, the register to store one transmission time stamp (TTS) and the memory to store reception time stamps for all received messages.

21.2 CAN features

21.2.1 Feature list

- Supports CAN specification
  - CAN2.0B: up to 8 bytes payload
  - CANFD: up to 64 bytes payload, ISO 11898-1:2015 or non-ISO Bosch
- Free programmable data rate
  - CAN2.0B defines data rates up to 1Mbit/s
  - CANFD is limited by the transceiver and the clock frequency of the CAN

    Note: CAN protocol states that the maximum allowed tolerance of oscillator for timer is at 1%. Therefore, in order to prevent the occurance of communication failure, an external crystal oscillator must be used as the clock source of CAN.
- Programmable baud rate prescaler (1 to 1/32)
- Receiver buffer (RB) 6-level depth
  - FIFO-like behavior
  - Received frames which are “not accepted” or “incorrect” don’t overwrite already stored frames
- Two transmit buffers
  - Primary transmit buffer (PTB) (one frame slot)
  - Optional 3-level secondary transmit buffer (STB), operation in FIFO or priority decision mode
- Independent and programmable internal acceptance filters
  - 16 acceptance filters
- Extended features
  - Limitation of re-arbitration and retransmission (1 to 7 or “unlimited”attempts)
  - Listen only mode
  - Loop back mode (internal and external)
  - Transceiver standby mode
- Extended status and error report
  - Status report for frames which are commanded to be transmitted
  - Capturing of last occurred kind of error and of arbitration lost position
  - Programmable error warning limit
- Configurable interrupt sources
- Time-stamping
  - ISO 11898-4 time-triggered CAN with partial hardware support
  - CiA 603 time-stamping
- Compatible to AUTOSAR
- Pptimized for SAE J1939

21.2.2 Interrupts

The CAN controller has four interrupt vectors that can be used to enable or disable interrupts by setting the CAN_INTEN register.

Figure 21-4 CAN1 transmit interrupt generation

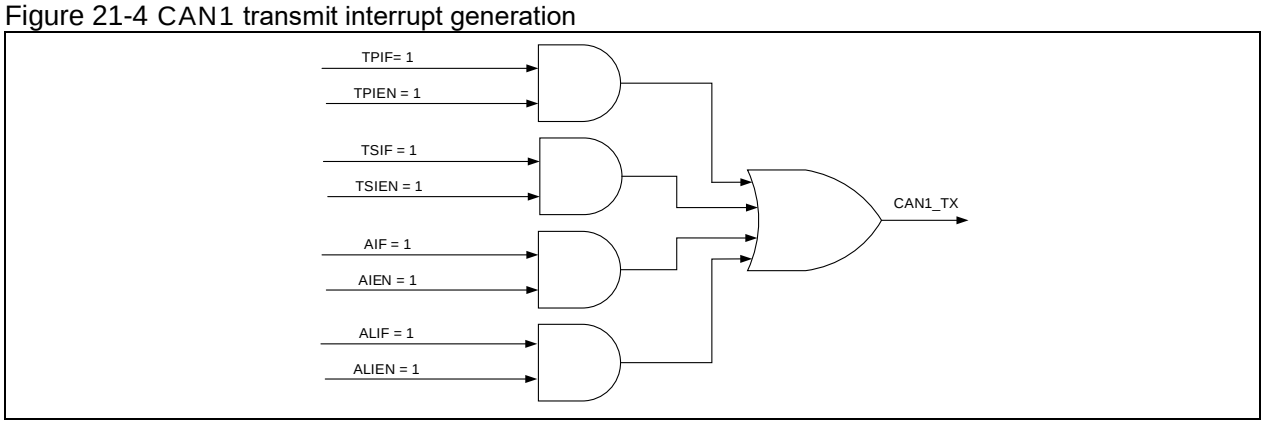


Figure 21-5 CAN1 receive interrupt generation

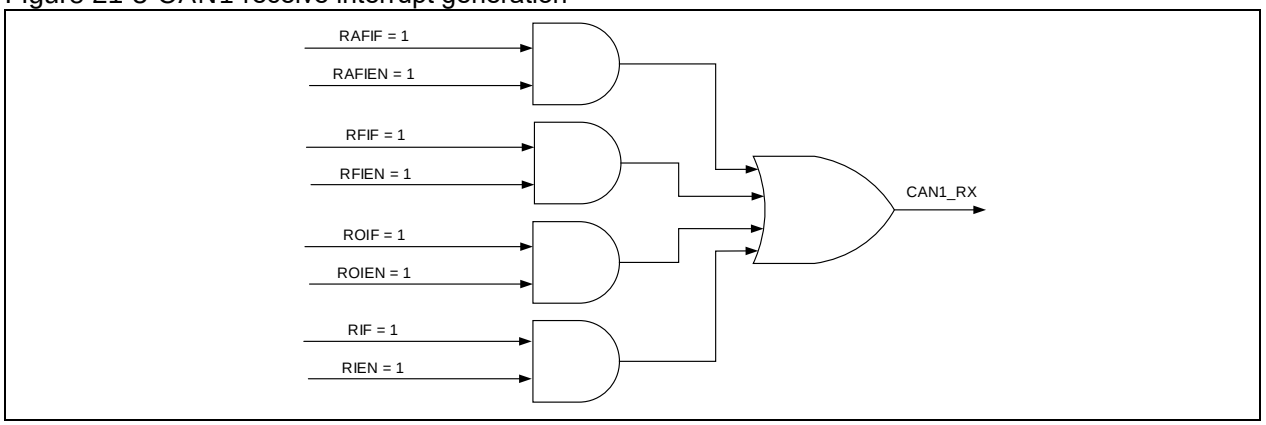


Figure 21-6 CAN1 status interrupt generation

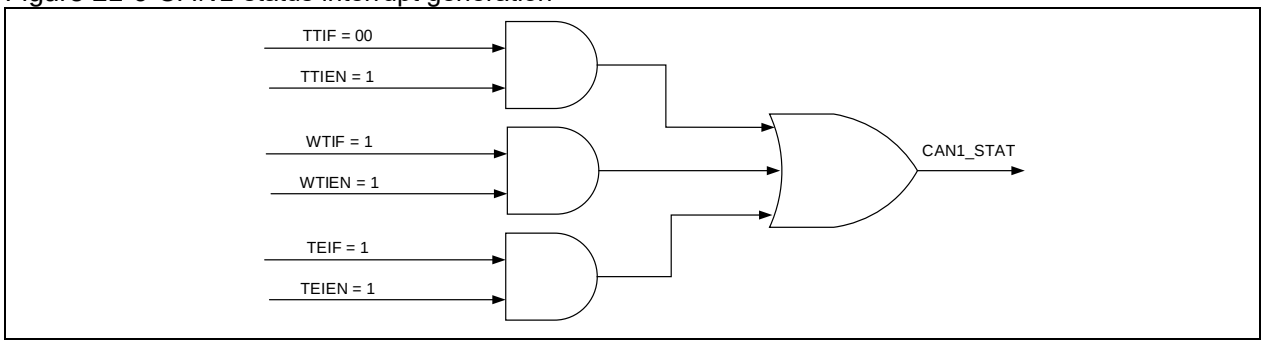
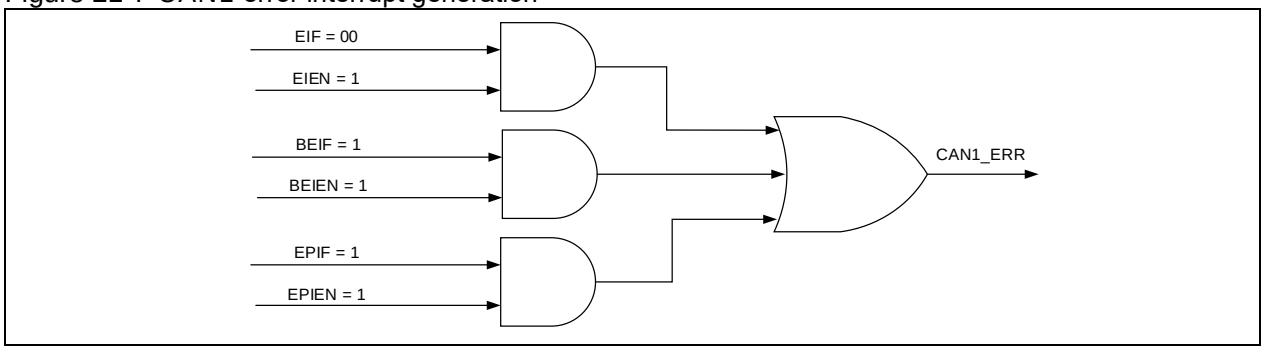


Figure 21-7 CAN1 error interrupt generation



21.2.3 Software interface

Table 21-1 shows the definition of the logical link control frame (LLC frame) plus time-stamps. LLC frames are exchanged between the host controller and CAN controller. This unified definition is used for frames to be transmitted (to be stored in TBUF), for received frames (to be read from RBUF) and to configure the acceptance filters (ACFC and ACFM).

For acceptance filtering, only the frame header consisting of identifier, format, type and acceptance field is relevant and used for ACFC and ACFM. Payload data as well as CiA603 and TTCAN time-stamps are not available for ACF.

Reception time-stamps (CiA 603 and TTCAN) are only avvilable in RBUF and not available for TBUF.

The reception time-stamps (CiA 603 and TTCAN) are located at a dynamic address offset position T behnnid the payload data. this dynamic position T is always word-aligned. To ease the handling of LLC frames by the application, the register LLCSIZE can be used.

Table 21-1 LLC frame (logical link control frame) definition (including time-stamps)

Address	Bit position								Function
offset	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	ID(7:0)								Identifier (ID)
1	ID(15:8)								
2	ID(23:16)								
3	TTSEN	-		ID(28:24)					
4					DLC(3:0)				Format (FMT)
5	-								
6	-			RMF	-	BRS	FDF	IDE	
7	-			LBF	ESI	KOER(2:0)			
8	-								Type (TYP)
9	-								
10	-								
11	HANDLE(7:0)								
12	-								Reserved
13	-								
14	-								
15	-								
16	D1(7:0)								Payload data
17	D2(7:0)								
…	…								
79	D64(7:0)								
T+0	RTS(7:0)								CiA 603 (reception)
…	…								
T+7	RTS(63:56)								
T+8	CYCLE_TIME(7:0)								TTCAN (reception)
T+9	CYCLE_TIME(15:8)								
T+10	-								
T+11	-								

Table 21-2 LLC frame abbreviations

Bit	Description		
ID	Frame identifier		
	Reception: yes	Transmission: yes	Acceptance filtering: yes
TTSEN	Transmit Time-Stamp enable For CiA 603 time-stamping, the acquuision of a transmit time stamp TTS can be selected: 0: no acquuision of a transmit time stampl for this frame 1: TTS update enabled		
	Reception: no	Transmission: yes	Acceptance filtering: no
DLC	Data length code, the DLC defines the number of payload bytes inside the frame		
	Reception:: yes	Transmission: yes	Acceptance filtering: yes
IDE	Identifier extension 0: Standard format: ID(28:18) 1: Extended format: ID(28:0)		
	Reception: yes	Transmission: yes	Acceptance filtering: yes
FDF	CANFD frame format FDF=0: CAN2.0 frame (up to 8 bytes payload) FDF=1: CANFD frame (up to 64 bytes payload)		
	Reception: yes	Transmission: yes	Acceptance filtering: yes
BRS	CANFD bit rate switch enable 0: nominal/slow bit rate for the complete frame 1: switch to data/fast bit rate for the data payload and the CRC of CANFD frames		
	Reception: yes	Transmission: yes	Acceptance filtering: yes
RMF	Remote frame 0: data frame 1: remote frame		
	Reception: yes	Transmission: yes	Acceptance filtering: yes
KOER	Kind of error KOER for received frames becomes meaningful if RBALL=1. If RBALL=1, then acceptance filtering is disabled in general.		
	Reception: yes	Transmission: no	Acceptance filtering: no
ESI	Error state indicator The protocol machine automatically embeds the correct value of ESI into transmitted frames. 0: CAN node is error active 1: CAN node is error passive ESI is only included in CANFD frames and therefore always 0 for CAN2.0B.		
	Reception: yes	Transmission: no	Acceptance filtering: yes
LBF	Loop-back frame LBF for received frames is set to 1 if the loop back mode is activated and CAN controller has received its own transmitted frame. This can be useful if LBME=1 while other nodes in the network also do transmissions.		
	Reception: yes	Transmission: no	Acceptance filtering: yes

HANDLE	Handle for frame identification The purpose of the handle is frame identification using TSTAT. It is suggested that the host application writes the value of a software counter to HANDLE.		
	Reception: no	Transmission: yes	Acceptance filtering: no
Payload Data	Payload data of the frame. Up to 8 bytes for Classic CAN2.0B, up to 64 bytes for CANFD. See Table 21-3 for details. Unused payload data words in RBUF do contain garbage data which needs to be ignored.		
	Reception: yes	Transmission: yes	Acceptance filtering: N/A
RTS	Receptio time stamp for CiA 603 time-stamping RTS is stored for each received frame. Therefore, in contrast to TTS, RTS is related to one specific received frame. If TSEN=0, then RTS=0.		
	Reception: yes	Transmission: N/A	Acceptance filtering: N/A
CYCLE_TIME	Cycle time (TTCAN time-stamp) CYCLE_TIME will be stored only for received frames. This is the cycle time at the SOF of the frame. The cycle time of a reference message is always 0.		
	Reception: yes	Transmission: N/A	Acceptance filtering: N/A

Table 21-3 Definition of the DLC

DLC (binary)	Frame type	Payload in bytes
DLC(3:0)=0000 to 1000	Classic CAN2.0B and CANFD	0 to 8
DLC(3:0)=1001 to 1111	Classic CAN2.0B	8
DLC(3:0)=1001	CANFD	12
DLC(3:0)=1010	CANFD	16
DLC(3:0)=1011	CANFD	20
DLC(3:0)=1100	CANFD	24
DLC(3:0)=1101	CANFD	32
DLC(3:0)=1110	CANFD	48
DLC(3:0)=1111	CANFD	64

21.3 Operating instruction

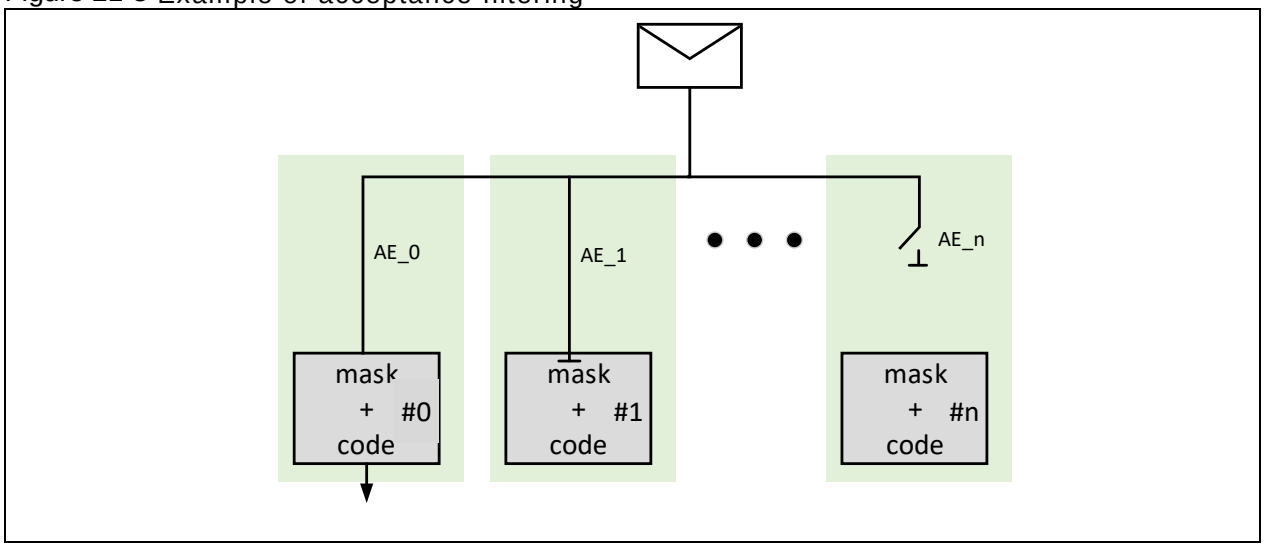
21.3.1 Acceptance filters

To reduce the load of received frames for the host controller, the CAN controller uses acceptance filters. CAN controller checks the frame header during acceptance filtering.

If a frame passes one of the filters then it will be accepted. If accepted, the frame will be stored into the RB and finally RIF is set if RIE is enabled. if the frame is not accepted, RIF is not set and the RB FIFO pointer is not increased. Frames that are not accepted will be discarded and overwritten by the next frame. No stored valid and accepted frame will be overwritten by any not accepted frame.

Acceptance filtering is only done for valid frames. (In case of errors during transmission CAN controller will do error handling according to the Can specification).

Figure 21-8 Example of acceptance filtering



The acceptance mask defines which bits shall be compared while the acceptance code defines the appropriate values. Setting an acceptance mask bit to 0 enables the comparison of the selected acceptance code bit with the corresponding bit of the received frame. Mask bits that are set to 1 are disabled for the acceptance check and this results in accepting the frame. (To “mask” a bit means to “don’t care”.)

Example: ID(28) is the most significant bit of the identifier of all CAN frames and transmitted first. Let’s assume ID(28) of ACFM is 0 and all other bits of ACFM are 1. To accept a received frame, the value of ID(28) of the received frames has to be equal to ID(28) of ACFC. All other bits are ignored for acceptance filtering.

Figure 21-8 gives an example of acceptance filtering using several filters. In this example the filter 0 and 1 are enabled by the AE_0 and AE_1 bits in the ACFCTRL register. All other filters are disabled and therefore do not accept any frame. For the two filters that are enabled, the combination of mask and code defines if a frame is accepted or not. In the example of Figure 22-8, filter 0 accepts the frame while filter 1 does not accept it.

Note: Disabling a filter by setting AE_x=0 blocks frames. In contrast to this, a mask bit in ACFM disables the check for this bit which results in accepting frames.

After power-on reset, CAN controller is configured to accept all frames (filter 0 is enabled by AE_0=1 and all bits in ACFM are read as 1. All other filters are disabled. Filter 0 is the only filter that has defined reset values for ACFC/ACFM while all other filters have undefined reset values).

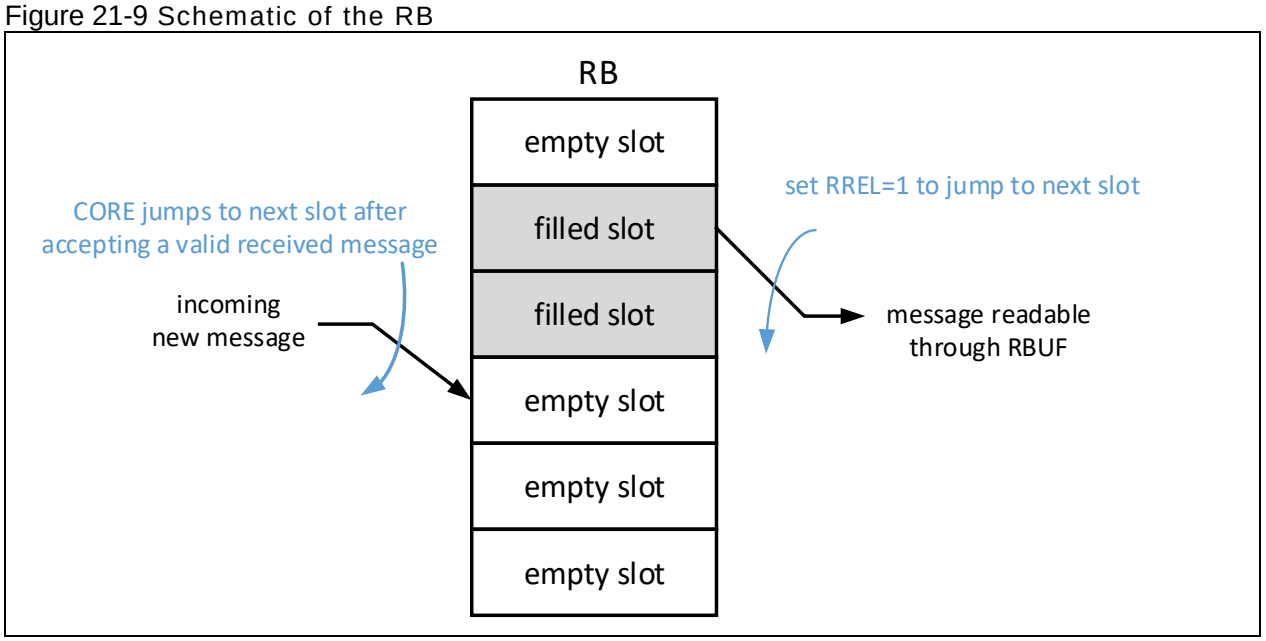
Additional notes:

- *Using proper settings for the mask bit is possible to accept groups of frames using only one single acceptance filter. Up to 16 acceptance filters can be included in the CAN, which means, that up to 16 filter groups are possible to be defined.*
- *The structure of acceptance filters is done based on the definition of an LLC frame to provide the best flexibility for acceptance filter group definitions. This for example makes it possible to accept all types of CAN frames with e.g. the most significant bit of the BaseID/PriorityID (which is ID(28)) set to a certain value.*
- *Some bits are not available for all types of CAN frames (CAN2.0B, CANFD) or are forced to a fixed value. If for example for bit IDE the mast is set to 0 and the code is set to 1, then this means to accept only frames where bit IDE=1.*
- *The host application is responsible to do proper filter settings. For example it does not make sense to unmask bits of the LLC frame. The CAN controller will not protect against such a misconfiguration. Therefore it is necessary to mask all bits (including the unused bits) except those of interest.*

21.3.2 Frame reception

The received data will be stored in the RB as shown in Figure 22-9. The RB has FIFO-like behavior. Every received frame that is valid and accepted sets RIF=1 if RIE is enabled. RSTAT is set depending on the fill state. When the number of filled buffers is equal to the programmable value AFWL, then RAFIF is set if RAFIE is enabled. in case, when all buffers are full, the RFIF is set if RFIE is enabled.

Figure 21-9 Schematic of the RB



The RB always maps the frame slot containing the oldest frame to the RBUF registers. The maximum payload length for Classic CAN2.0B is 8 bytes, for CAN FD frames 64 bytes. The individual length of each frame is defined by the DLC. The RB provides slots big enough to carry frames of maximum size. The host controller is required to set RREL to jump to the next RB slot. All RBUF bytes of the actual slot can be read in any order.

If the RB is full, the next incoming frame will be stored temporarily until it passes for valid (6^{th} EOF bit). Then if ROM=0, the oldest frame will be overwritten by the newest or if ROM=1, the newest frame will be discarded. In both cases ROIF is set if ROIE is enabled. If the host controller reads the oldest frame and sets RREL before a new incoming frame becomes valid then no frame will be lost.

21.3.3 Handling frame receptions

Without acceptance filtering,CAN controller would signal the reception of every frame and the ost would be required to decide if it was addressed. This would result in quite a big load on the host controller.

Beside using acceptance filters to reduce the load it is also possible to disable interrupts. For a basic operation RIF is set to 1 if RIE is enabled and the CAN has received a valid frame. To reduce the number of reception interrupts it is possible to use RAIE/RAFIF(RB almost full interrupt) or RFIE/RFIF (RB full interrupt) instead of RIE/RIF(reception interrupt). The “almost full limit” is programmable using AFWL.

The RB contains six RB slots. Reading the RB shall be done as follows:

- Read the oldest frame from the RB FIFO using the RBUF registers
- Release the RB slot with RREL=1. This selects the next frame (the next FIFO slot). RBUF will be updated automatically.
- Repeat these actions until RSTAT signals an empty RB.

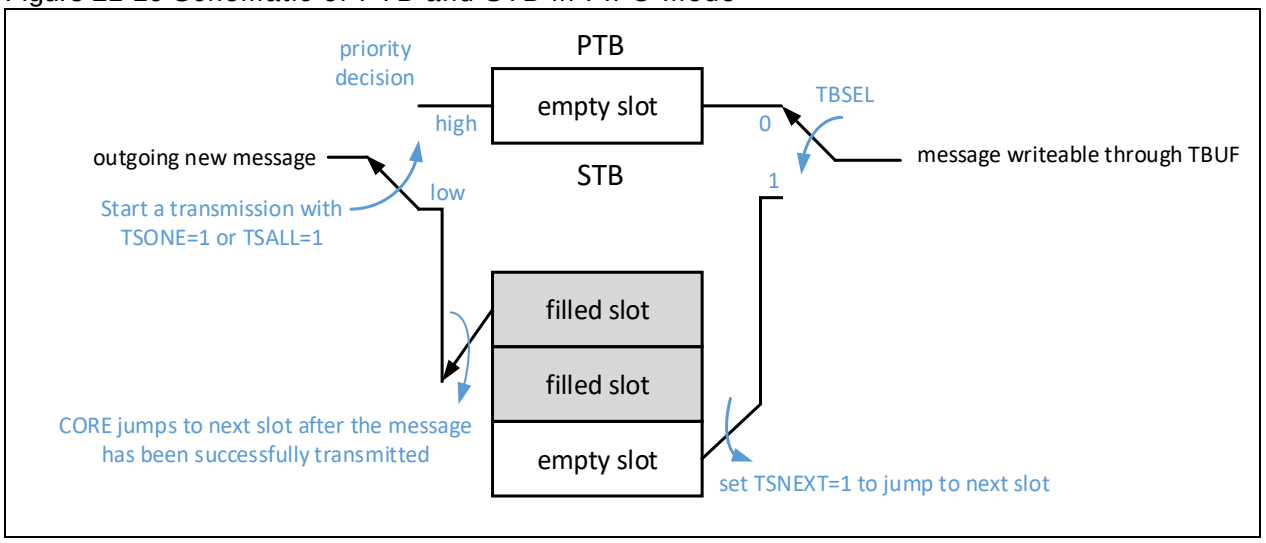
If the RB FIFO is full and a new received frame is recognized as valid (6th EOF bit) then one frame will be lost (see bit ROM). Before this event, no frame is lost. This should give enough time for the host controller to read at least one frame from the RB after the RB FIFO has been filled and the selected interrupt has occurred.

21.3.4 Frame transmission

Before starting any transmission, at least one of the transmit buffers (PTB or STB) has to be loaded with a frame (see Fifgure below). TSSTAT signals the fill state of the STB. (If TPE=1, then the PTB is write-locked). The TBUF registers provided access to both the PTB as well as to the STB. Below is the recommended programming flow:

- Set TBSEL to the desired value to select either PTB or STB
- Write the frame to the TBUF registers (All TBUF bytes can be written in any order)
- For STB, set TSNEXT=1 to finish loading of this STB slot
- For PTB, setting TSNEXT=1 has no imact and TSNEXT will automatically be cleared.

Figure 21-10 Schematic of PTB and STB in FIFO mode



The maximum payload length for CAN 2.0B frames is 8 bytes, and for CAN FD frames 64 bytes. The individual length of each frame is defined by the DLC. For CAN 2.0B remote frames (bit RMF), the DLC becomes meaningless, because classic CAN remote frames always have a data length of 0 bytes. Bit tpe should be set to start a transmission when using the PTB. To use the STB, TSONE has to be set to start a transmission of a single frame or TSALL to transmit all frames.

The PTB has always a higher priority than the STB. If both transmit buffers got the older to transmit, the PTB frame will be always sent first regardless of the frame identifiers. If a transmission from the STB is already active, it will be complted before the frame from the PTB is sent at the next possible transmit position (the next interframe slow). After the PTB transmission is completed or aborted, CAN returns to process other pending frames from the STB.

When the transmission is completed, the following transmission interrupts are set:

- For PTB, TPIF is set if TPIE is enabled
- For STB using TSONE, TSIF is set if one frame has been completed and TSIE is enabled
- For STB using TSALL, TSIF is set if all frames have been completed and if TSIE is enabled. In other words, TSIF is set if the STB becomes empty. Therefore, if the host controller writes an additional frame to the STB after a STALL transmission has been started then the addiontal frame will be also transmitted before TSIF will be set.

21.3.5 Frame transmission abort

If the situation arises, where a frame in a transmit buffer cannot be sent due to its low priority, this would block the buffer for a long time. In order to avoid this, the host controller can withdraw the transmission request by setting TPA or TSA respectivey, if the transmission has not yet been started.

Both TPA and TSA source a single interrupt flag: AIF. The CAN protocol machine executes an abort only if it does not transmit anything to the CAN bus (no active transmission will be interrupted). Therefore the following rules are valid:

- There is no abort duing bus arbitration
- If the node loses arbitration, the abort will be executed afterwards
- If the node wins arbitration, the frame will be transmitted
- There is no abort while a frame is transmitted
- If a frame is transmitted successfully then a successful transmission is signaled to the host controller.

In this case no abort is signaled. This is done by the appropriate interrupt and status bits
- After an unsuccessful transmission after any kind of error, the error counter is incremented and the abort will be executed
- If there is at least one frame left in the STB, while the host has commanded all frames to be transmitted (TSALL=1), then both the completed frame as well as the abort is signaled to the host

Beecause of these facts aborting a transmission may take some time depending on the CAN communication speed and frame length. If an abort is executed, this results in the following actions:
- TPA releases the PTB which results in TPE=0. The frame data is still stored in the PTB after releasing the PTB
- TSA releases one single frame slot or all frame slots of the STB. This depends on whether TSONE or TSALL was used to start the transmission. TSSTAT will be updated accordingly. Releasing a frame in the STB results in discarding the frame data because the host cannot access it.

Setting both TPA and TSA simultaneously is not recommended. If a host controller decides to do it anyway, then AIF will be set and both transmissions from PTB and STB will be aborted if possible. As already stated if one transmission will be completed before the abort can be executed this will result in signaling a successful transmission. Therefore, the following interrupt flas may be set if enabled:
- AIF (once for both PTB and STB transmission abort)
- TPIF+AIF
- TSIF+AIF
- TPIF+TSIF (very seldom, will only happen if the host does not handle TPIF immediately)
- TPIF+TSIF+AIF (very seldom, will only happen if the host does not handle TPIF and TSIF)

To clear the entire STB, both TSALL and TSA need to be set. In order to detect if a frame cannot be sent for a long time because it loses arbitration the host may use the ALIF/ALIE.

21.3.6 A full STB

After writing a frame to the STB, TSNEXT=1 marks a buffer slot filled and jumps to the next free frame slot. TSNEXT is automatically reset to 0 by CAN after this operation.

If the last frame slot has been filled and therefore all frame slots are occupied then TSNEXT stays set until a new frame slot becomes free. While TSNEXT=1, then writing to TBUF is blocked by CAN.

When a slot becomes free, then CAN controller automatically resets TSNEXT to 0. A slot becomes free if a frame from the STB is transmitted successfully or if the host requests an abort (TSA=1) or if the limits defined by RETLIM or REALIM have been reached. If a TSALL transmission is aborted, then TSNEXT is also reset, but additionally the complete STB is marked as empty.

21.3.7 Error handling

On one hand CAN controller does automatic error handling which means that in most cases the host controller does not need to care about errors. This includes automatic frame retransmission and automatic deletion of received frames with errors. On the other hand if required by the host, the CAN controller may ooptionally give detailed information about errors and signal every error to the host by interrupt. This enables to run an application like a CAN bus monitor at the host.

Every CAN node has 3 states of error handling:
- Error active: the node automatically transmits active error frames upon detection of an error
- Error passive: the node transmits a passive error frame upon detection of an error. This means that it does not transmit a dominant value to the bus, but expects an error frame transmitted by other can nodes.

  Error-passive nodes are alloed to transmit frames, but if they have been the transmitter of the previous frame they are required to suspend a further frame transmission for 8bits following the intermission.
- Bus off: after too many errors a node goes “bus off” where it stops touching the bus.

To handle the 3 error states every CAN node has two erro counters: the transmit and the receive error counter (readable via TECNT and RECNT). Both are incremented and decremented as defined by the CAN specification and the node enters the appropriate error state upon reaching a counter level defined by the CAN specification. See also Table 21-4 for more details.

The error counters are incremented if there are errors. Dangerous errors that are probably caused by this node will result in incrementing the counter by 8, errors that are probably caused by other nodes will result in incrementing the counter by 1. Valid frame transmission or valid receptions result in decrementing the counters. All this is defined by the CAN specification and handled automatically by the

CAN controller.

An error frame (bing the result of erro flags) is different to a data frame. It is a dominant pulse for at least 6 consecutive dominant bits, which is a stuff bit violation for other nodes. If a CAN node detects this violation, then it also transmits an error flag resulting in an error frame which is the superposition of erro flags from all nodes. (As a result, the duration of error frames is 6 to 12 bit times). A host controller cannot command to transmit an error frame. It is done fully automatically by the CAN controller.

If the CAN controller is commanded to transmit a frame then it automatically tries retransmission as fast as possible until the frame gets transmitted without error or the node goes bus off. If a frame is received and the CAN controller detects an error, then the received daa is discarded. Because of the automatically transmitted error frame, the sneder of the frame will retransmit the frame. Frames in RBUF are nevre overwritten by frames with errors. Only valid received frames may result in RBUF overflow.

Retransmission is limited by TECNT (if the node goes bus off), but can be further limited using RETLIM and REALIM.

Table 21-4 States of a CAN node

State	TECNT	RECNT	EPASS	BUSOFF
Error active	Both smaller than 128		0	0
Error passive	One of them not smaller than 128		1	0
Bus off	Bigger than 255	-	1	1

21.3.8 Bus off state

The “bus off” state is signaled using the status bit BUSOFF in the CTRLSTAT register. A CAN node enters the “bus off” state automatically if its transmit error counter becomes bigger than 255. Then it will not take part in further communications until it returns into the error-active state again. Setting BUSOFF to 1 also sets the EIF interrupt if EIE is enabled. A CAN node returns to error active state it it is reset by a pwer-on reset or if it receive 128 sequences of 11 recessive bits (recovery sequences). Please note that between each recovery sequence the bus may have dominant state. (note that the minimum time between two valid frames is sufficient to be recognizied as recovery requence)

In the “bus off” state, RECNT and TECNT stay unchanged. Please note that while entering bus off state, TECNT rolls over and therefore may hold a small value. Therefore it is recommended to use TECNT before the node enters bus off state and status bit BUSOFF afterwards.

If a frame is pending for transmission but the CAN node has entered bus off state, then this frame is kept pending. If a node reutrnes to erro-active state after bus off, then the transmission of the pending frame will be retried. If this is not desired, then the frame should be aborted by the host controller.

21.3.9 Extended status and error reprot

During CAN bus communication errors may occur. The following features support detection and analysis of them. This can be used for extended bus monitoring.

21.3.9.1 Programmable error warning limit

Errors during reception/transmission are counted by RECNT and TECNT. A programmable error warning limit EWL, located in the ERR register, can be used by the host controller for flexible reaction on those events. The limit values can be chosen in stpes of 8 errors from 8 to 128:

Limit of count of errors = (EWL+1)*8

The interrupt EIF will be set if enabled by EIE under the following conditions:

- The border of the error warning limit has been crossed in either direction by RECNT or TECNT
- BUSOFF bit is set to 1 or cleared by hardware

21.3.9.2 Arbitration lost capture (ALC)

The loss of arbitration is not an error and therefore not relevant for TECNT/RECNT but also unwanted.

The CAN controller is able to detec the exact bit position in the Arbitration Field where the arbitration has been lost. This event can be signaled by the ALIF interrupt if it is enabled by ALIE=1. The value of ALC stays unchanged until another event of arbitration lost occurs.

The value of ALC is defined as follows: a frame starts with the SOF bit and then the first bit of the ID is

transmitted. This first ID bit (which is ID(28)) has ALC value 0, the second ID bit ALC value 1 and so on. See Figure 21-2 and Figure 21-3 for the bit order in all types of CAN frames.

Notes:

- *Arbitration is only allowed in the arbitration field. Therefore the maximum value of ALC is 31 which is the RTR/RRS bit in extended frames.*
- *If a standard remote frame arbitrates with an extended frame, then the extended frame loses arbitration at the IDE bit and ALC will be 12. The node transmitting the standard remote frame will not notice that an arbitration has taken place because this node has won.*
- *If is impossible to lose arbitration outside of the arbitration field. Such an event would be a bit error.*

21.3.9.3 Kind of error (KOER)

CAN controller recognizes errors on the CAN bus and stores the last error event in the KOER bit. A CAN bus error can be signaled by the BEIF interrupt if it is enabled with BEIE=1. Every new error event overwrites the previous stored value of KOER. Therefore the hhost controller has to react quickly on the error event.

KOER is updated with each new error. Therefore it stays untouched when frames are successfully transmitted or received. This opens the opportunity for a delayed error investigation. If RBALL=1 then also frames with error will be stored inside RB. The LLC frame also includes the KOER information.

If ROP=1 then an error will result in a protocol exception. Such an erro will aso set BEIF and KOER will also be updated. In contrast ot this protocol exception for upward compatibility will not result in setting BEIF or updating KOER.

21.3.9.4 Reception of all data frames (RBALL)

If RBALL=1 then all received data frames are stored inside RB even those with error. This holds also fro loop back mode. Also acceptance filtering is disabled. Only data frames are stored in RB. Error frames or overload frames are not stored.

If CiA 603 time-stamping is enabled (TSEN=1) and the timestamp is configured for the EOF (TSPOS=1) then in the case of an error the time-stamp is acquired at the start of the error frame.

Most of the possible errors cn only occur if thenode is the transmitter of the frame. In this case a frame would only be stored in RBUF if a loop back mode is activated. Depending on the type of error parts of the frame stored in an RBUF slot may be valid while other parts are unknown. Table 22-5 lists the possibilities.

Table 21-5 RBALL and KOER

KOER	Condition	Description
No error	All	Successful reception
Bit error	Receiver	Can only occur in the ACK slot. All stored data are valid.
	Transmitter	The payload is always invalid. The header bits including the ID may be valid. In the arbitration phase detection of a wroing bit is part of the arbitration and therefore not a bit error. Buf if a stuff bit in the arbitration phase is detected wrong by the transmitter of the frame, then tis is a bit effort and in this case the header bits are invalid. Therefore the header bits cannot be trusted, but if the hader matches to an expcted header of a frame, it can be used for further investigation.
Form error	All	Only form errors in a data frame are covered. This includes errors in the CRC delimiter, the ACK delimiter or the EOF bits. All stored data are valid.
Stuff error	Receiver	The position of a stuff error is unknown. All stored data are invalid.
	Transmitter	Can only happen during arbitration. All stored data are invalid.
ACK error	Receiver	Can only occur if the node is in LOM. All stored data are valid.
	Transmitter	Can only occur in loop back mode without self-ACK. All stored data are valid.
CRC error	Receiver	All stored data are valid.

21.3.9.5 Transmit status (TSTAT_1 and TSTAT_2)

There are various status and interrupt bits, which can be used to keep track of the status of frame transmissions, e.g. TPIF, ALIF, AIF, BEIF and some more. In addition to that a comprehensive information is also provided by the transmit status registers TSTAT_1 and TSTAT_2.

TSTAT_1 holds information about the frame which is currently in transmission. As a consequence the content of TSTAT_1 is highly volatile. Once a frame transmission is successfully completed or stopped because of reaching the re-arbitration imit or the retransmission limit (defined in the RETLIM register), then this information is stored in TSTAT_2. In other words, TSTAT_2 is updated when a frame reaches a final state.

A transmit status provides the following information:

- A handle to identify the frame
- The status of the transmission of the frame referenced by the handle
- The related timestamp of the frame referenced by the handle

The timestamp is provided with the TTS register and it is only updated if bit TTSEN=1 in the LLC frame (see Table 21-1). TTS is only related to the TSTAT_2 register.

Bits TSTAT inside TSTAT_1 and TSTAT_2 encode the status of a frame as defined in Table 21-6.

To identify a frame the HANDLE is used. This is an identifier chosen by the host application and included in the LLC frame (see Table 21-2). It is suggested that the value of a software counter is used as HANDLE which is incremented with every new frame to be transmitted. This counter may roll over.

Table 21-6 TSTAT status encoding

Value	Label	Description
000	IDLE	No transmission in progress
001	ONGOING	Transmission active without any issues
010	LOST_ARBITRATION	Arbitration lost. Re-arbitration may take place with respect to REALIM.
011	TRANSMITTED	Transmission successfully completed
100	ABORTED	Transmission aborted (TPA, TSA)
101	DISTURBED	Transmission error. Retransmission may take place with respect to RETLIM.
110	REJECTED	Misconfiguration of the frame format in the LLC frame.
111	-	Reserved

There are several possible events which may cause an update of TSTAT_1 or TSTAT_2 or both. Table 21-7 gives a comprehensive overview about these events.

Table 21-7 TSTAT status events

Event	TSTAT_1	TSTAT_2
Power-on reset or RESET	IDLE	IDLE
Start of a transmission (data fetch from TBUF by CAN protocol machine)	ONGOING	-unchanged-
Successful completion of a transmission	IDLE	TRANSMITTED
Transmission aborted (TPA or TSA)	IDLE	ABORTED
Arbitration lost, re-arbitration limit REALIM not reached	LOST_ARBITRATION	unchanged
Arbitration lost, re-arbitration limit REALIM reached	IDLE	LOST_ARBITRATION
Error, retransmission limit RETLIM not reached	DISTURBED	-unchanged-
Error, retransmission limit RETLIM reached	IDLE	DISTURBED
Bus-off recovery	ONGOING	-unchanged-
Misconfiguration in the LLC frame detected at start of transmission	IDLE	REJECTED

All events that cause an update of TSTAT_1 or TSTAT_2, except for the event of a transmission abort, will occur step by step and as a consequence the host application can keep track of it. In contrast to this a transmission abort may be commanded at any time. On top of that there are complex scenarios of aborts which are impossible to be signaled properly by TSTAT_1 and TSTAT_2.

- If the STB is filled with more than one frame, a TSALL transmission is chosen and TSA is commanded, then several frames will be aborted. TSTAT_2 can only reflect the status of one frame.
- If both PTB and STB are used, the PTB transmission is currently active but the STB transmission is aborted, then the events of successful trasmissio of the frame from PTB and the abort of the frames from STB may happen almost at the same time. This would require to change the content of TSTAT_2 twice in a row and the host application cannot keep track of this.
- If the STB is used, a TSALL transmission is chosen and TSA is set after one frame has been successfully completed and before the next one is started. This would require to change the content of TSTAT_2 twrice in a row and the host application cannot keep track of this.

As a consequence the status ABORTED will only be set for TSTAT_2 if all of the following conditions are true:

- TPA is set while a PTB transmission is active or TSA is set while a STB transmission is active
- The transmission currently being active has status ONGOING, LOST_ARBITRATION or DISTURBED signaled by TSTAT_1

21.3.9.6 Frame rejection

If a frame is commanded to be transmitted and the frame format of this frame is misconfigured, then the frame will be rejected and not be transmitted.

Frame rejection will result in setting TPIF or TSIF (depending on if the PTB or STB was used and if the appropriate interrupt enable bit was set) but will be signaled using the transmit status REJECTED as defined in Chapter 21.3.9.5.

The reason for a frame format misconfiguration is a serious error in an upper application layer and shall not happen during normal operation. It is expected to happen only during application developemtn because of mistakes.

21.3.10 Bit rate switching and transceiver mode switching

Classic CAN frames run at one constant bit rate (“slow” nominal bit rate). CAN FD frames may switch the bit rate inside the frame if BRS=1. They start with “slow” nominal bit rate and switch to “fast” data bit rate at the sample point of the BRS bit and back at the sample point of the CRC delimiter or if there is an error.

Reaching higher bit rates depend on the bus topology and electrical characteristics of the transceivers. For CAN FD there are transceivers with signal improvemtn capability (CANFD SIC transceivers), which enable to reach higher bit rates.

Classic CAN and CAN FD transceivers use a concept of pull resistors. Logic value 1 is the result of the pull resistors and therefore called “recessive” while logic value 0 is actively driven and therefore called “dominant”. Using pull resistors results in asymmetries of the bus and limits the bit rate.

21.3.11 Extended features

21.3.11.1 Retransmission and re-arbitration limitation

Sometimes automatic retransmission or re-arbitrations are not desired, because repeated attempts may result in large delays and also increased bus load. Both can be limited by RETLIM and REALIM. These limitation are fully independent from each other.

Retransmission will takep palce after an error while re-arbitration will take place after arbitration has been lost. Retransmission may be stopped if the bus-off state is reached.

The counters for retransmission and for re-arbitration attemps will be incremented if the appropriate event takes place and wil be both reset to 0 when starting a new transmission (TPE, TSONE or TSALL) or in ase several frames are transmitted using TSALL, then the counters are reset to 0 if one of these frames is either successfully transmitted or finally not successful (e.g. retransmission/re-arbitraion limit reached). It does not matter if the PTB or STB was chosen as the source of the transmission – only the evnets of error, arbitration lost or successful transmission have an impact on the two counters. The following explains this behavior in a step-by-step example:

- STB transmission fails because of an error → retransmission counter incremented
- STB transmission loses arbitration → retransmission counter incremented
- PTB transmission successful → both counters reset (frame in STB still not transmitted

In the case of an immediate successful transmission there is no difference to normal transmission. But in the case of an unsuccessful transmission the following will happen:

- TPIF gets set if enabled, the appropriate transmit buffer slot gets cleared
- In case of an error, KOER and the error counters get updated. BEIF gets set if BEIE is enabled and the other error interrupt flags will act accordingly.

Therefore if retransmission or re-arbitration is limited, then TPIF alone does not indicate if the frame has been successfully transmitted. Therefore retransmission or re-arbitration limitation should only be used together with BEIF and ALIF if a feedback about success of a transmission is required. Using TSTAT_1 and TSTAT_2 (see 21.3.9.5) may also hep to keep track of the events.

If retransmission or re-arbitration is limited, TSALL=1 is set and there is more than one frame in the STB then for each frame the attempts will be counted. If any of the frames is not successfully transmitted before one of the limits is rached, then CAN controller advances to the next frame and finally stops if the STB is empty. In this scenario it is rather complex to determine what has happended and only the error coutner TECNT indicates what has happended. This can be quite complex to evaluate because if one of two frames got errors the host cannot detect which one was the successful one.

Limiting retransmission and re-arbitration enables to operate in time-slots to guarantee maximum delays of messages.

21.3.11.2 Listen only mode (LOM)

LOM provides the ability of monitoring the CAN bus without any influence to the bus:

- LOM is similar, but not fully compatible to the bus monitoring feature defined in ISO 11898-1:2015. The difference is the ACK: in LOM CAN relies on a dominant ACK from another CAN node
- LOM is similar, but no identical to the restricted operation mode defined in ISO 11898-1:2015. Restricted operation is discussed in Chapter 21.3.11.3

Errors will be monitored (KOER and KOER) in LOM.

In LOM, CAN controller is not able to write dominant bits onto the bus (no active error flags or overload flags, no acknowledge). This is done using the following rules:

- If LOM=1, the protocol machine acts as if in restricted operation mode where every error leads to a protocol exception event
- If LOM=1,, the protocol machine does not generate a dominant ACK
- The error counters are not modified in any direction

Important facts regarding ACKs in LOM:

- If a frame is transmitted by a node then an ACK visible at the bus will be only generated if at least one additional node is attached to the bus and permitted to generated the ACK. Then there will be no error and all nodes will receive the frame. Otherwise an ACK error invalidates the frame
- If there is an ACK error, then the node in LOM is able to detect this

Note: Activation of LOM cannot be done while a transmission is active. If LOM is enabled then no transmission can be started.

The loop back mode (external) LBME plays an important role for the behavior of LOM. If LBME is disabled, then LOM acts as described above and the node cannot write any dominat bit to the bus. Buf if LBME is activated, then the node is allowed to transmit a frame including the optional self-ACK. LBME allows only the only the transmission of a frame including the optional self-ACK, but the node will not respond with an ACK to frames from other nodes and will not generate error or overload frames. In summary the combination of LOM and LBME is “a silent receiver, which is able to transmit if necessary”.

21.3.11.3 Restricted operation (ROP)

A CAN node which is in restricted operation mode shall not transmit data frames unless it is a potential time master. Then such a node shall be able to transmit time reference messages to start up the network.

A CAN node which is in restricted operation mode is able to receive frames and will respond with ACK to valid frames. In case of an error or overload condition the node with ROP=1 will treat this as protocol exception and will enter bus integration. The error counters are not modified in any direction if ROP=1

Note: Activation of ROP cannot be done while a transmission is active

21.3.11.4 Loop back mode (LBMI and LBME)

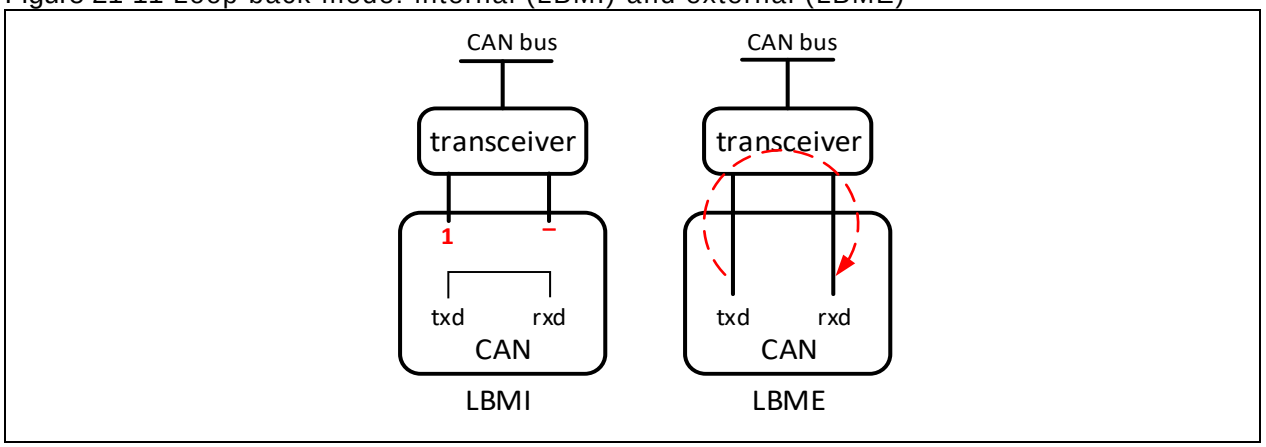
CAN-CTRL supports two Loop Back Modes: internal (LBMI) and external (LBME). Both modes result in reception of the own transmitted frame which can be useful for self-tests. See Figure 21-11 for details

In LBMI CAN-CTRL is disconnected from the CAN bus and the <txd> output is set to recessive. The output data signal is internally fed back to the input. In LBMI the node generates a self-ACK to avoid an ACK error

In LBME CAN-CTRL stays connected to the transceiver and a transmitted frame will be visible on the bus. With the help of the transceiver CAN-CTRL receives its own frame. In LBME the node does not generate a self-ACK when SACK=0, but generates a self-ACK when SACK=1. Therefore in LBME with SACK=0 there are two possible results upon a frame transmission:

- Another node receives the frame too and generates an ACK. This will result in a successful transmission and reception
- No other node is connected to the bus and this results in an ACK error. To avoid retransmissions and incrementing the error counters it is recommended to set RETLIM=0 if it is unknown if other nodes are connected.

Figure 21-11 Loop back mode: internal (LBMI) and external (LBME)



In Loop Back Mode CAN controller receives its own frame, stores it in the RBUF and sets the appropriate receive and transmit interrupt flags if enabled. Own received frames are flagged with LBF=1.

LBMI can be useful for chip-internal tests and software tests while LBME can test the transceiver and the connections to it

Note: LBMI and LBME shall not be changed together with TPE, TSONE or TSALL.

21.3.11.5 Transceiver standby mode

Using the register bit STBY the output signal <stby> is driven. It can be used to activate a standby mode for a transceiver. The behavior is compatible to the NXP TJA1049 transceiver and many more transceivers which provide a similar feature.

Once standby is activated no transmission is possible and therefore TPE, TSONE and TSALL cannot be set. On the other hand CAN-CTRL does not allow STBY to be set if a transmission is already active (TPE, TSONE or TSALL is set).

If STBY is set the transceiver enters a low-power mode. In this mode it is unable to receive a frame at full speed but monitors the CAN bus for a dominant state. If the dominant state is active for a time which is defined in the transceivers data sheet the transceiver will pull the <rxd> signal low. If <rxd> is low CANCTRL automatically clears STBY to 0 which disables standby mode for the transceiver. This is done silently without an interrupt to the host controller. (When standby mode is left, it is expected, that a frame from another node will be received.)

Switching from standby mode to active mode takes some time for the transceiver and therefore the initial wakeup frame cannot be received successfully. Therefore the node recently being in standby will not respond with an ACK. If no CAN node at the bus responds to the wakeup frame with an ACK then this results in an ACK error for the transmitter of the wakeup frame. Then the transmitter will automatically retransmit the frame. At the time of the retransmission the transceiver will be back in active mode and CAN-CTRL will receive the frame and will respond with an ACK.

21.3.11.6 Error counter reset

According to the CAN standard RECNT counts receive errors and TECNT counts transmit errors. After too many transmit errors a CAN node has to go to bus off state. Bit RESET does not modify the errors counters or the bus off state. The CAN specification defines rules how to exit from bus off state and to decrease the error counters. A good node will recover from all of this automatically if only a temporary error has caused the problems. The classic CAN 2.0B specification requires this automatic behavior without host controller interaction to avoid the babbling idiot problem.

The CAN FD specification relaxes this and allows the host controller to override the automatic error counter handling. But this should be used with extra care and is suggested to use only for debugging purposes.

Writing a 1 to the bit BUSOFF resets the error counters and therefore forces the node to leave the bus off state. This is done without setting EIF.

21.3.11.7 Software reset

If bit RESET in CFG_STAT is set to 1 then the software reset is active. Several components are forced to a reset state while RESET=1 but not all components are touched by RESET. Some components are only sensitive to a hardware reset. The reset values of all bits are always the same for software and hardware reset.

Table 21-8 Software reset

Register	Reset	Comment
ACFADR	No	-
ACFC	No	Register is writeable if RESET=1 and write-locked if 0.
ACFM	No	Register is writeable if RESET=1 and write-locked if 0.
AC_SEG_1	No	Register is writeable if RESET=1 and write-locked if 0.
AC_SEG_2	No	Register is writeable if RESET=1 and write-locked if 0.
AC_SJW	No	Register is writeable if RESET=1 and write-locked if 0.
AE_x	No	-
AFWL	No	-
AIF	Yes	-
ALC	Yes	-
ALIF	Yes	-
BEIF	Yes	-
BUSOFF	(No)	An error counter reset by setting BUSOFF=1 also resets BUSOFF.
EIF	No	-
EPASS	No	-
EPIF	Yes	-
EWARN	No	-
EWL	Yes	-
FD_ISO	No	Register is writeable if RESET=1 and write-locked if 0.
FD_SEG_1	No	Register is writeable if RESET=1 and write-locked if 0.
FD_SEG_2	No	Register is writeable if RESET=1 and write-locked if 0.
FD_SWJ	No	Register is writeable if RESET=1 and write-locked if 0.
FD_SSPOF	No	Register is writeable if RESET=1 and write-locked if 0.

F		
KOER	Yes	-
LBME	Yes	-
LBMI	Yes	-
LLCAOT	No	-
LLCFORMAT	No	-
LLCPBYTES	No	-
LOM	No	-
PRESC	No	Register is writeable if RESET=1 and write-locked if 0.
RAFIF	Yes	-
RBALL	Yes	-
RBUF	(Yes)	All RB slots are marked as empty. RBUF contains unknown data.
REALIM	Yes	-
RECNT	No	An error counter reset is possible by setting BUSOFF=1.
REF_ID	No	-
REF_IDE	No	-
RETLIM	Yes	-
RFIF	Yes	-
RIF	Yes	-
ROIF	Yes	-
ROM	No	-
ROP	No	-
ROV	Yes	All RB slots are marked as empty.
RREL	Yes	-
RSTAT	Yes	-
SACK	Yes	-
STBY	No	-
TBE	Yes	-
TBF	Yes	-
TBPTR	No	-
TBSEL	Yes	TBUF fixed to point to PTB.
TBUF	(Yes)	All STB slots are marked as empty. Because of TBSEL TBUF points to the PTB.
TECNT	No	An error counter reset is possible by setting BUSOFF=1.
TEIF	Yes	-
TEW	No	-
TPA	Yes	-
TPE	Yes	-
TSA	Yes	-
TSALL	Yes	-

TSMODE	No	-
TSEN	No	-
TSPOS	No	-
TSNEXT	Yes	-
TSONE	Yes	-
TPIF	Yes	-
TSFF	Yes	All STB slots are marked as empty.
TSIF	Yes	-
TSSTAT	Yes	All STB slots are marked as empty.
TTEN	Yes	-
TTIF	Yes	-
TTPTR	No	-
TTS	No	-
TTTBM	No	-
TTYPE	No	-
TT_TRIG	No	-
TT_WTRIG	No	-
T_TPRESC	No	-
WTIF	Yes	-

21.4 Time-triggered CAN (TTCAN)

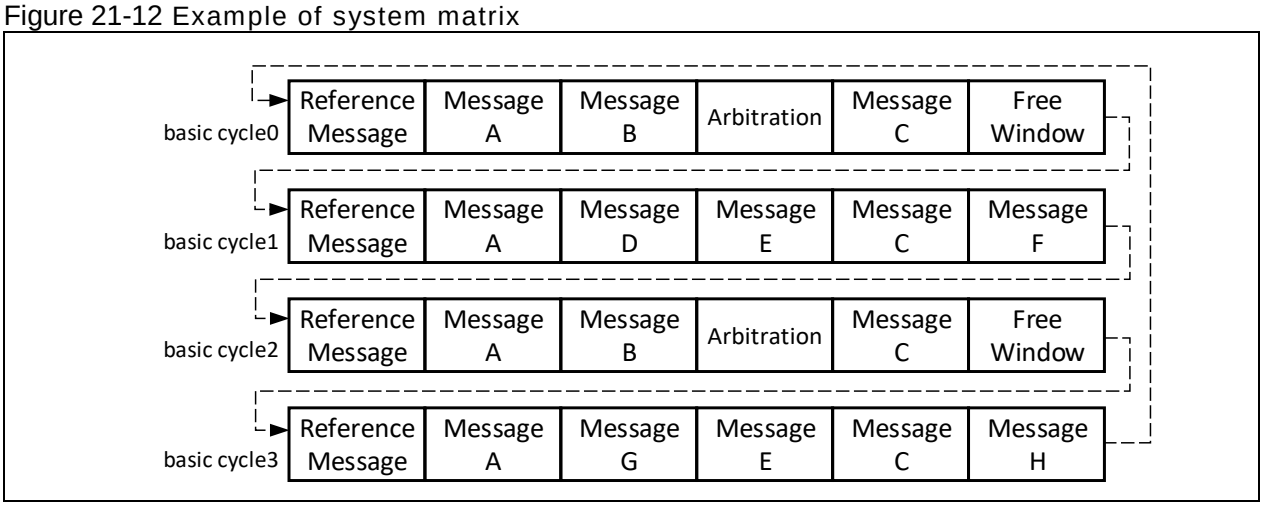
21.4.1 Introduction

Time-Triggered CAN (TTCAN) is an operation mode according to ISO 11898-4 where frames will be transmitted only at predefined time windows. There are three time window types:

- Exclusive time window (only one node is allowed to transmit one frame with a defined ID)
- Free time window (unused time window for further extension of the network)
- Arbitrating time window (several nodes may transmit a frame and arbitration takes place)

The timing is defined offline by a TTCAN system administrator and organized in a system matrix as shown in Figure 21-12. A row is called a basic cycle and it starts with a reference message. The reference message is transmitted by the time master. Other messages can be transmitted by any node including the time master.

Figure 21-12 Example of system matrix



The time of the SOF of a message is a Sync_Mark. The Sync_Mark of a reference message is the Ref_Mark. Timing is done by a free-running 16 bit timer. The difference between the timer and the value of the Ref_Mark is the cycle time. In other words: each basic cycle starts the cycle time at the Ref_Mark. The cycle time is stored in the RBUF as a time stamp.

The length of a time window is defined by the TTCAN administrator and it is long enough to carry one message. Therefore messages must be transmitted in single shot mode. There is only one exception from this rule: if several arbitrating time window are merged then it is possible to enable retransmission. But this needs to be disabled early enough to not influence the following time window.

Single shot means: no retransmission and no rearbitration.

If the trigger gets active this will be signaled to the host by an interrupt. Then the host needs to configure the next trigger for the next time window. This requires a real-time response by the host. The duration of one time window is the time that the host is allowed to use.

If CAN controller needs to operate as time master, then the reference message needs to be placed in a TBUF slot like all other messages and will be transmitted when the hardware trigger gets active. Detection of a reference message is done automatically by all of the time slaves and the time master. This is done by setting the correct REF_ID and REF_IDE bit of the reference message in the REF_MSG_x registers. Upon detection CAN-CTRL automatically updates the Ref_Mark which starts the cycle time.

Additionally to the trigger for messages, CAN-CTRL includes a watch trigger. This should be used to check whether the time since the last reference message has been too long. The host shall configure the watch trigger for the periodic case where each basic cycle is followed by the next basic cycle or for the event-triggered case where there is a time gap between the basic cycles and the next cycle is started upon an event. If the watch trigger gets active, this results in the watch interrupt.

Beside support for ISO 11898-4, CAN-CTRL offers a mixture of event-driven CAN communication combined with time-stamping for received messages. This mode is selected if register bit TTTBM=0. In this mode CAN-CTRL acts similar to event-driven CAN communication (TTEN=0), but reference messages can be detected and time-stamps for received messages are provided. Additionally only time triggers and the watch trigger are supported in this mode.

21.4.2 TBUF in TTCAN mode

The behavior if the TBUF depends on the register bit TTTBM. If TTTBM=1, then each TBUF slot can be addressed by the host controller which is required to use transmission triggers. Otherwise if TTTBM=0, then only time-stamping and time triggers are supported.

21.4.2.1 TBUF in TTCAN mode if TTTBM=1

In TTCAN mode with TTTBM=1 the STB is used as an array of message slots. Each slot can be addressed by TBPTR. The host can mark a slot as filled or as empty using TBF and TBE. Filled slots are write-locked. TBSEL and TSNEXT have no meaning in TTCAN mode and are ignored.

The PTB can be addressed by TBPTR=0. This makes the PTB useable as any other STB slot. In fact, the PTB has no special properties in this mode and is associated with the STB. This furthermore results in the fact that a successful transmission is always signaled using TSIF.

There is no FIFO operation and no priority decision for the TBUF in TTCAN mode. Furthermore only one frame can be selected for transmission.

A message trigger defines the time when a message needs to be transmitted (the beginning of a time window) and it selects the message using the pointer TTPTR. If the trigger event takes place, then selected message transmission is started. Finally the trigger interrupt is set to signal the host that the

next action needs to be prepared.

All transmissions can only be started using a trigger. See Chapter 9.5 for details. TPE, TSONE, TSALL and TPA are fixed to 0 and ignored in TTCAN mode.

21.4.2.2 TBUF in TTCAN mode if TTTBM=0

TTTBM=0 offers the combination of event-driven CAN communication and time-stamping for received messages.

In this mode the PTB and the STB provide the same behavior as if TTEN=0. Then the PTB always has higher priority than the STB and the STB can operated in FIFO mode (TSMODE=0) or in priority mode (TSMODE=1).

21.4.3 TTCAN operation

After power-up a time master needs to do the initialization as defined in ISO 11898-4. There may be up to 8 potential time masters in a CAN network. Each one has its own reference message ID (the last 3 bits of the ID). Potential time masters may try to transmit their reference message according to their priority. Lower-prioritized time masters shall try to transmit their reference messages later.

If TTEN is set, then the 16 bit timer is running. If a reference message is detected or if a time master successfully transmits its reference message, then CAN controller copies the Sync_Mark of this message to the Ref_Mark which sets the cycle time to 0. The reception of the reference message will set RIF respectively the successful transmission will set TPIF or TSIF. Then the host needs to prepare the trigger for the next action.

A trigger for an action can be a reception trigger. This just triggers the interrupt and it can be used to detect if an expected message is not received. Such a trigger can also be used for other actions, but this depends on the host application.

A different trigger is a transmission trigger. This starts the frame in the TBUF slot, where the trigger points to with TTPTR. If the TBUF slot is marked as empty, then no transmission is started, but the trigger interrupt is set.

It is possible that one host task updates a message slot with a new message if it is necessary by the host application. This could be e.g. a new sensor value. Later then if TTCAN requires this message to be transmitted, it will be activated by the trigger. In other words: one host task needs to be responsible for TTCAN and a different host task may update the message slots. This requires that one message slot is exclusively used for one message (e.g. slot 1 for a temperature sensor). If not enough TBUF slots are available then slots need to be shared by different messages.

The TTCAN host application needs to keep track of the system matrix. If a trigger gets active, then the host application needs to prepare the next transmission column. The host needs to handle also the basic cycles. The reference message includes the cycle count.

Note: Most operations in ISO 11989-4 require single shot transmission.

21.4.4 TTCAN timing

CAN-CTRL supports ISO 11898-4 level 1. This includes a 16 bit timer running at the speed of a CAN bit time (defined by PRESC, AC_SEG_1, AC_SEG_2). An additional prescaler is defined by T_PRESC. If TTEN=1 then the timer is counting continuously.

At the SOF of the message, the timer value is the Sync_Mark. If the message was a reference message, then this value is copied to the Ref_Mark. The cycle time is the timer value minus the Ref_Mark. It is the time that is used as time-stamp for received messages or as trigger time for messages, that need to be transmitted. An overflow protection is included and therefore the cycle time is always strictly monotonic inside a basic cycle.

ISO 11898-4 does not include CAN FD baud rate switching. CAN-CTRL therefore always operates the

timer with slow nominal bit rate. The timer is running freely and it is not influenced by synchronization or baud rate switching. Therefore a timer tick is not synchronous to the start of a CAN bit.
The timer is running in the CAN clock domain while all control and status bits are in the host clock domain. Therefore the pure timer value is not readable for the host as this would require clock domain crossing. Trigger events need to be used to synchronize actions required by the host to the timer. Interrupts because of trigger events, reception or transmission require clock domain crossing and therefore include some clocks delay. This only becomes relevant if after a trigger the host application decides to start a transmission. (Therefore it is recommended to use a transmission trigger instead.) In all other cases the host application has enough time to prepare all actions for the next trigger event (which is the next time window). Almost all hosts are fast enough to do a lot of actions during the duration of a CAN frame (which is the duration of a time window).
Timing according ISO 11898-4 level 1 is not perfect. The cycle time counts not synchronously with the CAN bits, because a CAN bit can be shortened or lengthened because of CAN synchronization. Therefore the cycle time of one node may differ compared to the cycle time of other nodes. In many cases this will be +/- 1 tick but it is not limited to this. The begin of a new basic cycle with the transmission of a reference message will resynchronize the nodes.

If a transmission trigger becomes active, then the appropriate transmission can only be started with the next CAN bit. Therefore the earliest point of transmission of the SOF of the frame will be at TT_TRIG+1.

21.4.5 TTCAN trigger types

The trigger type is defined by TTYPE. TTPTR is a pointer to a TB message slot and TT_TRIG defines the cycle time of the trigger.

Triggers and the related actions shall finish before the maximum cycle time 0xffff is reached as this is the maximum length of a basic cycle. Except for the immediate trigger, all triggers set TTIF if TTIE is enabled.

If TTTBM=0, only time triggers are supported. Using other triggers in this mode will result in setting TEIF

If a trigger is activated after a write access to TT_TRIG_1, then write access to TT_CFG_0, TT_CFG_1, TT_TRIG_0 and TT_TRIG_1 are locked until the trigger time is reached (TTIF is set if TTIE is set) or an error is detected (TEIF is set). Therefore no new trigger can override an active trigger. The write lock is also removed if TTEN=0.

21.4.5.1 Immediate trigger

An immediate trigger starts the immediate transmission of a frame that is pointed to by TTPTR. TTIF is not set. To start a trigger, TT_TRIG_1 needs to be written. The value that is written, does not care for an immediate trigger.

In TTCAN mode TPE, TSONE, TSALL cannot be used. The immediate trigger is the replacement. Only the transmission of one frame can be started with an immediate trigger. The host must not command a second immediate trigger before the first transmission has been completed successfully or unsuccessfully.

For an immediate trigger it is possible to limit re-arbitration and retransmission using register RELIM. A transmission can be aborted using TSA. (TPA has no meaning.)

If TTPTR of an immediate trigger points to an empty slot, then TEIF is set.

21.4.5.2 Time trigger

A time trigger just generates an event by setting TTIF and therefore generates an interrupt. No other actions are done.

The time trigger can be used as a reception trigger. If the node expects a message to be received in a time window, then if the message is missing and RIF is not set, a reception trigger can be used to signal this. A reception trigger shall be set after the latest moment when the message is expected to be successfully received.

If TT_TRIG is lower than the actual cycle time, then TEIF is set.

A time trigger can be used if TTTBM=0. This is the only trigger type, which is available in this mode.

21.4.5.3 Single shot transmit trigger

A single shot transmit trigger is intended to be used for exclusive time windows, where a message needs to be transmitted in single shot mode. The selected message is defined by TTPTR. Single shot mode is automatically used regardless of the state of RELIM. The register RELIM is ignored and stays unchanged.

Single shot transmit triggers are intended to be used for exclusive time windows. For this ISO 11898-4 defines a transmit enable window of up to 16 ticks of the cycle time. The register bits TEW(3:0)+1 define the number of ticks. A frame cannot be started if the bus is occupied by another frame. This should not happen in an exclusive time window, but the transmit enable window ensures that there is no delayed start which would result in a violation of the next time window. If the transmit enable window closes and the frame could not be started, then it is aborted. As a result the TB slot of the frame will be marked as empty and AIF will be set. The data of the frame in the TB slot will not be touched and therefore the slot only needs to be marked as filled again if the same data shall be transmitted in a next attempt.

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

21.4.5.4 Transmit start trigger

A transmit start trigger is intended to be used for merged arbitrating time windows, where several nodes may transmit messages and CAN arbitration takes place. The selected message is defined by TTPTR. The content of RELIM defines if re-arbitration or retransmission will take place.

If the selected frame cannot be transmitted (arbitration loss, several transmissions after an error), then the transmission can be stopped using the transmit stop trigger.

If TT_TRIG is lower than the actual cycle time, then TEIF is set and no action is done.

21.4.5.5 Transmit stop trigger

A transmit stop trigger is intended to be used to stop a transmission, that is started with the transmit start trigger (Chapter 21.4.5.4). If a transmission is stopped, then the frame is aborted. Therefore AIF is set and the TB slot of the frame is marked as empty. The data of the frame in the TB slot will not be touched and therefore the slot only needs to be marked as filled again if the same data shall be transmitted in a next attempt.

The behavior of a transmit stop trigger is similar to the end of the transmit enable window used by the single shot transmit trigger. If a transmit stop trigger points to an empty slot (which means, that the message has already been transmitted), then not action is done and TEIF is not set.

If TT_TRIG is lower than the actual cycle time, then TEIF is set but the stop is executed.

21.4.6 TTCAN watch trigger

In contrast to the generic trigger types explained in Chapter 21.4.5 the watch trigger has a dedicated interrupt flag WTIF. If the cycle count equal to the value defined by TT_WTRIG, then WTIF is set is WTIE is set.

The watch trigger is intended to be used if the time since the last valid reference message has been too long. Reference messages can be received in a periodic cycle or after an event. The host application needs to take care of this and has to adjust the watch trigger accordingly.

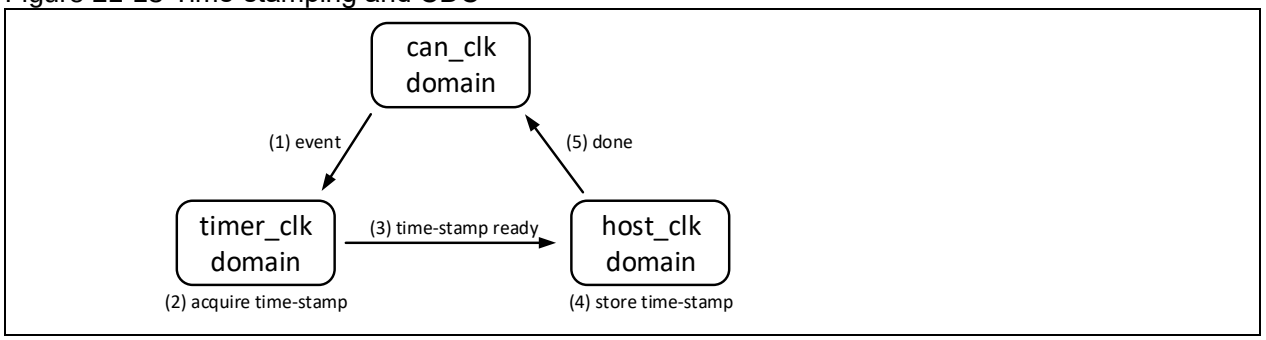
The default value of WTIE is 1 and therefore the default watch trigger 0xffff is automatically valid.

If TT_WTRIG is updated and it is lower than the actual cycle time, then TEIF is set.

21.5 CiA 603 time-stamping

CAN in Automation (CiA) defines in specification CiA 603 a method for time-stamping with at least 16 bits, which is optionally supported by CAN controller. The basic concept of CiA 603 is to have a free-running timer which counts clock cycles and not CAN bit times. As shown in Figure 21-13 after an event a copy of the timer is taken. This is the time-stamp. This timestamp is stored into RTS or TTS respectively where it can be read by the host. After it is stored, a handshake signal announces the completion of the action resulting in a CDC mechanism with 3 clock domains.

Figure 21-13 Time-stamping and CDC



Time-stamps are acquired at the sample point of SOF or the EOF bit where the frame is taken to be valid. This can be selected by the configuration bit TSPOS. Seven recessive bits after the ACK delimiter form the EOF of a CAN / CAN FD frame. A frame gets valid for a receiver at the last but one bit of EOF, but for a transmitter at the last bit of EOF.

Software-based time-stamping, which is widely used in many systems, relies on the reception and transmission interrupt. Time-stamping at EOF where the frame is valid is similar to software-based timestamping.

CiA 603 is supposed to support time-stamping and time-synchronization of AUTOSAR. For AUTOSAR one node in the CAN network is the time master. A time master transmits a synchronization message (SYNC message). The time-stamp of the SYNC message is acquired by the time master and all time slaves. The difference in time between the event of commanding a SYNC message until the time when the SYNC message actually gets transmitted will be transmitted in a follow-up (FUP) message by the time master. Therefore it is sufficient that CAN-CTRL supports only one time-stamp for transmitted frames (TTS) but individual time-stamps for all received frames (RTS). Generation of a time-stamp for transmitted frames can be enabled or disabled for each frame individually using bit TTSEN inside a TBUF slot.

CiA defines rules to read out the timer as well as to modify the timer. CAN-CTRL does not include the timer, but relies on an external timer. CAN-CTRL only includes the mechanism of time-stamping, the register to store TTS and the memory to store RTS for each frame.

Register bit TSEN enables or disables time-stamping in general. If disabled TTS and RTS are zero

21.6 CAN bit time

21.6.1 Bit rates

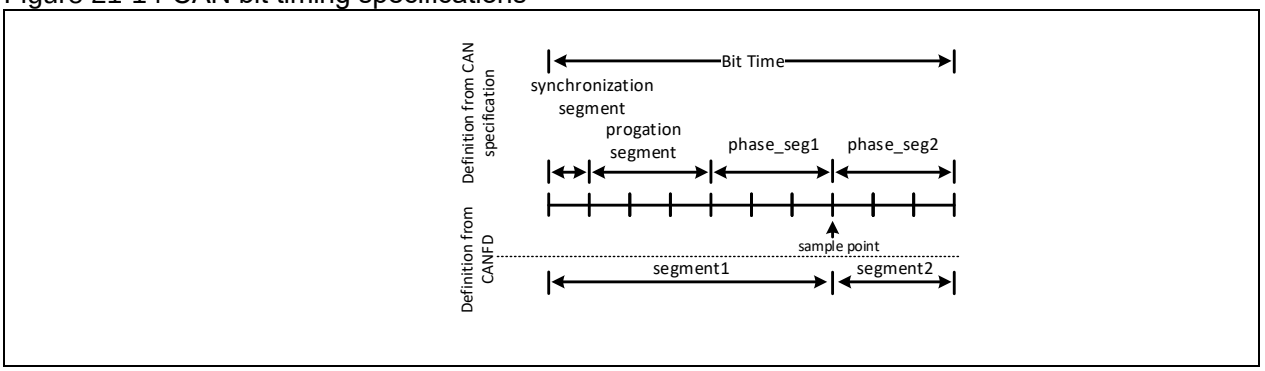
Classic CAN 2.0B defines a bit rate of up to 1Mbit/s. For CAN FD and CAN XL there is no fixed limitation. For real life systems the data rates are limited by the used transceiver, the desired CAN bus topology and the achievable clock frequency for CAN-CTRL which depends on the used target cell library.

CAN-CTRL can be programmed to arbitrarily chosen bit rates only limited by the range of the bit settings in the appropriate bit timing and prescaler registers.

Classic CAN operates at one selected bit rate for the entire frame. In contrast to this CAN FD starts a frame with the so-called “nominal bit rate” (slow) and the inside the frame switch to the so-called “data bit rate” (fast). Switching back to nominal bit rate is done at the end of the frame or after an error.

21.6.2 Bit timing definitions

Figure 21-14 CAN bit timing specifications



CAN communication defines mechanisms known as clock-data-recovery (“resynchronization”) as part of the CAN protocol specification. These mechanisms are based on the definition of a CAN bit as shown in Figure 21-14. For all classic and FD frames the definition of the CAN bit time follows the same concept: The CAN bit time BT consists of several segments. Each segment consists of a number of time quantas n_{TQ}. The duration of a time quanta tTQ is:

$$t_{TQ} = \frac{n_{prescaler}}{f_{can\_clk}}$$

The values of n_{TQ} and $n_{prescaler}$ have to be chosen depending on the system clock frequency $fcan\_clk$ to match the desired bit time BT. The bit rate BR = 1⁄BT. “X” refers to the placeorder with AC or FC as a prefix.

$$BT = n_{TQ} * t_{TQ} = n_{TQ} * \frac{n_{prescaler}}{f_{can_{clk}}} = t_{Seg\_1} + t_{Seg\_2}$$

$$t_{Seg\_1} = (x\_SEG\_1 + 2) * t_{TQ}$$

$$t_{Seg\_2} = (x\_SEG\_2 + 1) * t_{TQ}$$

$$t_{SJW} = (x\_SJW + 1) * t_{TQ}$$

$$n_{prescaler} = PRESC + 1$$

CAN-CTRL simplifies the bit timing configuration as shown in Figure 11-1: the synchronization segment, the propagation segment and the phase_seg1 are grouped into the so-called segment 1. The duration of segment 1 is t_{Seg_1} while duration of the segment 2 is t_{Seg_2}.

SJW is the synchronization jump width. Synchronization works in steps of one TQ. The purpose of synchronization is to ensure that the transition of the signal levels of the received data stream falls into the synchronization segment to shift the sample point to a position where the received signal is stable. If the transition arrives earlier or later, then a CAN node will shorten or lengthen a CAN bit by removing or adding a few TQs. SJW limits the number of TQ used for one synchronization event.

Propagation of signals in CAN networks takes some time. Signal reflection (“ringing”) is one of the

physical effects, which are responsible for the propagation delay. Propagation shall be completed within the propagation segment of a CAN bit.

CAN nodes do resynchronization to the received bit stream. Resynchronization is done at recessive-todominant edges. CAN nodes insert so-called stuff-bits to guarantee that such resynchronization events occur at a known maximum interval. As a consequence there are several bits to be received without resynchronization and because of the oscillator tolerances transmitter and receiver will slightly drift away. The next resynchronization event will compensate this. The <phase_seg1> and <phase_seg2> segments are reserved for this phase error between transmitter and receiver. Therefore <phase_seg1> and <phase_seg2> are of equal duration.

The maximum value of SJW is equal to <phase_seg2>. In other words: the amount of resynchronization is at maximum as big as the expected maximum phase error between two resynchronization events.

The configuration of the CAN bit time with the goal to reach the highest possible bit rate is a very complex topic and as written in the INM document CAN in Automation provides a guideline in the CiA601-3 document. In short:

- Signal propagation in a network is related to the propagation segment
- Oscillator tolerances between transmitter and receiver is related to the phase segment 1 and 2

But some rules of the thumb will also give a good point to start:

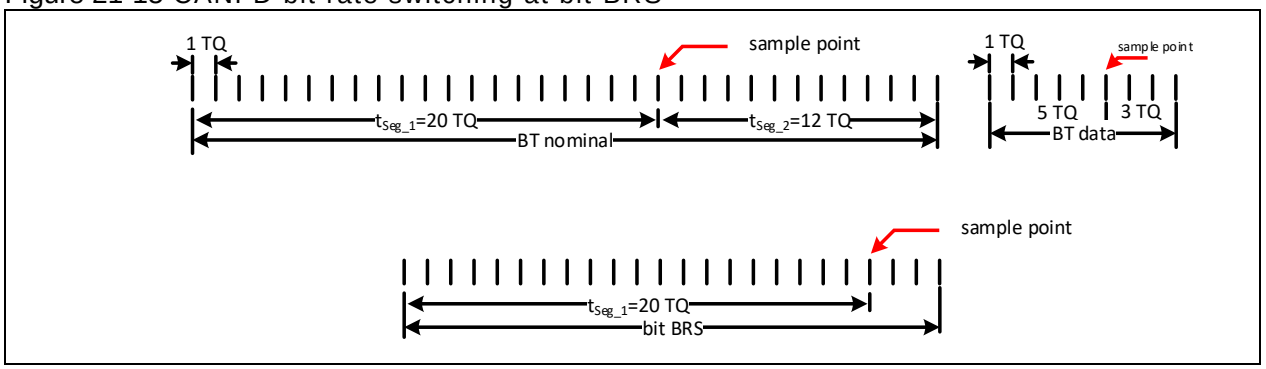
- The sample point should be roughly in the range of 60% to 80% of the bit time. Often it is the best choice to set it to 75% to 80%.
- If possible a lower prescaler is preferable resulting in as many TQ per bit as possible.
- In most cases it is preferable to set SJW equal to segment 2

21.6.3 CANFD bit rate switching and the sample point

If CAN FD frames with bit rate switching are used (BRS=1), then the exact position of the sample point is important and needs to be equal for all nodes of a CAN network.

The bit rate is switched after the sample point of the BRS bit and the sample point of the CRC delimiter. Therefore the lengths of these bits are intermediate. As can be seen in Figure 21-15 the position of the sample point makes a big difference for the absolute length of the BRS bit. If the data rate is much higher than the nominal bit rate then an earlier or later sample point may lead to false samples at fast data bit rate.

Figure 21-15 CANFD bit rate switching at bit BRS



In summary the position of the sample point is very important for bit rate switching. Therefore there is the general recommendation for all CAN FD nodes to use the same timing parameters in a CAN network. In others words segment 1 and segment 2 need to be configured to the same length for all nodes. Furthermore it is recommended to use also the same length of one TQ to make synchronization for all nodes equal.

21.6.4 TDC and RDC

For CAN FD nodes transmitter delay compensation (TDC) can be optionally enabled while receiver delay compensation (RDC) is automatically active (Classic CAN2.0 frame node does not support TDC and RDC). TDC and RDC have the following background: For communication with CAN FD data bit rate it may be that the delay of the transmitting transceiver or the delay of the bus is bigger than a bit time. This needs to be compensated. Without TDC, the bit rate in the data-phase of FD or frames is limited by the fact that the transmitter detects a bit error if it cannot receive its own transmitted bit latest at the sample point of that bit.

Figure 21-16 Transmitter delay

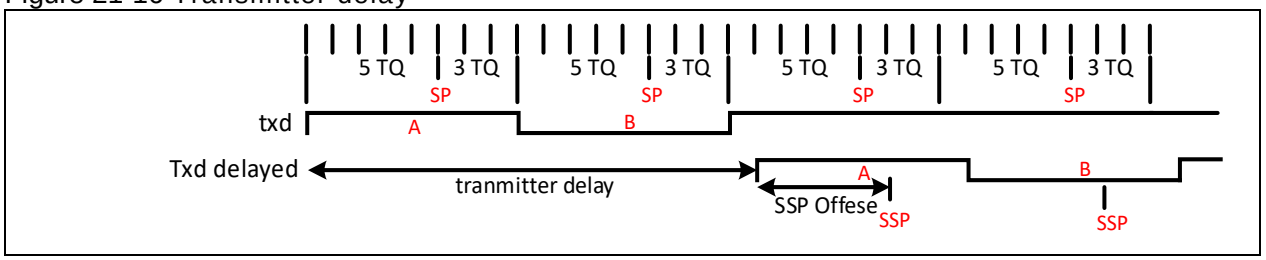


Figure 21-16 gives an example of the effect of a big transmitter delay. In this figure a <txd> data stream is shown starting with bits A and B. One bit time consists of $t_{Seg\_1} = 5TQ$ before the Sample Point (SP) and $t_{Seg\_2} = 3TQ$ behind. In this example the transmitter delay is greater than 2 bit times. Therefore the original SP cannot be used to sample the correct bit value and the CAN FD specification defines an additional Secondary Sample Point (SSP). If TDC is enabled (when SSPOFF $\neq 0$ and CES=1) then CANCTRL automatically determines the transmitter delay. The position of the SSP is the measured transmitter delay plus the SSP Offset which is defined by the configuration bits SSPOFF. SSPOFF must be given as a number of TQ and it is suggested to set the SSP offset equal to $t_{Seg\_1} + 1$.

Hint: it is recommended to set it to $t_{Seg\_1} + 1$ and not to t_{Seg_1}, because of the quantization error when sampling some data.

ISO 11898-1:2015 requires to use only PRESC=0 or 1 if TDC is used. With this requirement CAN controller is capable of automatically determining a transmitter delay of up to 3 bit times of the fast data bit rate.

Side note: apparently RDC is only in effect if $n_{prescaler} > 1$

21.7 CAN registers

ACF/RBUF/TBUF register can only be read/written by words (32 bits). Others peripheral registers are written acess by bytes (8 bits) and read access by words (32 bits).

Table 21-9 CAN register map and reset values

Register	Offset	Reset value
TNCFG	0000h	0x0200 0000
ACTIME	0004h	0x0505 0008
FDTIME	0008h	0x0202 0003
LBTCFG	0010h	0x7700 0000
STS	0014h	0x0000 0000
TSTAT	001Ch	0x0000 0000
TTS	0020h	0x0000 0000
CTRLSTAT	0028h	0x0090 0080
ERR	002Ch	0x0000 0010
REFMSG	0030h	0x0000 0000
TTCFG	0034h	0x0000 0000
TTTRIG	0038h	0xFFFF 0000
ACF	**0044h~0064h**	
ACFCTRL	0044h	0x0001 0000
FCID	0048h	0xXXXX XXXX
FCFMT	004Ch	0xXXXX XXXX
FCTYP	0050h	0xXXXX XXXX
FMID	0058h	0xXXXX XXXX
FMFMT	005Ch	0xXXXX XXXX
FMTYP	0060h	0xXXXX XXXX
Res 1	0064h	0xXXXX XXXX
RBUF（including RTS）	**0070h~00CBh**	
RBID	0070h	0xXXXX XXXX
RBFMT	0074h	0xXXXX XXXX
RBTYP	0078h	0xXXXX XXXX
RBDAT1	0080h	0xXXXX XXXX
RBDAT2	0084h	0xXXXX XXXX
…	…	…
RBCIA1	T+0	0xXXXX XXXX
RBCIA2	T+4	0xXXXX XXXX
RBTTCAN	T+8	0xXXXX XXXX
TBUF	**00CCh~011Bh**	
TBID	00CCh	0xXXXX XXXX
TBFMT	00D0h	0xXXXX XXXX
TBTYP	00D4h	0xXXXX XXXX
Res 2	00D8h	0xXXXX XXXX
TBDATA1	00DCh	0xXXXX XXXX
TBDATA2	00E0h	0xXXXX XXXX
…	…	…

LLCFORMAT	0124h	0x0000 0000
LLCSIZE	0128h	0x0010 0000
INTEN	0140h	0x0000 0000

21.7.1 CAN CiA 603 time-stamp and node control register (TNCFG)

Bit	Name	Reset value	Type	Description
Bit 31: 26	Reserved	0x0	resd	Kept at default value.
Bit 25	TSPOS	0x1	rw	Time stamping position 0: SOF 1: EOF TSPOS can only be changed if TSEN=0, but it is possible to modify TSPOS with the same write access that sets TSEN=1.
Bit 24	TSEN	0x0	rw	Time stamping enable 0: Disabled 1: Enabled
Bit 23: 18	Reserved	0x0	resd	Kept at default value.
Bit 17	ROP	0x0	rw	Restricted operation 0: Restricted operation disabled 1: Restricted operation enabled Note: ROP cannot be changed while a transmission is active. No transmission can be started if ROP is enabled.
Bit 16: 0	Reserved	0x0	resd	Kept at default value.

21.7.2 Classic CAN2.0B bit timing register (ACTIME)

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at default value.
Bit 30: 24	AC_SJW	0x5	ro	Synchronization jump width t_{SJW} = (AC_SJW + 1) * t_{TQ}
Bit 23	Reserved	0x0	resd	Kept at default value.
Bit 22: 16	AC_SEG_2	0x5	rw	Bit timing segment 2 t_{SEG_2} = (AC_SEG_2 + 1) * t_{TQ}
Bit 15: 9	Reserved	0x0	resd	Kept at default value.
Bit 8: 0	AC_SEG_1	0x8	rw	Bit timing segment 1 The sample point will be set to t_{SEG_1} = (AC_SEG_1 + 2) * t_{TQ}

21.7.3 CANFD bit timing register (FDTIME)

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at default value.
Bit 30: 24	FD_SJW	0x2	rw	Synchronization jump width t_{SJW} = (FD_SJW + 1) * t_{TQ}
Bit 23	Reserved	0x0	resd	Kept at default value.
Bit 22: 16	FD_SEG_2	0x2	rw	Bit timing segment 2 t_{SEG_2} = (FD_SEG_2 + 1) * t_{TQ}
Bit 15: 8	Reserved	0x0	resd	Kept at default value.

Bit 7: 0	FD_SEG_1	0x3	rw	Bit timing segment 1 The sample point will be set to t_{SEG_1} = (FD_SEG_1 + 2) * t_{TQ}

21.7.4 CAN limit and bit time configuration register (LBTCFG)

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at default value.
Bit 30: 28	RETLIM	0x7	rw	Retransmission limit 111: Unlimited (only limited by the transmit error counterTECNT) 110: 7 attempts … 000: 1 attempt (no retransmission) If there is any error during transmission, a CAN node is able to automatically retry it as soon as the bus is idle. The number of retransmission attempts can be limited using RETLIM. RETLIM may be updated at any time, but after an update the content of the register needs to be transferred to the CAN protocol machine using clock domain crossing. During this short amount of time (a few clocks) RETLIM is write-locked.
Bit 27	Reserved	0x0	resd	Kept at default value.
Bit 26: 24	REALIM	0x7	rw	Re-arbitration limit 111: Unlimited 110: 7 attempts … 000: 1 attempt (no re-arbitration) If two or more CAN nodes try to transmit frames simultaneously, then the lower-priority frames lose arbitration and silently step back without interrupting the higher-priority frame. A CAN node is able to automatically retry it as soon as the bus is idle. The number of rearbitration attempts can be limited using REALIM. RELIM may be updated at any time, but after an update the content of the register needs to be transferred to the CAN protocol machine using clock domain crossing. During this short amount of time (a few clocks) REALIM is write-locked.
Bit 23: 16	Reserved	0x0	resd	Kept at default value.
Bit 15: 8	FD_SSPOFF	0x0	rw	Secondary sample point offset Transmitter Delay Compensation (TDC) is enabled if SSPOFF≠0 and CES=1. TDC will be activated during the data phase of a CAN FD frame if BRS is active. The transmitter delay plus SSPOFF defines the time of the secondary sample point for TDC. SSPOFF is given as a number of TQ.
Bit 7: 5	Reserved	0x0	resd	Kept at default value.
Bit 4: 0	PRESC	0x0	rw	Prescaler The prescaler divides the system clock to synthesize the time quanta TQ. Nprescaler = PRESC + 1 or Ttq = Nprescaler/fcan_clk

21.7.5 CAN status register (STS)

Bit	Name	Reset value	Type	Description
Bit 31	EWARN	0x0	ro	Error warning limit reached 0: RECNT and TECNT counters are less than EWL 1: One of the error counters RECNT or TECNT is equal or bigger than EWL
Bit 30	EPASS	0x0	ro	Error passive status 0: Node is error active 1: Node is error passive
Bit 29: 14	Reserved	0x00	resd	Kept at default value.
Bit 13	WTIF	0x0	rw1c	Watch trigger interrupt flag WTIF will be set if the cycle count reaches the limited defined by TT_WTRIG and WTIE is set.
Bit 12	TEIF	0x0	rw1c	Trigger error interrupt flag The conditions when TEIF will be set.There is no bit to enable or disable the handling of TEIF.
Bit 11	TTIF	0x0	rw1c	Time trigger interrupt flag TTIF will be set if TTIE is set and the cycle time is equal to the trigger time TT_TRIG. Writing an one to TTIF resets it. Writing a zero has no impact. TTIF will be set only once. If TT_TRIG gets not updated, then TTIF will be not set again in the next basic cycle.
Bit 10	EPIF	0x0	rw1c	Error passive interrupt flag EPIF will be activated if EPASS changes and if EPIE=1
Bit 9	ALIF	0x0	rw1c	Arbitration lost interrupt flag
Bit 8	BEIF	0x0	rw1c	Bus error interrupt flag
Bit 7	RIF	0x0	rw1c	Receiv interrupt flag 0: No frame has been received. 1: Data or a remote frame has been received and is available in the receive buffer.
Bit 6	ROIF	0x0	rw1c	RB overflow interrupt flag 0: No RB overwritten 1: At least one received frame has been overwritten in the RB
Bit 5	RFIF	0x0	rw1c	RB full interrupt flag 0: RB FIFO is not full 1: All RB are full, If no RB wll be released until the next valid frame is received, the oldest frame will be lost.
Bit 4	RAFIF	0x0	rw1c	RB almost full interrupt flag 0: Number of illed RB slots ＜AFWL_i 1: Number of filled RB slots ≥AFWL_i
Bit 3	TPIF	0x0	rw1c	Transmission primary interrupt flag 0: No transmission of the PTB has been completed 1: The requested transmission of the PTB has been successfully completed In TTCAN mode, TPIF will never be set. Then only 只有 TSIF is valid.
Bit 2	TSIF	0x0	rw1c	Transmission secondary interrupt flag 0: No transmission of the STB has been completed succesfully 1: The requested transmission of the STB has been successfully completed

				In TTCAN mode TSIF will signal all successful transmissions, regardless of storage location of the frame.
Bit 1	EIF	0x0	rw1c	Error interrupt flag 0: There has been no change. 1: The border of the error warning limit has been crossed in either direction, or the BUSOFF bit has been changed in either direction.
Bit 0	AIF	0x0	rw1c	Abort interrupt flag 0: No abort has been executed 1: After setting TPA or TSA the appropriated frame(s) have been aborted. It is recommended to not set both TPA and TSA simultaneously because both source AIF.

21.7.6 CAN transmit status register (TSTAT)

Bit	Name	Reset value	Type	Description
Bit 31: 27	Reserved	0x0	resd	Kept at default value
Bit 26: 24	TSTAT_2	0x0	ro	Current transmission status code 000: No transmission in progress (IDLE) 001: Transmission active without any issues (ONGOING) 010: Arbitration lost. Re-arbitration may takep place with respect to REALIM (LOST_ARBITRATION) 011: Transmission successfully completed (TRANSMITTED) 100: Transmission aborted (TPA, TSA) 101: Transmission error. Retransmission may take place with respect to RETLIM (DISTURBED) 110: Misconfiguration of the frame format in the LCC frame (REJECTED) 111: Reserved
Bit 23: 16	HANDLE_2	0x0	ro	Handle for frame identification It is suggested, that the host application writes the value of a software counter to HANDLE. Such a software counter could be incremented with every new frame to be transmitted and should roll over.
Bit 15: 11	Reserved	0x0	resd	Kept at default value
Bit 10:8	TSTAT_1	0x0	ro	Final transmission status code 000: No transmission in progress (IDLE) 001: Transmission active without any issues (ONGOING) 010: Transmission lost. Re-arbitration may take place with respect to REALIM (LOST_ARBITRATION) 011: Transmission successfully completed (TRANSMITTED) 100: Transmission aborted (TPA,TSA) (ABORTED) 101: Transmission error. Retransmission may take place with respect to RETLIM (DISTURBED) 110: Misconfiguration of the frame format in the LCC frame (REJECTED) 111: Reserved
Bit 7: 0	HANDLE_1	0x0	ro	Handle for frame identification It is suggested, that the host application writes the value of a software counter to HANDLE. Such a software counter could be incremented with every new frame to be transmitted and should roll over.

21.7.7 CAN transmission time stamp 32 bit (TTS)

Bit	Name	Reset value	Type	Description
Bit 31: 0	TTS	0x0	ro	Transmission time stamp TTS holds the time stamp of the last transmitted frame for CiA 603 time stamping. Every new frame overwrites TTS if TTSEN=1. The time-stamp is 32 bit. The unused bits are forced to 0. The TTS is intended to be used by the time master to acquire the time-stamp of the SYNC message.

21.7.8 CAN control and status register (CTRLSTAT)

Bit	Name	Reset value	Type	Description
Bit 31	SACK	0x0	rw	Self-acknowledge 0: No self-ACK 1: Self-ACK when LBME=1
Bit 30	ROM	0x0	rw	Receive buffer overflow mode 0: The oldest frame will be overwritten 1: The new frame will not be stored
Bit 29	ROV	0x0	ro	Receive buffer overflow 0: No overflow 1: Overflow. At least one frame is lost ROV is cleared by setting RREL=1.
Bit 28	RREL	0x0	rw	Receive buffer release The host controller has read the actual RB slot and releases it. Afterwards CAN-CTRL points to the next RB slot. RSTAT gets updated. 0: No release 1: Release: The host has read the RB
Bit 27	RBALL	0x0	rw	Receive buffer stores all data frames 0: Normal operation 1: RB stores correct data frames as well as data frame swith error
Bit 26	Reserved	0x0	resd	Kept at default value.
Bit 25: 24	RSTAT	0x00	ro	Receive buffer status 00: Empty 01: > empty and < almost full (AFWL) 10: ≥ almost full (programmable threshold by AFWL) but not full and no overflow 11: full (stays set in case of overflow – for overflow signaling see ROV)
Bit 23	FD_ISO	0x1	rw	CAN FD ISO mode 0: Bosch CAN FD (non-ISO) mode 1: ISO CAN FD mode (ISO 11898-1:2015) ISO CAN FD mode has a different CRC initialization value and an additional stuff bit count. Both modes are incompatible and must not be mixed in one CAN network. This bit has no impact to CAN 2.0B. This bit is only writeable if RESET=1.
Bit 22	TSNEXT	0x0	rw	Transmit buffer secondary next 0: No action

				1: STB slot filled, select next slot After all frame bytes are written to the TBUF registers, the host controller has to set TSNEXT to signal that this slot has been filled. Then CAN-CTRL connects the TBUF registers to the next slot. Once a slot is marked as filled a transmission can be started using TSONE or TSALL. It is possible to set TSNEXT and TSONE or TSALL together in one write access. TSNEXT has to be set by the host controller and is automatically reset by CAN-CTRL immediately after it was set. Setting TSNEXT is meaningless if TBSEL=0. In this case TSNEXT is ignored and automatically cleared. It does not do any harm. If all slots of the STB are filled, TSNEXT stays set until a slot becomes free. TSNEXT has no meaning in TTCAN mode and is fixed to 0.
Bit 21	TSMODE	0x0	rw	Transmit buffer secondary operation mode 0: FIFO mode 1: Priority decision mode In FIFO mode frames are transmitted in the order in that they are written into the STB. In priority decision mode the frame with the highest priority in the STB is automatically transmitted first. The ID of a frame is used for the priority decision. A lower ID means a higher priority of a frame. A frame in the PTB has always the highest priority regardless of the ID. TSMODE shall be switched only if the STB if empty.
Bit 20	TTTBM	0x1	rw	TTCAN transmit buffer mode If TTEN=0 then TTTBM is ignored, otherwise the following is valid: 0: separate PTB and STB, behavior defined by TSMODE 1: full TTCAN support: buffer slots selectable by TBPTR and TTPTR For event-driven CAN communication (TTEN=0), the system provides PTB and STB and the behavior of the STB is defined by TSMODE. Then TTTBM is ignored. For time-triggered CAN communication (TTEN=1) with full support of all features including time-triggered transmissions, TTTBM=1 needs to be chosen. Then the all TB slots are addressable using TTPTR and TBPTR. For time-triggered CAN communication (TTEN=1) with only support of reception timestamps, TTTBM=0 can be chosen. Then the transmit buffer acts as in event-driven mode and the behavior can be selected by TSMODE. TTTBM shall be switched only if the TBUF is empty.
Bit 19	Reserved	0x0	resd	Kept at default value.
Bit 18	TSFF	0x0	rw	If TTEN=0 or TTTBM=0: Transmit secondary buffer full flag 1: STB is filled with the maximal number of frames 0: STB is not filled with the maximal number of frames If TTEN=1 and TTTBM=1: Transmit buffer slot full flag 1: The buffer slot selected by TBPTR is filled 0: The buffer slot selected by TBPTR is empty The update of TSFF after setting TSNEXT=1 takes a one additional cycle of <host_clk>. Therefore reading TSFF must not be done immediately after setting TSNEXT=1.
Bit 17: 16	TSSTAT	0x0	ro	Transmit secondary status bits

				If TTEN=0 or TTTBM=0 00: STB is empty 01: STB is less than or equal to 1/2 full 10: STB more than 1/2 full 11: STB is full If TTEN=1 and TTTBM=1: 00: PTB and STB are empty 01: PTB and STB are not empty and not full 11: PTB and STB are full The update of TSSTAT after setting TSNEXT=1 takes a one additional cycle of <host_clk>. Therefore reading TSSTAT must not be done immediately after setting TSNEXT=1.
Bit 15	TBSEL	0x0	rw	Transmit buffer select Selects the transmit buffer to be loaded with a frame. Use the TBUF registers for access. TBSEL needs to be stable all the time the TBUF registers are written and when TSNEXT is set. 0: PTB (high-priority buffer) 1: STB The bit will be reset to the hardware reset value if (TTEN=1 and TTTBM=1).
Bit 14	LOM	0x0	rw	Listen only mode 0: Disabled 1: Enabled LOM cannot be changed while a transmission is active. No transmission can be started if LOM is enabled and LBME is disabled. LOM=1 and LBME=0 disables all transmissions. LOM=1 and LBME=1 disables the ACK for received frames and error frames, but enables the transmission of own frames.
Bit 13	STBY	0x0	rw	Transceiver standby mode 0: Disabled 1: Enabled This register bit is connected to the output signal stby which can be used to control a standby mode of a transceiver. STBY cannot be set to 1 if TPE=1, TSONE=1 or TSALL=1. If the host sets STBY to 0 then the host needs to wait for the time required by the transceiver to start up before the host requests a new transmission.
Bit 12	TPE	0x0	rw	Transmit primary enable 0: No transmission for the PTB 1: Transmission enable for the frame in the high-priority PTB If TPE is set, the frame from the PTB will be transmitted at the next possible transmit position. A started transmission from the STB will be completed before, but pending new frames are delayed until the PTB frame has been transmitted. TPE stays set until the frame has been transmitted successfully or it is aborted using TPA.

				The host controller can set TPE to 1 but can not reset it to 0. This would only be possible using TPA and aborting the frame. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0), (TTEN=1 and TTTBM=1) or ROP=1.
Bit 11	TPA	0x0	rw	Transmit primary abort 0: No abort 1: Aborts a transmission from PTB which has been requested by TPE=1 but not started yet. (The data bytes of the frame remains in the PTB.) The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TPA automatically de-asserts TPE. The host controller can set TPA to 1 but can not reset it to 0. During the short time while CAN-CTRL resets the bit, it cannot be set by the host. The bit will be reset to the hardware reset value if RESET=1 or (TTEN=1 and TTTBM=1). TPA should not be set simultaneously with TPE.
Bit 10	TSONE	0x0	rw	Transmit secondary one frame 0: No transmission for the STB 1: Transmission enable of one in the STB. In FIFO mode this is the oldest frame and in priority mode this is the one with the highest priority. TSONE in priority mode is difficult to handle, because it is not always clear which frame will be transmitted if new frames are written to the STB meanwhile. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. TSONE stays set until the frame has been transmitted successfully or it is aborted using TSA. The host controller can set TSONE to 1 but can not reset it to 0. This would only be possible using TSA and aborting the frame. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0), (TTEN=1 and TTTBM=1) or ROP=1.
Bit 9	TSALL	0x0	rw	Transmit secondary all frames 0: No transmission for the STB 1: Transmission enable of all frames in the STB. The controller starts the transmission as soon as the bus becomes vacant and no request of the PTB (bit TPE) is pending. TSALL stays set until all frames have been transmitted successfully or they are aborted using TSA. The bit will be reset to the hardware reset value if RESET=1, STBY=1, (LOM=1 and LBME=0), (TTEN=1 and TTTBM=1) or ROP=1.
Bit 8	TSA	0x0	rw	Transmit secondary abort 0: No abort 1: Aborts a transmission from STB which has been requested but not started yet. For a TSONE transmission, only one frame is aborted while for a TSALL Transmission, all frames are aborted. One or all frame slots will be released which updates TSSTAT.

				All aborted frames are lost because they are not accessible any more. If in priority mode a TSONE transmission is aborted, then it is not clear which frame will be aborted if new frames are written to the STB meanwhile. 。 The bit has to be set by the host controller and will be reset by CAN-CTRL. Setting TSA, automatically de-asserts TSONE or TSALL respectively. The host controller can set TSA to 1 but can not reset it to 0. The bit will be reset to the hardware reset value if RESET=1. TSA should not be set simultaneously with TSONE or TSALL.
Bit 7	RESET	0x1	rw	RESET request bit 0: No local reset of CAN-CTR 1: The host controller performs a local reset of CAN-CTRL. The some register (e.g for node configuration) can only be modified if RESET=1. Bit RESET forces several components to a reset state. Note that a CAN node will participate in CAN communication after RESET is switched to 0 after 11 CAN bit times. This delay is required by the CAN standard (bus idle time). If RESET is set to 1 and immediately set to 0, then it takes some time until RESET can be read as 0 and becomes inactive. The reason is clock domain crossing from host to CAN clock domain. RESET is held active as long as needed depending on the relation between <host_clk> and <can_clk>.
Bit 6	LBME	0x0	rw	Loop back mode, external 0: Disabled 1: Enabled LBME cannot be changed while a transmission is active. LBME shall not be changed together with TPE, TSONE or TSALL.
Bit 5	LBMI	0x0	rw	Loop back mode, internal 0: Disabled 1: Enabled LBMI cannot be changed while a transmission is active. LBMI shall not be changed together with TPE, TSONE or TSALL.
Bit 4: 1	Reserved	0x0	resd	Kept at default value.
Bit 0	BUSOFF	0x0	rw	Bus off 0: The controller status is “bus on” 1: The controller status is “bus off” Writing a 1 to BUSOFF will reset TECNT and RECNT. This should be done only for debugging.

Note: Setting both TSONE and TSALL is meaningless. While TSALL is already set, it is impossible to set TSONE and vice versa. If both TSONE and TSALL are set simultaneously then TSALL wins and TSONE is cleared by CAN-CTRL.

21.7.9 CAN error configuration and status register (ERR)

Bit	Name	Reset value	Type	Description
Bit 31: 24	TECNT	0x0	ro	Transmit error count TECNT is incremented and decremented as defined in the CAN specification. In case of the “bus off state” TECNT may overflow.
Bit 23: 16	RECNT	0x0	ro	Receive error count RECNT is incremented and decremented as defined in the CAN specification. RECNT does not overflow.
Bit 15: 13	KOER	0x0	ro	Error code 000: No error 001: Bit error 010: Form error 011: Stuff error 100: Acknowledgement error 101: CRC error 110: Other error (dominant bits after own error flag, received active Error Flag too long, dominant bit during Passive-Error-Flag after ACK error) 111: Reserved KOER is updated with each new error. Therefore it stays untouched when frames are successfully transmitted or received.
Bit 12: 8	ALC	0x0	ro	Arbitration Lost Capture (bit position in the frame where the arbitration has been lost)
Bit 7: 4	AFWL	0x1	rw	Receive buffer almost full warning limit AFWL defines the internal warning limit with nRB the number of available RB slots. The RAFIF is set when the number of filled RB slots are equal to AFWL. The valid range is AFWL=[1…6]. AFWL=0 is meaningless and automatically treated as 0x1. AFWL> 6 is meaningless and automatically treated as 6 AFWL= 6 *is a valid value, but note that RFIF also exists.*
Bit 3:0	EWL	0xB	rw	Programmable error warning limit =(EWL+1)*8 Possible Limit values: 8, 16, … 128. The value of EWL controls EIF. EWL needs to be transferred using CDC from host to CAN clock domain. During transfer EWL register bits are write-locked for the host for a few clocks until CDC is complete.

21.7.10 CAN TTCAN: reference message (REFMSG)

Bit	Name	Reset value	Type	Description
Bit 31	REF_IDE	0x0	rw	Reference message IDE bit
Bit 30: 29	Reserved	0x0	resd	Kept at default value.
Bit 28: 0	REF_ID	0x0	rw	Reference message identifier If REF_IDE is 1: REF_ID(28:0) is valid (base ID + identifier extension)

				0: REF_ID(10:0) is valid (base ID only) REF_ID is used in TTCAN mode to detect a reference message. This holds for time slaves (reception) as well as for the time master (transmission). If the reference message is detected and there are no errors, then the Sync_Mark of this frame will become the Ref_Mark. REF_ID(2:0) is not tested and therefore the appropriate register bits are forced to 0. These bits are used for up to 8 potential time masters. CAN-CTRL recognizes the reference message only by ID. The payload is not tested. Additional note: A time master will transmit a reference message in the same way as a normal frame. REF_ID is intended for detection of a successful transmission of a reference message.

21.7.11 CAN TTCAN: trigger configuration register (TTCFG)

Bit	Name	Reset value	Type	Description
Bit 31: 27	Reserved	0x0	resd	Kept at default value.
Bit 26: 25	T_PRESC	0x0	rw	TTCAN timer prescaler 00: 1 01: 2 10: 4 11: 8 The TTCAN time base is a CAN bittime defined by PRESC, AC_SEG_1 and AC_SEG_2. With T_PRESC an additional prescaling factor of 1, 2, 4 or 8 is defined. T_PRESC can only be modified if TTEN=0, but it is possible to modify T_PRESC and set TTEN simultaneously with one write access.
Bit 24	TTEN	0x0	rw	Time trigger enable 1: TTCAN enabled, timer is running 0: disabled
Bit 23	TBE	0x0	rw	Set TB slot to “Empty” 1: slot selected by TBPTR shall be marked as “empty” 0: no action TBE is automatically reset to 0 as soon as the slot is marked as empty and TSFF=0. If a transmission from this slot is active, then TBE stays set as long as either the transmission completes or after a transmission error or arbitration loss the transmission is not active any more. If both TBF and TBE are set, then TBE wins.
Bit 22	TBF	0x0	rw	Set TB slot to “Filled” 1 - slot selected by TBPTR shall be marked as “filled” 0 - no action TBF is automatically reset to 0 as soon as the slot is marked as filled and TSFF=1. If both TBF and TBE are set, then TBE wins.
Bit 21: 16	TBPTR	0x00	rw	Pointer to a TB frame slot 0x00: Pointer to the PTB Others: Pointer to a slot in the STB

				The frame slot pointed to by TBPTR is readable / writable using the TBUF registers. Write access is only possible if TSFF=0. Setting TBF to 1 marks the selected slot as filled and setting TBE to 1 marks the selected slot as empty. TBSEL and TSNEXT are unused in TTCAN mode and have no meaning. TBPTR can only point to buffer slots, that exist in the hardware. Unusable bits of TBPTR are fixed to 0. TBPTR is limited to the PTB and 63 STB slots. More slots cannot be used in TTCAN mode. If TBPTR is too big and points to a slot that is not available, then TBF and TBE are reset automatically and no action takes place.
Bit 15: 12	TEW	0x0	rw	Transmit enable window For a single shot transmit trigger there is a time of up to 16 ticks of the cycle time where the frame is allowed to start. TWE+1 defines the number of ticks. TEW=0 is a valid setting and shortens the transmit enable window to 1 tick
Bit 11	Reserved	0x0	resd	Kept at default value.
Bit 10: 8	TTYPE	0x00	rw	Trigger type 000: Immediate Trigger for immediate transmission 001: Time Trigger for receive triggers 010: Single Shot Transmit Trigger for exclusive time windows 011b: Transmit Start Trigger for merged arbitrating time windows 100: Transmit Stop Trigger for merged arbitrating time windows others - no action The time of the trigger is defined by TT_TRIG. TTPTR selects the TB slot for the transmit triggers.
Bit 7: 6	Reserved	0x0	resd	Kept at default value.
Bit 5: 0	TTPTR	0x00	rw	Transmit trigger TB slot pointer If TTPTR is too big and points to a slot that is not available, then TEIF is set and no new trigger can be activated after a write access to TT_TRIG_1. If TTPTR points to an empty slot, then TEIF will be set at the moment, when the trigger time is reached.

21.7.12 CAN TTCAN: trigger time (TTTRIG)

Bit	Name	Reset value	Type	Description
Bit 31: 16	TT_WTRIG	0xFFFF	rw	Watch trigger time TT_WTRIG(15:0) defines the cycle time for a watch trigger. The initial watch trigger is the maximum cycle time 0xffff.
Bit 15: 0	TT_TRIG	0x0000	rw	Trigger time TT_TRIG(15:0) defines the cycle time for a trigger. For a transmission trigger the earliest point of transmission of the SOF of the appropriate frame will be TT_TRIG+1

Note: A write access to TT_TRIG (TTTRIG[15:0]) starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing) and activates the trigger. If the trigger is active, then a write lock for register

TTCFG[31:24] is active, and TT_TRIG bit is active. The write lock becomes inactive when the trigger time is reached (TTIF gets set is TTIE is enabled) or an error is detected (TEIF gets set).

A write access to TT_WTRIG (TTTRIG[31:16]) starts a data transfer of the trigger definition to the CAN clock domain (clock domain crossing) and activates the trigger. If the trigger is active, then a write lock for register TT_WTRIG (TTTRIG[31:16]) is active. The write access to TT_WTRIG_1 is necessary to make a new trigger active.

21.7.13 CAN acceptance filter control register (ACFCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 16	AE_x	0x01	rw	Acceptance filter enable Acceptance filter Enable 1: acceptance filter enabled 0: acceptance filter disabled Each acceptance filter (ACFC / ACFM) can be individually enabled or disabled. Only filter number 0 is enabled by default after hardware reset. Disabled filters reject a frame. Only enabled filters can accept a frame if the appropriate ACFC / ACFM configuration matches.
Bit 15: 4	Reserved	0x0	resd	Kept at default value
Bit 3: 0	ACFADR	0x00	rw	Acceptance filter address ACFADR points to a specific acceptance filter. The selected filter is accessible using the registers ACFC and ACFM. A value of ACFADR>15 is meaningless and automatically treated as value 15.

21.7.14 CAN filter code identifier ID (FCID)

Refer to Table 21-1 for filter code identifier.

Bit	Name	Reset value	Type	Description
Bit 31: 0	FCID_x	0x	rw	Filter code identifier The register configuration value is identical to the level on which the bus receives data

21.7.15 CAN filter code format (FCFMT)

Refer to Table 21-1 for FMT

Bit	Name	Reset value	Type	Description
Bit 31: 0	FCFMT_x	0x	rw	Filter code format The register configuration value is identical to the level on which the bus receives data

21.7.16 CAN filter code type (FCTYP)

Refer to Table 21-1 for TYP

Bit	Name	Reset value	Type	Description
Bit 31: 0	FCTYP_x	0x	rw	Filter code type The register configuration value is identical to the level on which the bus receives data.

21.7.17 CAN filter mask identifier (FMID)

Refer to Table 21-1 for ID

Bit	Name	Reset value	Type	Description
Bit 31: 0	FMID_x	0x	rw	Filter mask identifier 0: Acceptance identifier filtering is enabled 1: Acceptance identifier filtering is disabled

21.7.18 CAN filter mask format (FMFMT)

Refer to Table 21-1 for FMT

Bit	Name	Reset value	Type	Description
Bit 31: 0	FMFMT_x	0x	rw	Filter mask format 0: Acceptance identifier filtering is enabled 1: Acceptance identifier filtering is disabled

21.7.19 CAN filter mask type (FMTYP)

Refer to Table 21-1 for TYP

Bit	Name	Reset value	Type	Description
Bit 31: 0	FMTYP_x	0x	rw	Filter mask type 0: Acceptance identifier filtering is enabled 1: Acceptance identifier filtering is disabled

21.7.20 CAN reserve register 1 (Res 1)

Bit	Name	Reset value	Type	Description
Bit 31: 0	Reserved	0x	rw	This field is applicable to CAN XL frames only. CAN XL is not supported for this series. Software writes 0xFFFFFFFF in this bit.

21.7.21 CAN receiver buffer identifier (RBID)

Refer to Table 21-1 for ID

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBID	0x	rw	RBUF Identifier

21.7.22 CAN receive buffer format (RBFMT)

Refer to Table 21-1 for FMT

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBFMT	0x	rw	RBUF Format

21.7.23 CAN receive buffer type (RBTYP)

Refer to Table 21-1 for TYP

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBTYP	0x	rw	RBUF Type

21.7.24 CAN receive buffer data area (RBDATn)

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBDATn	0x	rw	RBUF Data n

21.7.25 CAN receive buffer CiA 603 acceptance time-stamp 1 (RBCIA1)

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBCIA1	0x	rw	RBUF CiA 603 acceptance time-stamp 1

21.7.26 CAN receive buffer CiA 603 acceptance time-stamp 2 (RBCIA2)

Bit	Name	Reset value	Type	Description
Bit 31: 0	RBCIA2	0x	rw	RBUF CiA 603 acceptance time-stamp 2

21.7.27 CAN receive buffer TTCAN period time (RBTTCAN)

Bit	Name	Reset value	Type	Description
Bit 15: 0	RBTTCAN	0x	rw	RBTTCAN period time

21.7.28 CAN transmit buffer identifier (TBID)

Refer to Table 21-1 for ID

Bit	Name	Reset value	Type	Description
Bit 31: 0	TBID	0x	rw	TBUF Identifier

21.7.29 CAN transmit buffer format (TBFMT)

Refer to Table 21-1 for FMT

Bit	Name	Reset value	Type	Description
Bit 31: 0	TBFMT	0x	rw	TBUF Format

21.7.30 CAN transmit buffer type (TBTYP)

Refer to Table 21-1 for TYP

Bit	Name	Reset value	Type	Description
Bit 31: 0	TBTYP	0x	rw	TBUF Type

21.7.31 CAN reserve register 2 (Res 2)

Refer to Table 21-1 for ACF

Bit	Name	Reset value	Type	Description
Bit 31: 0	Reserved	0x	rw	This field is applicable to CAN XL frames only. CAN XL is not supported for this series. Software writes 0x00000000 in this bit.

21.7.32 CAN transmit buffer data area (TBDATn)

Bit	Name	Reset value	Type	Description
Bit 31: 0	TBDATn	0x	rw	TBUF Data n

21.7.33 CAN LLC frame calculator input (LLCFORMAT)

Bit	Name	Reset value	Type	Description
Bit 31: 0	LLCFORMAT	0x00	rw	A write access to LLCFORMAT results in the automatic calculation of the size information of an LLC frame inside LLCSIZE. The layout of LLCFORMAT is identical to the layout of the format word of the LLF frame. Only bits relevant for the calculation of LLCSIZE do exist inside LLCFORMAT. Other bits are always 0. Relevant bits are: DLC, FDF, and RMF.

21.7.34 CAN LLC frame calculator output (LLCSIZE)

Bit	Name	Reset value	Type	Description
Bit 31: 16	LLCAOT	0x10	ro	LLC address offset of time-stamps LLCAOT provides the address offset of the time-stamps (value T) of an LLC frame Value T is always word-aligned. Example: For a frame with 1 byte of payload the value T will be: T=16+4=20
Bit 15: 0	LLCPBYTES	0x0	ro	LLC frame payload bytes LLCPBYTES provides the number of payload data bytes inside LLC frame

21.7.35 CAN interrupt enable register (INTEN)

Bit	Name	Reset value	Type	Description
Bit 31: 14	Reserved	0x00	resd	Kept at default value.
Bit 13	WTIEN	0x0	rw	WTIF flag ouput interrupt enable. It is disabled by default
Bit 12	TEIEN	0x0	rw	TEIF flag ouput interrupt enable. It is disabled by default
Bit 11	TTIEN	0x0	rw	TTIF flag ouput interrupt enable. It is disabled by default
Bit 10	EPIEN	0x0	rw	EPIF flag ouput interrupt enable. It is disabled by default
Bit 9	ALIEN	0x0	rw	ALIF flag ouput interrupt enable. It is disabled by default
Bit 8	BEIEN	0x0	rw	BEIF flag ouput interrupt enable. It is disabled by default
Bit 7	RIEN	0x0	rw	RIF flag ouput interrupt enable. It is disabled by default
Bit 6	ROIEN	0x0	rw	ROIF flag ouput interrupt enable. It is disabled by default
Bit 5	RFIEN	0x0	rw	RFIF flag ouput interrupt enable. It is disabled by default
Bit 4	RAFIEN	0x0	rw	RAFIF flag ouput interrupt enable. It is disabled by default
Bit 3	TPIEN	0x0	rw	TPIF flag ouput interrupt enable. It is disabled by default
Bit 2	TSIEN	0x0	rw	TSIF flag ouput interrupt enable. It is disabled by default
Bit 1	EIEN	0x0	rw	EIF flag ouput interrupt enable. It is disabled by default
Bit 0	AIEN	0x0	rw	AIF flag ouput interrupt enable. It is disabled by default

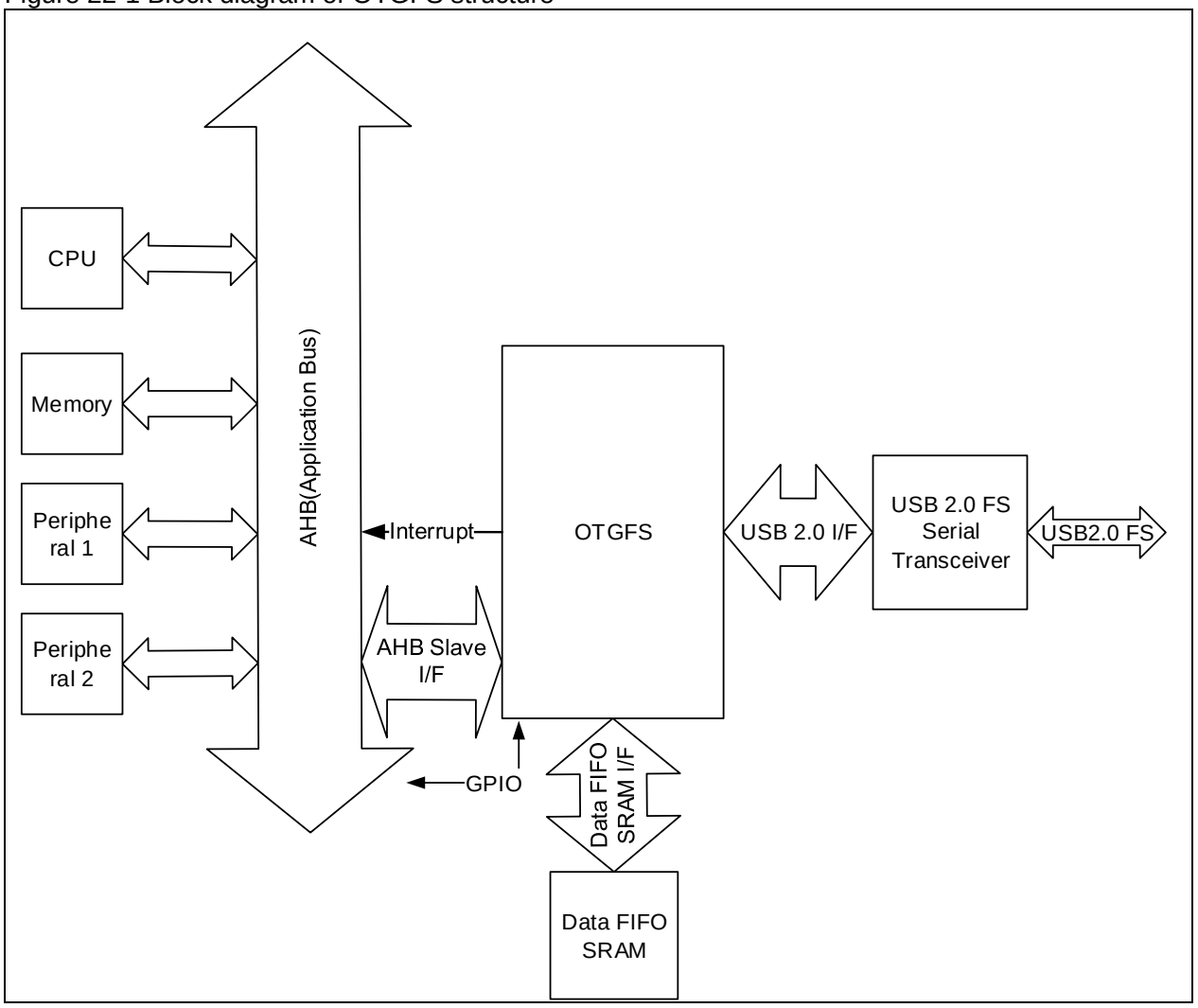
22 Universal serial bus full-seed device interface (OTGFS)

As a full-speed dual-role device, the OTGFS is fully compliant with the Universal Serial Bus Specification Revision2.0.

22.1 OTGFS structure

Figure 22-1 shows the block diagram of the OTGFS structure. The OTGFS module is connected to the AHB and has a dedicated SRAM of 1280 bytes.

Figure 22-1 Block diagram of OTGFS structure



22.2 OTGFS functional description

OTGFS supports FS (12Mb/s) and LS (1.5Mb/s) in master mode, and FS (12Mb/s) in device mode.

The OTGFS module consists of an OTGFS controller, PHY and 1280-byte SRAM.

The OTGFS supports control transfer, bulk transfer, interrupt transfer and synchronous transfer.

The OTGFS is a USB full-speed dual role device (DRD) controller. The status of the ID line determines whether the OTGFS acts as a host or device. When the ID line is floating, the OTGFS is used as a device. It is used as a host while the ID line is grounded. An internal 1.5KΩ pull-up resistor and 1.5KΩ pull-down resistor are embedded in the OTG PHY for the sake of dual role device.

In device mode, the OTGFS supports one bidirectional control endpoint, 7 IN endpoints, and 7 OUT endpoints; in hose mode, the OTGFS supports 16 host channels.

The OTGFS supports SOF and OE pulse features: a SOF pulse generates at a SOF packet, the pulse can output to pins and the timer 2; an OE pulse generates when the OTGFS outputs data and is output to pins.

As a device, a unified FIFO buffer is allocated for all OUT endpoints, and a separate FIFO buffer is provided to each of IN endpoints. As a host, a unified receive FIFO is allocated for all receive channels, a unified transmit FIFO for all non-periodic transmit channels, and a unified transmit FIFO for all periodic transmit channels.

Suspend mode is supported. If the STOPPCLK is set in the OTGFS_PCGCCTL register, the OTGFS enters Suspend mode when the bus signal is not received for 3ms in a row. The LP_MODE bit in the OTGFS_GCCFG register can be used to disable PHY receive in order to reduce power consumption.

22.3 OTGFS clock and pin configuration

22.3.1 OTGFS clock configuration

The OTGFS interface has two clock sources: 48MHz±0.25% for OTGFS controller, and AHB bus clock for reading OTGFS control register.

The USB full-speed device bus speed standard is 12Mb/s±0.25%, so it is important to supply 48MHz±0.25% for OTGFS to perform USB bus sampling.

USBFS 48M clock has two sources:

- HICK 48M

  When the HICK 48M clock is used as a USB control clock, it is recommended to enable ACC feature.
- PLLU

  PLLU must be configured as 48MHz clock output (see CRM_PLLCFG for details).

Note: The AHB clock frequency must be greater than 30MHz when OTGFS is enabled.

22.3.2 OTGFS pin configuration

The OTGFS input/output pins are multiplexed with GPIOs. The GPIOs are used as OTGFS in the following conditions:

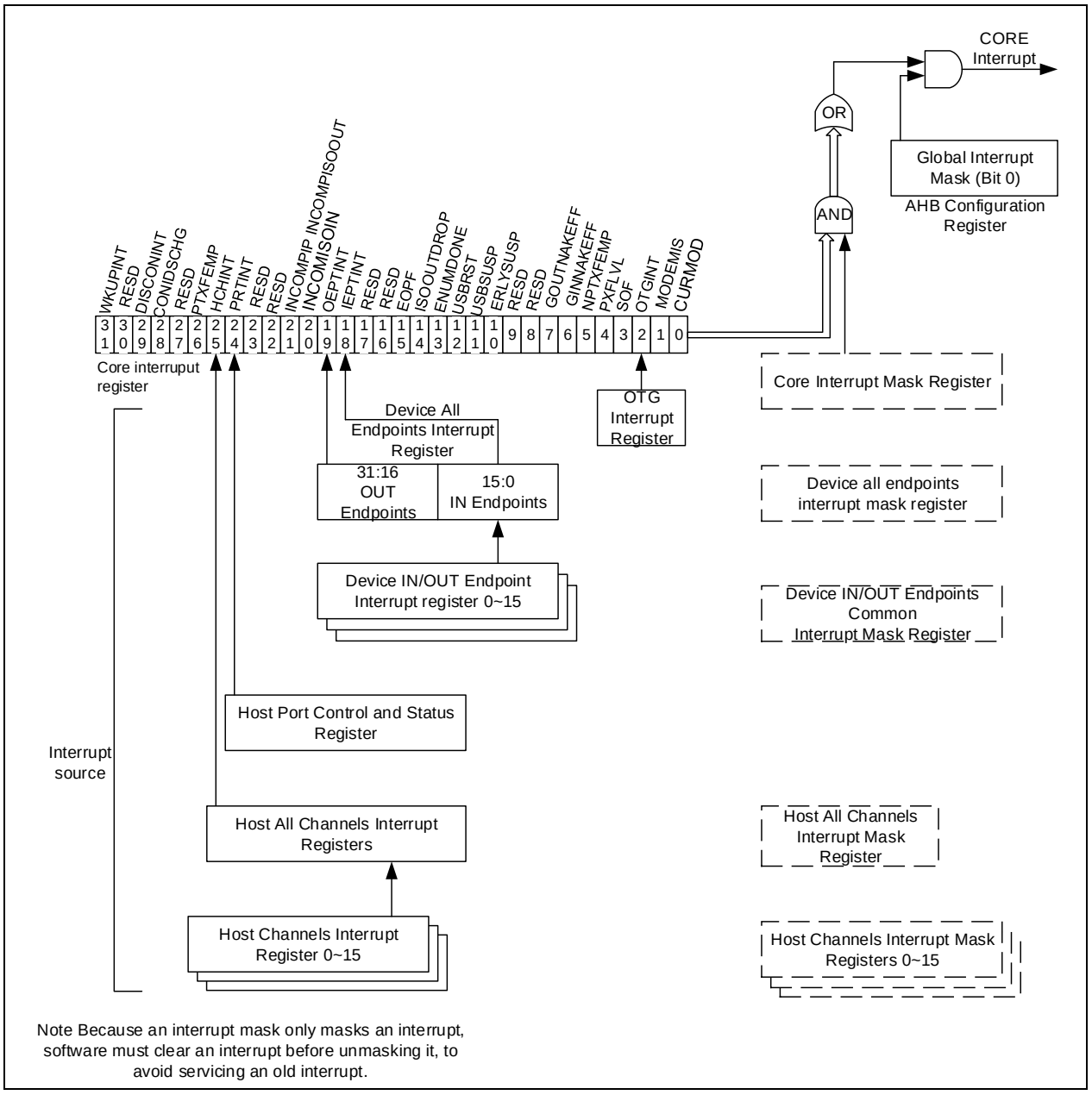
Table 22-1 OTGFS input/output pins

Pin	GPIO	Description
OTGFS_SOF	PA8	Enable OTGFS in CRM, and PA8 multiplexed function register as 0xa
OTGFS_VBUS	PA9	Enable OTGFS in CRM and PA9 multiplexed function register as 0xa
OTGFS_ID	PA10	Enable OTGFS in CRM, and PA10 multiplexed function register as 0xa
OTGFS_D-	PA11	Enable OTGFS in CRM and set PWRDOWN=1
OTGFS_D+	PA12	Enable OTGFS in CRM and set PWRDOWN=1
OTGFS_OE	PA4	Enable OTGFS in CRM and PA4 multiplexed function register as 0xd
	PA13	Enable OTGFS in CRM,and PA13 multiplexed function register as 0xa
	PC9	Enable OTGFS in CRM and PC9 multiplexed function register as 0xb

22.4 OTGFS interrupts

Figure 22-2 shows the OTGFS interrupt hierarchy diagram. Refer to the OTGFS interrupt register (OTGFS_GINTSTS) and OTGFS interrupt mask register (OTGFS_GINTMSK).

Figure 22-2 OTGFS interrupt hierarchy



22.5 OTGFS functional description

22.5.1 OTGFS initialization

If the cable is connected during power-on, the current operation mode bit (CURMOD bit) in the controller interrupt register indicates the current operating mode. The OTGFS controller enters host mode when A-type plug is connected or device mode when a B-type plug is connected.

This section describes how to initialize OTGFS controller after power-on. The application must follow the initialization sequence, however in host or device mode. All controller global registers are initialized according to the controller configuration.

1. Configure the following fields in the global AHB configuration register:
- Global interrupt mask bit = 0x1
- Non-periodic transmit FIFO empty level
- Periodic transmit FIFO empty level

2. Configure the following fields in the global interrupt mask register:
- OTGFS_GINTMSK.RXFLVLMSK = 0x0

3. Configure the following fields in the OTGFS_GUSBCFG register:
- Full-speed timeout standard bit
- USB turnaround time bit

4. The software must unmask the following bits in the OTGFS_GINTMSK register:
- OTG interrupt mask
- Mode mismatch interrupt mask

5. The software can read the CURMOD bit in the OTGFS_GINTSTS register to determine whether the OTGFS controller is operating in host or device mode.

22.5.2 OTGFS FIFO configuration

22.5.2.1 Device mode

A dynamic FIFO allocation is required during power-on or USB reset. In device mode, the application must meet the following conditions before modifying FIFO SRAM allocation.
- OTGFS_DIEPCTLx/ OTGFS_DOEPCTLx.EPENA = 0x0
- OTGFS_DIEPCTLx/ OTGFS_DOEPCTLx.NAKSTS = 0x1

The TXFNUM bit in the OTGFS_GRSTCTL register is used to refresh the controller transmit FIFO. Refer to Section Refresh controller transmit FIFO for more information.

Attention should be paid to the following information during FIFO SRAM allocation:

(1) Receive FIFO SRAM allocation
- SRAM for SETUP Packets: 13 WORDs must be reserved in the receive FIFO to receive one SETUP Packet on control endpoint. The controller does not use these locations that are reserved for SETUP packets.
- One WORD is to be reserved for global OUT NAK
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size/4) + 1 must be allocated to receive data packets. In most cases, two (largest packet size/4) + 1 spaces are recommended so that the USB can receive the subsequent packet while the previous packet is being transferred to the AHB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.
- Transfer complete status information, along with the last packet for each endpoint, is also pushed to the FIFO
- One location must be reserved for the disable status bit of each endpoint
- Typically, two WORDs for each OUT endpoint are recommended.

(2) Transmit FIFO SRAM allocation

The minimum SRAM space required for each IN endpoint transmit FIFO is the maximum data packet size for that particular IN endpoint. The more the space allocated to the transmit IN endpoint FIFO, the better the USB performance, and this helps to avoid latency on the AHB line.

Table 22-2 OTGFS transmit FIFO SRAM allocation

FIFO name	SRAM size
Receive FIFO	rx_fifo_size, including setup packets, OUT endpoint control information and OUT data packets.
Transmit FIFO 0	tx_fifo_size[0]
Transmit FIFO 1	tx_fifo_size[1]
Transmit FIFO 2	tx_fifo_size[2]
……	……
Transmit FIFO i	tx_fifo_size[i]

Configure the following registers according to the above mentioned:

1. OTGFS receive FIFO size register (OTGFS_GRXFSIZ)
- OTGFS_GRXFSIZ.RXFDEP = rx_fifo_size

2. Endpoint 0 TX FIFO size register (OTGFS_DIEPTXF0)
- OTGFS_DIEPTXF0.INEPT0TXDEP = tx_fifo_size[0]
- OTGFS_DIEPTXF0.INEPT0TXSTADDR = rx_fifo_size

3. Device IN endpoint transmit FIFO#1 size register (OTGFS_DIEPTXF1)
- OTGFS_DIEPTXF1.INEPTXFSTADDR = OTGFS_DIEPTXF0.INEPT0TXSTADDR + tx_fifo_size[0]

4. Device IN endpoint transmit FIFO#2 size register (OTGFS_DIEPTXF2)

- OTGFS_DIEPTXF2.INEPTXFSTADDR = OTGFS_DIEPTXF1.INEPTXFSTADDR + tx_fifo_size[1]

5. Device IN endpoint transmit FIFO#i size register (OTGFS_DIEPTXFi)

- OTGFS_DIEPTXFi.INEPTXFSTADDR = OTGFS_DIEPTXFi-1.INEPTXFSTADDR +tx_fifo_size[i-1]

6. After SRAM allocation, refresh transmit FIFO and receive FIFO to ensure normal FIFO running.

- OTGFS_GRSTCTL.TXFNUM = 0x10
- OTGFS_GRSTCTL.TXFFLSH = 0x1
- OTGFS_GRSTCTL.RXFFLSH = 0x1

The application cannot perform other operations on the controller until the TXFFLSH and RXFFLSH bits are cleared.

22.5.2.2 Host mode

In host mode, the application must confirm the following status before changing FIFO SRAM allocation:

- All channels have been disabled
- All FIFOs are empty

After FIFO SRAM allocation is complete, the application must refresh all FIFOs in the controller through the TXFNUM bit in the OTGFS_GRSTCTL register.

After allocation, the FIFO pointers must be reset by refreshing operation to ensure normal FIFO running. Refer to Section Refresh controller transmit FIFO for more information.

(1) Receive FIFO SRAM allocation

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size/4) + 2 must be allocated to receive data packets. If more synchronous endpoints are enabled, then at least two (largest packet size/4) + 2 spaces must be allocated to receive back-to-back packets. In most cases, two (largest packet size/4) + 2 spaces are recommended so that the USB can receive the subsequent packet while the previous packet is being transferred to the AHB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.

Transfer complete status information and channel abort information, along with the last packet in the host channel is also pushed to the FIFO. Thus, two WORDs must be allocated for this.

(2) Transmit FIFO SRAM allocation

The minimum SRAM space required for the host non-periodic transmit FIFO is the largest packet size of all non-periodic OUT channels. The more the space allocated to the non-periodic FIFO, the better the USB performance, and this helps to avoid latency on the AHB line. Typically, two largest packet sizes of space is recommended so that the AHB can get the next data packet while the current packet is being transferred to the USB. If there is a longer latency on AHB, sufficient spaces must be reserved to receive multiple packets in order to prevent synchronous data packet loss.

(3) Internal storage space allocation

Table 22-3 OTGFS internal storage space allocation

FIFO Name	Data SRAM Size
Receive FIFO	rx_fifo_size
Non-periodic transmit FIFO	tx_fifo_size[0]
Periodic transmit FIFO	tx_fifo_size[1]

Configure the following registers according to the above mentioned:

1. OTGFS receive FIFO size register (OTGFS_GRXFSIZ)

- OTGFS_GRXFSIZ.RXFDEP = rx_fifo_size

2. OTGFS Non-periodic TX FIFO size register (OTGFS_GNPTXFSIZ)

- OTGFS_GNPTXFSIZ.NPTXFDEP = tx_fifo_size[0]
- OTGFS_GNPTXFSIZ. NPTXFSTADDR = rx_fifo_size

3. OTGFS host periodic transmit FIFO size register (OTGFS_HPTXFSIZ)

- OTGFS_HPTXFSIZ.PTXFSIZE = tx_fifo_size[1]
- OTGFS_HPTXFSIZ.PTXFSTADDR = OTGFS_GNPTXFSIZ.NPTXFSTADDR + tx_fifo_size[0]

4. After SRAM allocation, refresh transmit FIFO and receive FIFO to ensure normal FIFO running.

- OTGFS_GRSTCTL.TXFNUM = 0x10
- OTGFS_GRSTCTL.TXFFLSH = 0x1
- OTGFS_GRSTCTL.RXFFLSH = 0x1
- The application cannot perform other operations on the controller until the TXFFLSH and RXFFLSH bits are cleared.

22.5.2.3 Refresh controller transmit FIFO

The application refreshes all transmit FIFOs through the TXFFLSH bit in the OTGFS_GRSTCTL register:

- Check whether GINNAKEFF=0 or not in the OTGFS_GINTSTS register. If this bit has been cleared, write 0x1 to the OTGFS_DCTL.SGNPINNAK register. When the NACK valid interrupt is set, it means that the controller does not read FIFO.
- Wait until GINNAKEFF = 0x1 in the OTGFS_GINTSTS register, indicating that the NAK configuration has taken effect for all IN endpoints.
- Poll the OTGFS_GRSTCTL register and wait until AHBIDLE=1. AHBIDLE = H indicates that the controller does not write the FIFO.
- Confirm whether TXFFLSH = 0x0 or not in the OTGFS_GRSTCTL register. If TXFFLSH is cleared, write the transmit FIFO number to be refreshed into the OTGFS_GRSTCTL.TXFNUM register.
- Set TXFFLSH = 0x1 in the OTGFS_GRSTCTL register, and wait until it is cleared.
- Set the CGNPINNAK bit in the OTGFS_DCTL register.

22.5.3 OTGFS host mode

22.5.3.1 Host initialization

The following steps must be respected to initialize the controller:

1. Unmask interrupt through the PRTINTMSK bit in the OTGFS_GINTMSK register
2. Program the OTGFS_HCFG register
3. Set PRTPWR = 0x1 in the OTGFS_HPRT register to drive VBUS supply on the USB
4. Wait until that the PRTCONDETbit is set in the OTGFS_HPRT0 register, indicating that the device is connected to the port
5. Set PRTRST = 0x1 in the OTGFS_HPRT register to issue a reset operation
6. Wait for at least 10 ms to ensure the completion of the reset
7. Set PRTRST = 0x0 in the OTGFS_HPRT register
8. Wait for the interrupt (PRTENCHNG bit in the OTGFS_HPRT register)
9. Read the PRTSPD bit in the OTGFS_HPRT register to get the enumeration speed
10. Configure the HFIR register according to the selected PHY clock value
11. Select the size of the receive FIFO by setting the OTGFS_GRXFSIZ register
12. Select the start address and size of the non-periodic transmit FIFO by setting the OTGFS_GNPTXFSIZ register
13. Select the start address and size of the periodic transmit FIFO by setting the OTGFS_HPTXFSIZ register

To communicate with the device, the application must enable and initialize at least one channel according to OTGFS channel initialization requirements.

22.5.3.2 OTGFS channel initialization

To communicate with the device, the application must enable and initialize at least one channel according to the following steps:

1. Unmask the following interrupts by setting the OTGFS_GINTMSK register:

- Non-periodic transmit FIFO empty for OUT transfers
- Non-periodic transmit FIFO half empty for OUT transfers

2. Unmask the interrupts of the selected channels by setting the OTGFS_HAINTMSK register
3. Unmask the transfer-related interrupts in the host channel interrupt register by setting the OTGFS_HCINTMSKx register
4. Configure the total transfer size (in bytes), and the expected number of the packets (including short packets) for the OTGFS_HCTSIZx register of the selected channel. The application must configure the PID bit according to the initial data PID (it is the PID on the first OUT transfer, or to be received from the first IN transfer)
5. Configure the transfer size to ensure that the transfer size of the channel is a multiple of the largest packet size
6. Configure the OTGFS_HCCHARx register of the selected channel according to the device endpoint characteristics such as type, speed and direction (the channel cannot be enabled by setting the enable bit until the application is ready for packet transfer or reception)

22.5.3.3 Halting a channel

The application can disable a channel by writing 0x1 to the CHDIS and CHENA bits in the OTGFS_HCCHARx register. This enables the host to refresh the summited requests (if any) and generates a channel halted interrupt. The application cannot re-allocate channels for other transactions until an interrupt is generated in the OTGFS_HCINTx register (CHHLTD bit). Those transactions that have already been started on the USB line are not interrupted by the host.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-period channel) or the periodic request queue (when disabling a periodic channel). The application can refresh the submitted requests when the request queue is full (before disabling the channel) by setting CHDIS=0x1, and CHENA=0 in the OTGFS_HCCHARx register.

When there is a transaction input in the request queue, the controller will trigger a RXFLVL interrupt. The application must generate a channel halted interrupt through the OTGFS_GRXSTSP register.

The application is expected to abort a channel on any of the following conditions:

- When an interrupt (XFERC bit) is received in the OTGFS_HCINTx register during a non-periodic IN transfer
- When an STALL , XACTERR , BBLERR or DTGLERR interrupt in the OTGFS_HCINTx register is received for an IN or OUT channel
- When a DISCONINT (device disconnected) interrupt event is received in the OTGFS_GINTSTS register, the application must check the PRTCONSTS bit in the OTGFS_HPRT register. This is because when the device is disconnected with the host, the PRTCONSTS bit will be reset in the OTGFS_HPRT register. The application must initiate a software reset to ensure that all channels have been cleared. Once the device is reconnected, the host must start a USB reset.
- When the application needs to abort a transfer before normal completion

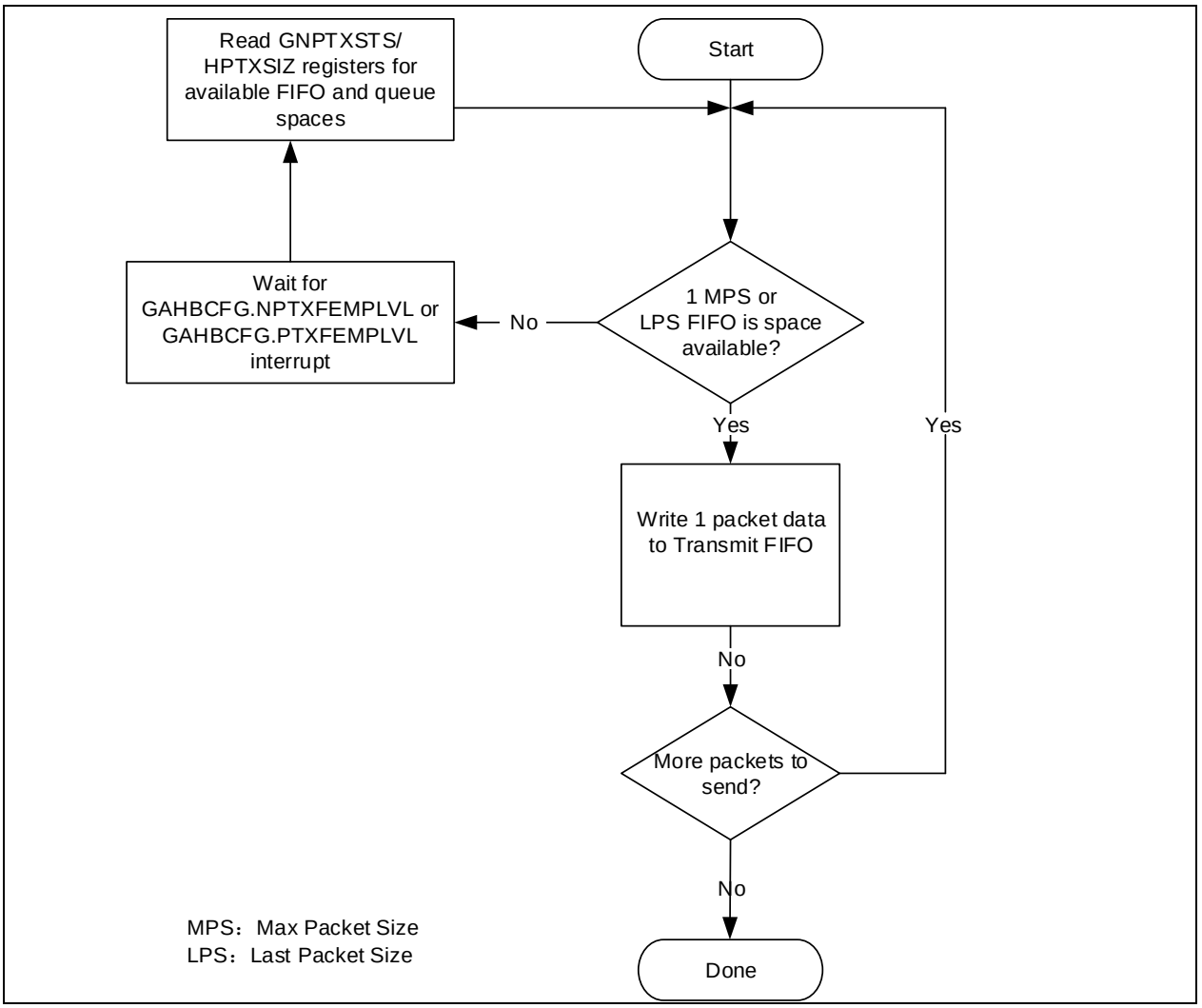
22.5.3.4 Queue depth

Up to 8 interrupt and synchronous transfer requests are supported in the periodic hardware transfer request queue; while up to 8 control and bulk transfer requests are allowed in the non-periodic hardware transfer request queue.

- Writing the transmit FIFO

Figure 22-3 shows the flow chart of writing the transmit FIFO. The OTGFS host automatically writes a request (OUT request) to the periodic/non-periodic request queue when writing the last one WORD packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in WORDs. If the packet size is not aligned with WORD, the application must use padding. The OTGFS host determines the actual packet size according to the programmed maximum packet size and transfer size.

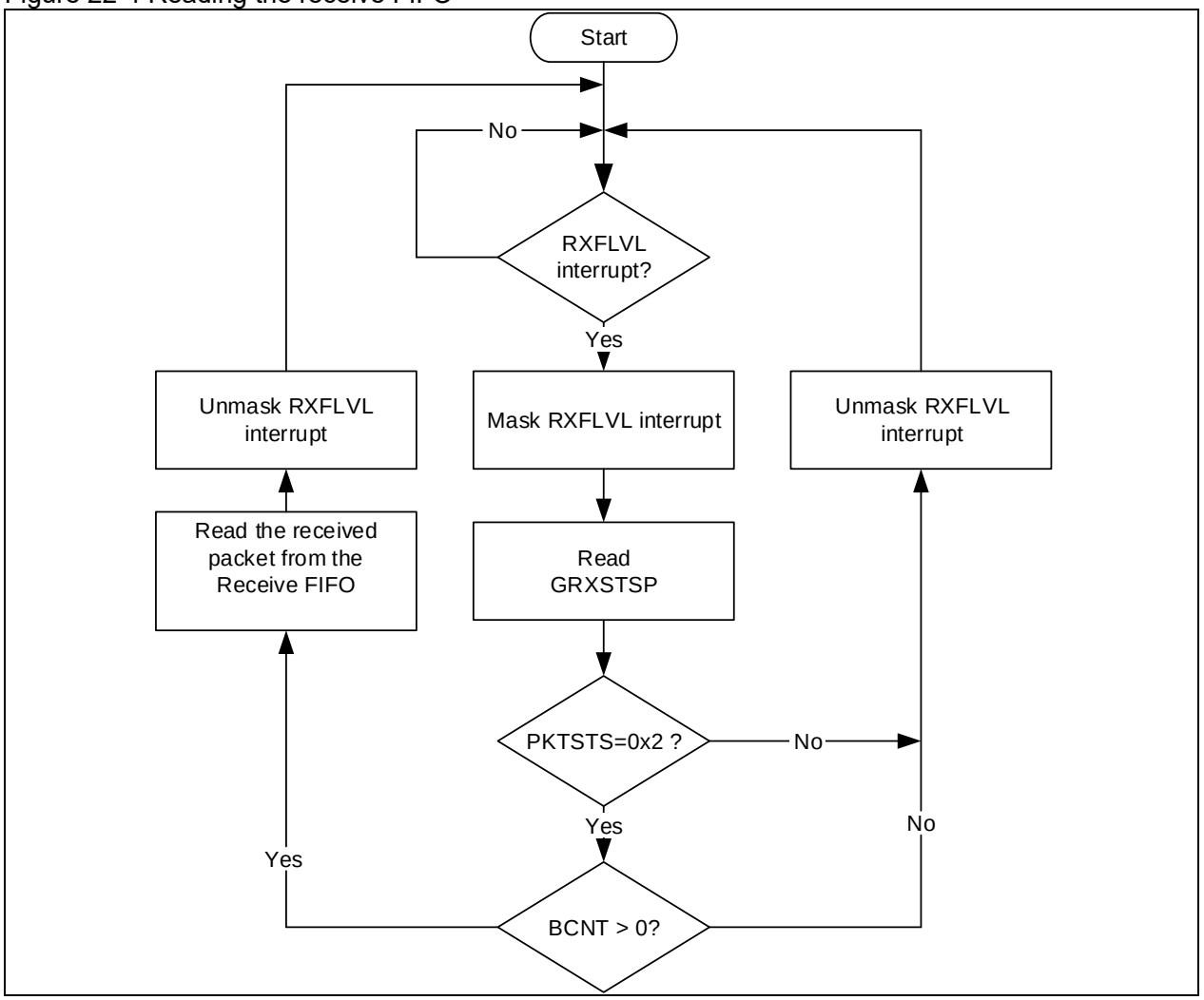
Figure 22-3 Writing the transmit FIFO



- Reading the receive FIFO

Figure 22-4 shows the flow chart of reading the receive FIFO. The application must ignore all packet statuses other than IN data packet (0x0010)

Figure 22-4 Reading the receive FIFO



22.5.3.5 Special cases

(1) Handling babble conditions

The OTGFS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more than the largest packet size for the channel. Port babble occurs if the controller continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF)

When the OTGFS controller detects a packet babble, it stops writing data to the receiver buffer and waits for the completion of packet. When it detects the end of packet, the OTGFS flushes the data already written in the receiver buffer and generates a babble interrupt.

When the OTGFS controller detects a port babble, it flushes the receive FIFO and disables the port. Then the controller generates a Port disable interrupt. Once receiving the interrupt, the application must determine that this is not caused by an overcurrent condition (another cause of the port disable interrupt) by checking the PRTOVRCACT bit in the OTGFS_HPRT register, then perform a software reset. The controller does not send any more tokens if a port babble signal is detected.

(2) Handling device disconnected conditions

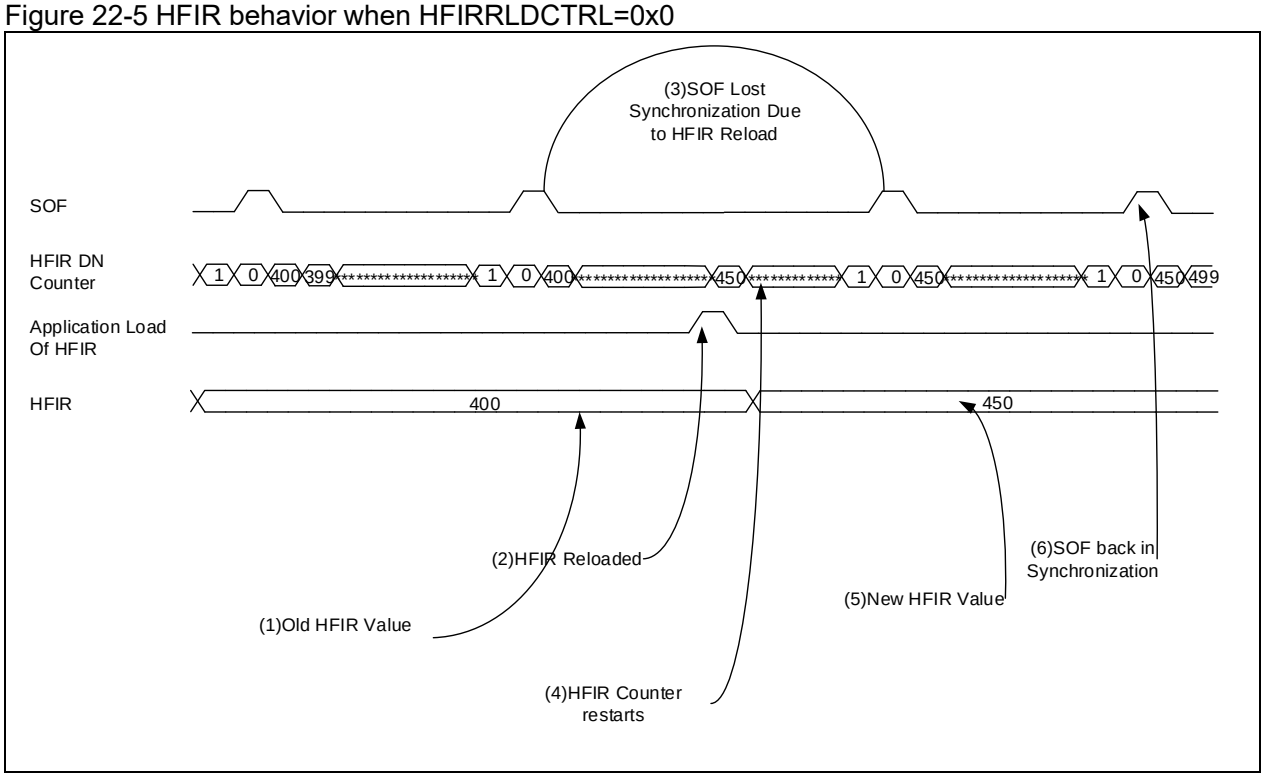
If the device is suddenly disconnected, an interrupt is generated on a disconnect event (DISCONINT bit in the OTGFS_GINTSTS register). Upon receiving this interrupt, the application must start a software reset through the CSFTRST in the OTGFS_GRSTCTL register.

22.5.3.6 Host HFIR feature

The host frame interval register (HFIR) defines the interval between two consecutive SOFs (full-speed) or Keep-Alive tokens. This field contains the number of PHY clock for the required frame interval. This is mainly used to adjust the SOF duration based on PHY clock frequencies.

Figure 22-5 shows the HFIR behavior when the HFIRRLDCTRL is set to 0x0 in the OTGFS_HFIR register.

Figure 22-5 HFIR behavior when HFIRRLDCTRL=0x0

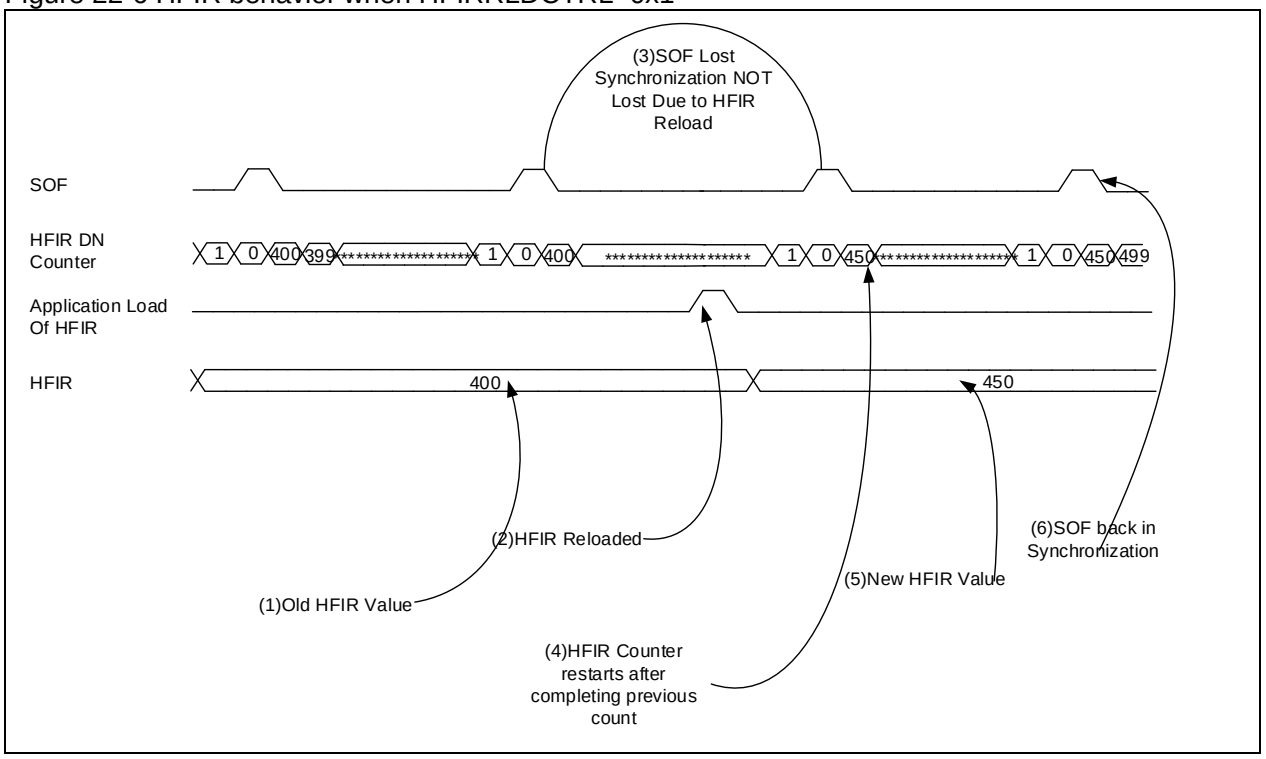


The sequence of operation is as follows:

1. After power-on reset, the current HFIR value set by the application is shown
2. The application loads a new value into the HFIR register
3. The HFIR downcounter is reloaded, so it will immediately restart counting to cause SOF synchronization loss
4. Restart HFIR counter
5. The HFIR register receives a new programmed value
6. Obtain SOF synchronization again after the first SOF is generated using the HFIR new feature

Figure 22-6 shows the HFIR behavior when HFIRRLDCTRL=0x1 in the OTGFS_HFIR register.

Figure 22-6 HFIR behavior when HFIRRLDCTRL=0x1



The sequence of operation is as follows:

1. After power-on reset, the current HFIR value set by the application is shown
2. The application loads a new HFIR value; the HFIR counter does not apply this new value, but continues counting until it reaches 0
3. The counter generates a SOF when it reaches 0 using the old HRIF value
4. The HFIR counter applies a new value
5. New HFIR value takes effect

The SOF synchronization resumes after going through above-mentioned steps.

22.5.3.7 Initialize bulk and control IN transfers

Figure 22-7 shows a typical bulk or control IN transfer operation. Refer to channel 2 (ch_2) for more information. The assumptions are as follows:

- The application is attempting to receive two largest-packet-size packets (transfer size is 1024 bytes)
- The receive FIFO contains at least one largest-packet-size packet and two status WORDs per each packet (72 bytes for full-speed transfer)
- The non-periodic request queue depth is 4

(1) Operation process for common bulk and control IN transfers

The sequence of operations shown in Figure 22-7 is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements)
2. Set the CHENA bit in the OTGFS_HCCHAR2 register to write an IN request to the non-periodic request queue
3. The controller issues an IN token after completing the current OUT transfer
4. The controller generates a RXFLVL interrupt as soon as the receive packet is written into the receive FIFO
5. To handle the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, and then read the receive FIFO. Following this step to unmask the RXFLVL interrupt
6. The controller generates the RXFLVL interrupt when the transfer complete status is written into the receive FIFO

7. The application must read the receive packet status, and ignore it when the receive packet status is not an IN data packet
8. The controller generates the XFERC interrupt as soon as the receive packet is read
9. To handle the XFERC interrupt, disable the channel (see Halting a channel) and stop writing the OTGFS_HCCHAR2 register. The controller writes a channel halted request to the non-periodic request queue once the OTGFS_HCCHAR2 register is written
10. The controller generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO
11. Read and ignore the receive packet status
12. The controller generates a CHHLTD interrupt as soon as the halt status is read from the receive FIFO
13. In response to the CHHLTD interrupt, the processor does not allocate the channel for other transfers.

(2) Handling interrupts

The following code describes the interrupt service routine related to the channel during bulk and control IN transfers

```
Unmask (XACTERR/XFERC/BBLERR/STALL/DATATGLERR)
if (XFERC)
    {
    Reset Error Count
    Unmask CHHLTD
    Disable Channel
    Reset Error Count
    Mask ACK
    }
else if (XACTERR or BBLERR or STALL)
    {
    Unmask CHHLTD
    Disable Channel
    if (XACTERR)
        {
        Increment Error Count
        Unmask ACK
        }
    }
else if (ChHltd)
    {
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
    else
        {
        Re-initialize Channel
        }
    }
else if (ACK)
    {
    Reset Error Count
    Mask ACK
```

```
    }
else if (DATATGLERR)
    {
    Reset Error Count
}
```

22.5.3.8 Initialize bulk and control OUT/SETUP transfers

Figure 22-7 shows a typical bulk or control transfer OUT/SETUP transfer operation. Refer to channel 1 (ch_1) for more information. It is necessary to send two bulk transfer OUT packets. The control transfer SETUP operation is the same, just the fact that it has only one packet. The assumptions are as follows:

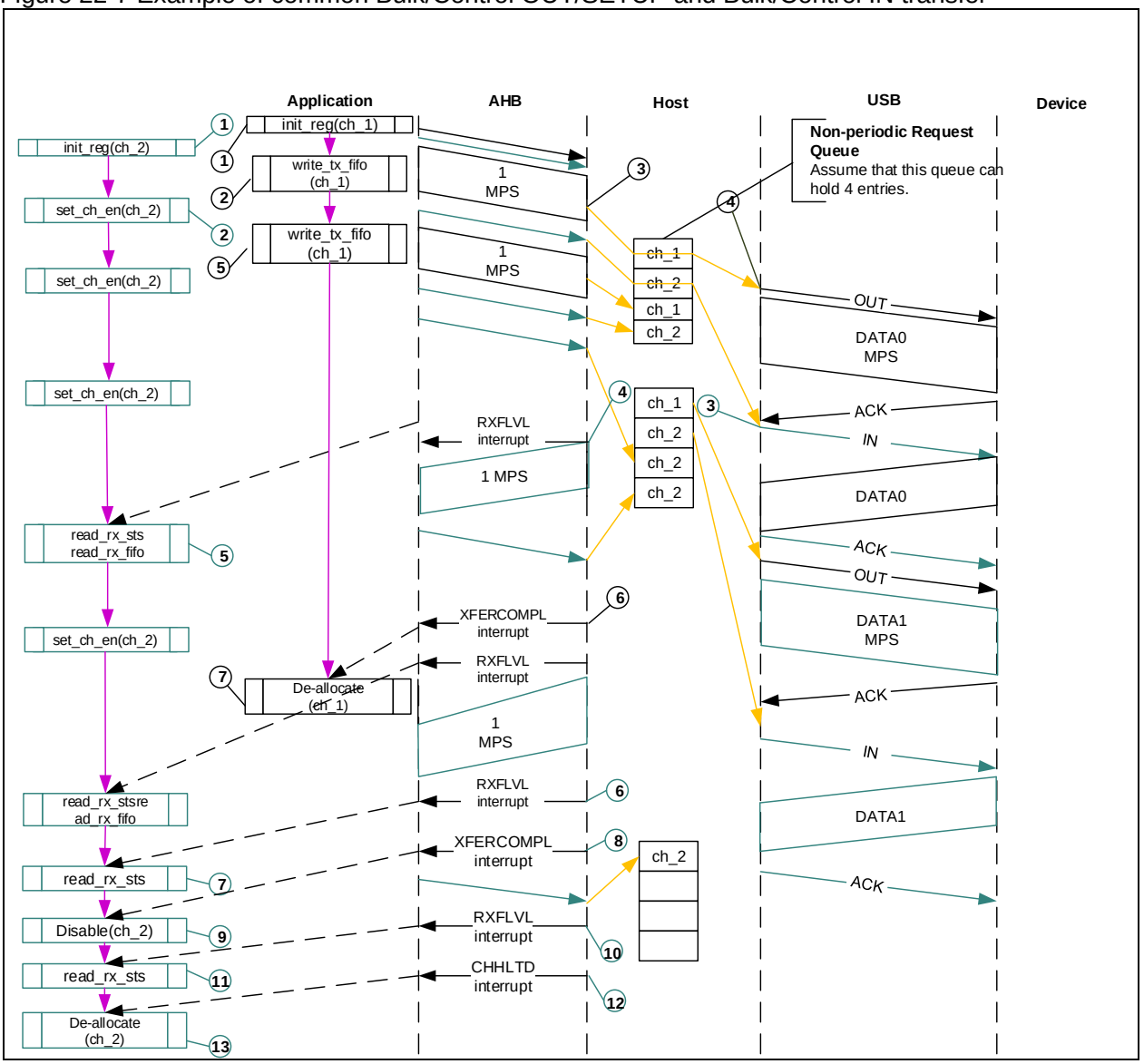
- The application is attempting to send two largest-packet-size packets (transfer size is 1024 bytes)
- The non-periodic transmit FIFO can store two packets (128 bytes for full-speed transfer)
- The non-periodic request queue depth is 4

(1) OUT/SETUP operation process for common bulk and control transfer

The sequence of operations shown in Figure 22-7 is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements)
2. Write the first packet for channel 1
3. Along with the last WORD write, the controller writes a request to the non-periodic request queue
4. The controller sends an OUT token in the current frame as soon as the non-periodic queue becomes empty
5. Write the second packet (the last one) to the channel 1
6. The controller generate an XFERC interrupt as soon as the previous transfer is completed successfully
7. In response to the XFERC interrupt, the processor does not allocate the channel for other transfers.

Figure 22-7 Example of common Bulk/Control OUT/SETUP and Bulk/Control IN transfer



(2) Handling interrupts

The following code describes the interrupt service routine related to the channel during bulk and control transfer OUT/SETUP operation.

```
Unmask (NAK/XACTERR/NYET/STALL/XFERC)
if (XFERC)
    {
    Reset Error Count
    Mask ACK
    De-allocate Channel
    }
else if (STALL)
    {
    Transfer Done = 1
    Unmask CHHLTD
    Disable Channel
    }
else if (NAK or XACTERR or NYET)
    {
    Rewind Buffer Pointers
```

```
    Unmask CHHLTD
    Disable Channel
    if (XactErr)
        {
        Increment Error Count
        Unmask ACK
        }
    else
        {
        Reset Error Count
        }
    }
else if (CHHLTD)
    {
    Mask CHHLTD
    if (Transfer Done or (Error_count == 3))
        {
        De-allocate Channel
        }
    else
        {
        Re-initialize Channel (Do ping protocol for HS)
        }
    }
else if (ACK)
    {
    Reset Error Count
    Mask ACK
}
```

Notes:

- The application can only write the transmit FIFO when the transmit FIFO and request queue has free spaces. The application must check whether there is a free space in the transmit FIFO through the NPTXFEMP bit in the OTGFS_GINTSTS register.
- The application can only write a request when the request queue has free spaces and wait until an XFERC interrupt is received.

22.5.3.9 Initialize interrupt IN transfers

Figure 22-8 shows the operation process of a typical interrupt IN transfer. Refer to channel 2 (ch_2). The assumptions are as follows:

- The application is attempting to receive one largest-packet-size packet (transfer size is 1024 bytes) from an odd frame
- The receive FIFO can store at least one largest-packet-size packet and two status WORDs per packet (1031 bytes for full-speed transfer)
- The periodic request queue depth is 4

(1) Common interrupt IN operation process

The sequence of operations shown in Figure 22-8 (channel 2) is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS_HCCHAR2 register
2. Set the CHENA bit in the OTGFS_HCCHAR2 register to write an IN request to the periodic request queue
3. The OTGFS host writes an IN request to the periodic request queue each time the CHENA is set in the OTGFS_HCCHAR2 register

4. The OTGFS host attempts to send an IN token in the next frame (odd)
5. The OTGFS host generates a RXFLVL interrupt as soon as an IN packet is received and written to the receive FIFO
6. To handle the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask the interrupt after reading the entire packet
7. The controller generates the RXFLVL interrupt when the transfer complete status is written to the receive FIFO. The application must read and ignore the receive packet when the receive packet is not an IN packet
8. The controller generates an XFERC interrupt as soon as the receive packet is read
9. To handle the XFERC interrupt, read the PKTCN bit in the OTGFS_HCTSIZ2 register. If the PKTCNT bit in the OTGFS_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer. If PKTCNT == 0 in the OTGFS_HCTSIZ2 register, re-initialize the channel for the next transfer. In this case, the application must reset the ODDFRM bit in the OTGFS_HCCHAR2 register.

(2) Handling interrupts

The following code describes the interrupt service routine related to the channel during interrupt IN transfer.

```
Unmask (NAK/XACTERR/XFERC/BBLERR/STALL/FRMOVRUN/DATATGLERR)
if (XFERC)
  {
  Reset Error Count
  Mask ACK
  if (HCTSIZx.PKTCNT == 0)
    {
    De-allocate Channel
    }
  else
    {
    Transfer Done = 1
    Unmask CHHLTD
    Disable Channel
    }
  }
else if (STALL or FRMOVRUN or NAK or DATATGLERR or BBLERR)
  {
  Mask ACK
  Unmask CHHLTD
  Disable Channel
  if (STALL or BBLERR)
    {
    Reset Error Count
    Transfer Done = 1
    }
  else if (!FRMOVRUN)
    {
    Reset Error Count
    }
  }
else if (XACTERR)
  {
```

```
  Increment Error Count
  Unmask ACK
  Unmask CHHLTD
  Disable Channel
  }
else if (CHHLTD)
  {
  Mask CHHLTD
  if (Transfer Done or (Error_count == 3))
    {
    De-allocate Channel
    }
  else Re-initialize Channel (in next b_interval - 1 uF/F)
    }
  }
else if (ACK)
  {
  Reset Error Count
  Mask ACK
}
```

The application can only write a request to the same channel when the remaining space in the request queue reaches the number defined in the MC field, before switching to other channels (if any).

22.5.3.10 Initialize interrupt OUT transfers

Figure 22-8 shows a typical interrupt OUT transfer operation. Refer to channel 1 (ch_1). The assumptions are as follows:

- The application is attempting to send one largest-packet-size packet (transfer size is 1024 bytes) to every frame.
- The periodic transmit FIFO can store one packet (1KB bytes for full-speed transfer).
- The periodic request queue depth is 4.

(1) Common interrupt IN operation process

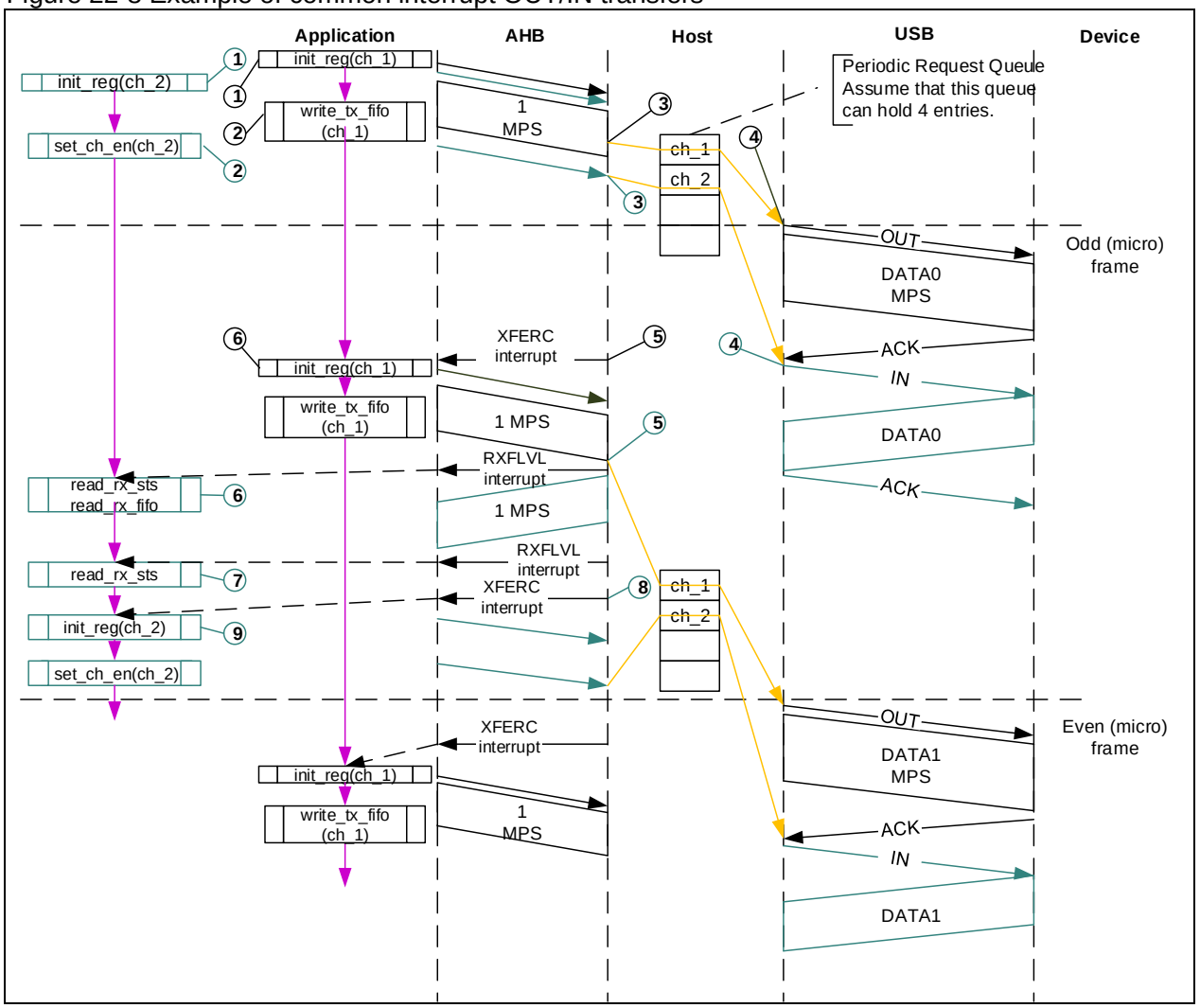
The sequence of operations shown in Figure 22-8 (channel 1) is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS_HCCHAR2 register.
2. Write the first packet to the channel 1.
3. Along with the last WORD write of each packet, the host writes a request to the periodic request queue.
4. The host sends an OUT token in the next frame (odd).
5. The host generates an XFERC interrupt after the last packet is transmitted successfully.
6. In response to the XFERC interrupt, re-initialize the channel for the next transfer.

(2) Handling interrupts

Figure 22-8 shows an example of common interrupt OUT/IN transfers

Figure 22-8 Example of common interrupt OUT/IN transfers



The following code describes the interrupt service routine related to the channel during interrupt OUT transfers.

```
Unmask (NAK/XACTERR/STALL/XFERC/FRMOVRUN)
if (XFERC)
{
Reset Error Count
Mask ACK
De-allocate Channel
}
else if (STALL or FRMOVRUN)
{
Mask ACK
Unmask CHHLTD
Disable Channel
if (STALL)
{
Transfer Done = 1
}
}
else if (NAK or XACTERR)
{
Rewind Buffer Pointers
```

```
Reset Error Count
Mask ACK
Unmask CHHLTD
Disable Channel
}
else if (CHHLTD)
{
Mask CHHLTD
if (Transfer Done or (Error_count == 3))
{
De-allocate Channel
}
else
{
Re-initialize Channel (in next b_interval - 1 uF/F)
}
}
else if (ACK)
{
Reset Error Count
Mask ACK
}
```

Before switching to other channels (if any), the application can only write packets based on the number defined in the MC filed to the transmit FIFO and request queue when the transmit FIFO has free spaces. The application can determine whether the transmit FIFO has free spaces through the NPTXFEMP bit in the OTGFS_GINTSTS register.

22.5.3.11 Initialize synchronous IN transfers

Figure 22-9 shows the operation process of a typical synchronous IN transfer. Refer to channel 2 (ch_2). The assumptions are as follows:

- The application is attempting to receive one largest-packet-size packet (transfer size is 1024 bytes) from the next odd frame
- The receive FIFO can store at least one largest-packet-size packet and two status WORDs per packet (1031 bytes for full-speed transfer)
- The periodic request queue depth is 4

(1) Common interrupt IN operation process

The sequence of operations shown in Figure 22-9 (channel 2) is as follows:

1. Initialize channel 2 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS_HCCHAR2 register
2. Set the CHENA bit in the OTGFS_HCCHAR2 register to write an IN request to the periodic request queue
3. The OTGFS host writes an IN request to the periodic request queue each time the CHENA is set in the OTGFS_HCCHAR2 register
4. The OTGFS host attempts to send an IN token in the next frame (odd)
5. The OTGFS host generates a RXFLVL interrupt as soon as an IN packet is received and written to the receive FIFO
6. To handle the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask the interrupt after reading the entire packet
7. The controller generates the RXFLVL interrupt when the transfer complete status is written to the receive FIFO. The application must read and ignore the receive packet when the receive packet is

not an IN packet (GRXSTSR.PKTSTS!= 0x0010)

8. The controller generates an XFERC interrupt as soon as the receive packet is read
9. To handle the XFERC interrupt, read the PKTCN bit in the OTGFS_HCTSIZ2 register. If the PKTCNT bit in the OTGFS_HCTSIZ2 is not equal to 0, disable the channel before re-initializing the channel for the next transfer. If PKTCNT == 0 in the OTGFS_HCTSIZ2 register, re-initialize the channel for the next transfer. In this case, the application must reset the ODDFRM bit in the OTGFS_HCCHAR2 register.

(2) Handling interrupts

The following code describes the interrupt service routine related to the channel during synchronous IN transfers.

```
Unmask (XACTERR/XFERC/FRMOVRUN/BBLERR)
if (XFERC or FRMOVRUN)
  {
  if (XFERC and (HCTSIZx.PKTCNT == 0))
    {
    Reset Error Count
    De-allocate Channel
    }
  else
    {
    Unmask CHHLTD
    Disable Channel
    }
  }
else if (XACTERR or BBLERR)
  {
  Increment Error Count
  Unmask CHHLTD
  Disable Channel
  }
else if (CHHLTD)
  {
  Mask CHHLTD
  if (Transfer Done or (Error_count == 3))
    {
    De-allocate Channel
    }
  else
    {
    Re-initialize Channel
    }
}
```

22.5.3.12 Initialize synchronous OUT transfers

Figure 22-9 shows a typical synchronous OUT transfer operation. Refer to channel 1 (ch_1). The assumptions are as follows:

- The application is attempting to send one largest-packet-size packet (transfer size is 1024 bytes) to every frame from the next odd frame
- The periodic transmit FIFO can store one packet (1KB bytes for full-speed transfer)
- The periodic request queue depth is 4

(1) Common interrupt IN operation process

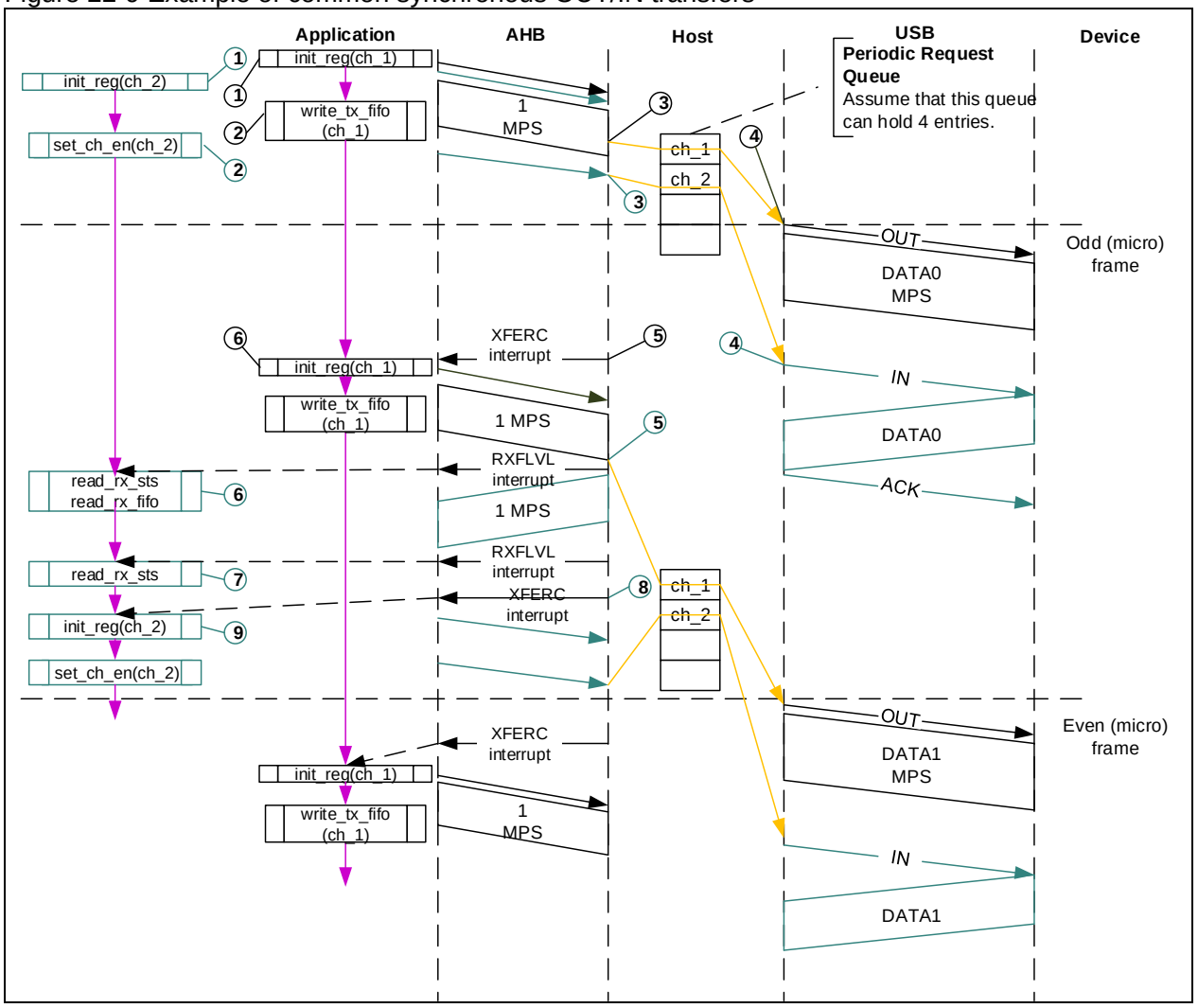
The sequence of operations shown in Figure 22-9 (channel 2) is as follows:

1. Initialize channel 1 (according to OTGFS channel initialization requirements). The application must set the ODDFRM bit in the OTGFS_HCCHAR2 register
2. Write the first packet to the channel 1
3. Along with the last WORD write of each packet, the host writes a request to the periodic request queue
4. The OTGFS host sends an OUT token in the next frame (odd)
5. The host generates an XFERC interrupt after the last packet is transmitted successfully
6. In response to the XFERC interrupt, re-initialize the channel for the next transfer.

(2) Handling interrupts

Figure 22-9 shows an example of common synchronous OUT transfers

Figure 22-9 Example of common synchronous OUT/IN transfers



The following code describes the interrupt service routine related to the channel during synchronous OUT transfers.

```
Unmask (FRMOVRUN/XFERC)
if (XFERC)
  {
  De-allocate Channel
  }
else if (FRMOVRUN)
  {
  Unmask CHHLTD
  Disable Channel
```

```
  }
else if (CHHLTD)
  {
  Mask CHHLTD
  De-allocate Channel
}
```

22.5.4 OTGFS device mode

22.5.4.1 Device initialization

The application must perform the following steps to initialize the controller during power-on or after switching a mode from host to device:

1. Program the following fields in the OTGFS_DCFG register
- Device speed
- Non-zero-length status OUT handshake
- Periodic frame interval

2. Clear the SFTDISCON bit in the OTGFS_DCTL register. The controller will start connection after clearing this bit

3. Program the OTGFS_GINTMSK register to unmask the following interrupts:
- USB reset
- Enumeration done
- Early suspend
- USB suspend
- SOF

4. Wait for the USBRESET interrupt in the OTGFS_GINTSTS register. It indicates that a reset signal has been detected on the USB (lasting for about 10ms). Upon receiving this interrupt, the application must follow the steps defined in USB initialization on USB reset.

5. Wait for the ENUMDONE interrupt in the OTGFS_GINTSTS register. It indicates the end of USB reset. Upon receiving this interrupt, the application must read the OTGFS_DSTS register to determine the enumeration speed and perform the steps defined in Endpoint initialization on enumeration completion. At this time, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

22.5.4.2 Endpoint initialization on USB reset

This section describes the operations required for the application to perform when a USB reset signal is detected:

1. Set the NAB bit for all OUT endpoints
- OTGFS_DOEPCTLx.SNAK = 0x1(for all OUT endpoints)

2. Unmask the following interrupt bits
- OTGFS_DAINTMSK.INEP0 = 0x1(control IN endpoint 0)
- OTGFS_DAINTMSK.OUTEP0 = 0x1(control OUT endpoint 0)
- OTGFS_DOEPMSK.SETUP = 0x1
- OTGFS_DOEPMSK.XFERC = 0x1
- OTGFS_DIEPMSK.XFERC = 0x1
- OTGFS_DIEPMSK.TIMEOUT = 0x1

3. To receive/transmit data, the device must perform Device initialization steps to initialize registers

4. Allocate SRAM for each endpoint
- Program the OTGFS_GRXFSIZ register to be able to receive control OUT data and SETUP data. If the allocated SRAM is equal to at least 1 largest-packet-size of control endpoint 0 + 2 WORDs (for the status of the control OUT data packet) +10 WORDs (for setup packets)
- Program the OTGFS_DIEPTXF0 register to be able to transmit control IN data. The allocated SRAM is equal to at least 1 largest-packet-size of control endpoint 0

5. Reset the device address in the device configuration register

6. Program the following fields in the endpoint-specific registers to ensure that control OUT endpoint 0 is able to receive a SETUP packet

- OTGFS_DOEPTSIZ0.SUPCNT = 0x3(to receive up to 3 consecutive SETUP packets)

At this point, all initialization required to receive SETUP packets is done.

22.5.4.3 Endpoint initialization on enumeration completion

This section describes the operations required for the application to perform when an enumeration completion interrupt signal is detected:

- Upon detecting the enumeration completion interrupt signal, read the OTGFS_DSTS register to get the enumeration speed.
- Program the MPS bit in the OTGFS_DIEPCTL0 register to set the maximum packet size. This operation is used to configure control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
- Unmask SOF interrupts.

At this point, the device is ready to receive SOF packets and has been configured to perform control transfers on control endpoint 0.

22.5.4.4 Endpoint initialization on SetAddress command

This section describes the operations required for the application to perform when the application receives a SetAddress command in a SETUP packet

- Program the OTGFS_DCFG register with the device address received in the SetAddress command.
- Program the controller to send an IN packet.

22.5.4.5 Endpoint initialization on SetConfiguration/SetInterface command

This section describes the operations required for the application to perform when the application receives a SetConfiguration / SetInterface command in a SETUP packet

- When a SetConfiguration command is received, the application must program the endpoint registers according to the characteristics of the valid endpoints defined in the new configuration
- When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command
- Some endpoints that were valid in the previous configuration are not valid in the new configuration. These invalid endpoints must be disabled
- Refer to Endpoint activation and USB endpoint deactivation for more information on how to activate or disable a certain endpoint
- Unmask the interrupt for each valid endpoint and mask the interrupts for all invalid endpoints in the DAINTMSK register
- Refer to OTGFS FIFO configuration for more information on how to program SRAM for each FIFO
- After all required endpoints are configured, the application must program the controller to send a status IN packet

At this point, the device controller has been ready to receive and transmit any type of data packet.

22.5.4.6 Endpoint activation

This section describes how to activate a device endpoint or configure an existing device endpoint to a new type.

1.Program the following bits in the OTGFS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTGFS_DOEPCTLx register (for OUT or bidirectional endpoints)

- Largest packet size
- USB valid endpoint = 0x1
- Endpoint start data toggle (for interrupt and bulk endpoints
- Endpoint type
- Transmit FIFO number

2. Once the endpoint is activated, the controller starts decoding the tokens issued to this endpoint and sends out a valid handshake for each valid token received for the endpoint

22.5.4.7 USB endpoint deactivation

This section describes how to deactivate an existing endpoint. Disable the suspended transfer before performing endpoint deactivation.

- Clear the USB valid endpoint bit in the OTGFS_DIEPCTLx register (for IN or bidirectional endpoints) or the OTGFS_DOEPCTLx register (for OUT or bidirectional endpoints)
- Once the endpoint is deactivated, the controller will ignore the tokens issued to this endpoint, which causes a USB timeout.

22.5.4.8 Control write transfers (SETUP/Data OUT/Status IN)

This section describes the steps required for control write transfers.

The application programming process is as follows:

1. When the SETUP bit is set in the OTGFS_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS_DOEPTSIZx register to receive the subsequent SETUP packet

2. If the last SETUP packet received before the generation of the SETUP interrupt indicates data OUT stage, program the controller to perform OUT transfers based on Asynchronous OUT data transfer operation

3. The application can receive up to 64-byte data for a single OUT data transfer of control endpoint 0. If the application expects to receive more than 64-byte data during data OUT stage, it must re-enable the endpoint to receive another 64-byte data, and it must continue this operation until the completion of all data reception in data stage

4. When the XFERC interrupt is set in the OTGFS_DOEPINTx register during the last OUT transfer, it indicates the end of data OUT stage of control transfer

5. Once the completion of data OUT stage, the application must perform the following steps:

- If the application needs to transfer a new SETUP packet, it must re-enable control OUT endpoints (refer to OUT data transfers)

  OTGFS_DOEPCTLx.EPENA = 0x1

- To execute the received SETUP commands, the application must configure the corresponding registers in the controller. This is optional, depending on the received SETUP command type

6. During status IN stage, the application must follow the requirements of Non-periodic (for bulk and control) IN data transfers to program registers to perform data IN transfers

7. When the XFERC interrupt is set in the OTGFS_DOEPINTx register is set, it indicates that the status stage of control transfers is started. As soon as Data transfer complete mode and Status stage start bit are set in the receive FIFO packet status register, the controller generates an interrupt. The Transfer complete interrupt can be cleared through the XFERC bit in the OTGFS_DOEPINTx register

Repeat above-mentioned steps until an interrupt (XFERC bit in the OTGFS_DIEPINTx register) is generated on the endpoint, which indicates the end of control write transfers.

22.5.4.9 Control read transfers (SETUP/Data IN/Status OUT)

This section describes the steps required for control read transfers.

The application programming process is as follows:

- When the SETUP bit is set in the OTGFS_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS_DOEPTSIZx register to receive the subsequent SETUP packet
- If the last SETUP packet received before the generation of the SETUP interrupt indicates data IN stage, program the controller to perform IN transfers based on Non-periodic IN data transfer operation

- The application can receive up to 64-byte data for a single IN data transfer of control endpoint 0. If the application expects to receive more than 64-byte data during data IN stage, it must re-enable the endpoint to receive another 64-byte data, and it must continue this operation until the completion of all data transfers in data stage
- Repeat above-mentioned steps until the XFERC interrupt is generated in the OTGFS_DIEPINTx register for each IN transfer on the endpoint
- When the XFERC interrupt is set in the OTGFS_DOEPINTx register during the last IN transfer, it indicates the end of data OUT stage of control transfer
- To execute data OUT transfer at status OUT stage, the application must configure the controller. This is optional.

The application must program the NZSTSOUTHSHK bit in the OTGFS_DCFG register, and then send data OUT transfer at status stage

The XFERC interrupt bit is set in the OTGFS_DOEPINTx register to indicate the end of status OUT stage of control transfer, marking the completion of control read transfers.

22.5.4.10 Control transfers (SETUP/Status IN)

This section describes the two-phase control transfer operation.

The application programming process is as follows:

1. When the SETUP bit is set in the OTGFS_DOEPINTx register, it indicates that a valid SETUP packet has been sent to the application, and data stage is initiated, see OUT data transfers. At the end of the SETUP stage, the application must rewrite 3 to the SUPCNT bit in the OTGFS_DOEPTSIZx register to receive the subsequent SETUP packet
2. The application decodes the last SETUP packet received before the generation of the SETUP interrupt. If the SETUP packet indicates two-level control commands, the application must perform the following steps:

- Set OTGFS_DOEPCTLx.EPENA = 0x1
- The application must program the registers in the controller to perform the received SETUP commands

3. For status IN stage, the application must program the registers based on Non-periodic (bulk and control) IN data transfers to perform data IN transfers

4. The XFERC interrupt bit is set in the OTGFS_DIEPINTx register to indicate the end of status IN stage of control transfers.

22.5.4.11 Read FIFO packets

This section describes how to read FIFO packets (OUT data and SETUP packets)

1. The application must read the OTGFS_GRXSTSP register as soon as the RXFLVL interrupt bit is detected in the OTGFS_GINTSTS register
2. The application can mask the RXFLVL interrupt bit in the OTGFS_GINTSTS register by setting RXFLVL = 0x0 in the OTGFS_GINTMSK register, until it has read the data packets from the receive FIFO
3. If the received packet byte is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is read from the receive data FIFO

4. The receive FIFO packet status reading indicates one of the following conditions:

5. Global OUT NAK mode: PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = Don’t Care (0x0) and DPID = Don’t Care (0x00), indicating that the global OUT NAK bit has taken effect

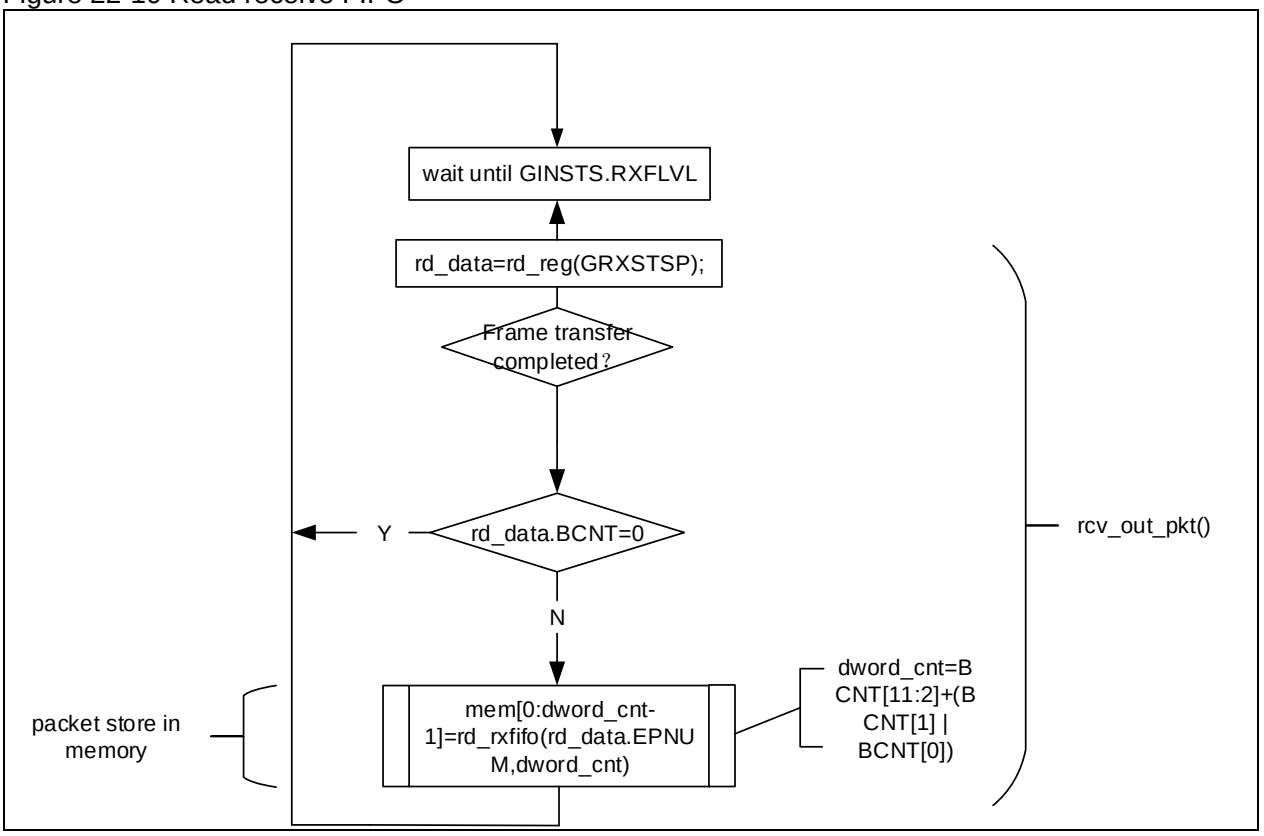
- SETUP packet mode: PKTSTS = SETUP, BCnt = 0x008, EPNUM = Control EP Num and DPID = D0, indicating that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO
- Setup stage done mode: PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num and DPID = Don’t Care (0x00), indicating the completion of the Setup stage for the specified

endpoint, and the start of the data stage. After this request is popped from the receive FIFO, the controller triggers a Setup interrupt on the specified control OUT endpoint

- Data OUT packet mode: PKTSTS = DataOUT, BCnt =size of the received data OUT packet (0 ≤ BCNT ≤ 1024), EPNUM =Endpoint number on which the data packet was received, DPID =Actual data PID
- Data transfer complete mode: PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM =OUT endpoint number on which the data transfer is complete, DPID = Don't Care (0x00). These data indicate that an OUT data transfer for the specified OUT endpoint has been complete. After this request is popped from the receive FIFO, the controller triggers a Transfer Completed interrupt on the specified OUT endpoint. PKTSTS code can be found in the OTGFS_GRXSTSR / OTGFS_GRXSTSP register

7. After the valid data is popped from the receive FIFO, the RXFLVL interrupt bit in the OTGFS_GINTSTS register must be unmasked
8. Step 1-5 must be repeated each time the application detects the interrupt line due to the RXFLVL bit in the OTGFS_GINTSTS register. Reading an empty receive FIFO will result in unexpected behavior. Figure 22-10 shows a flowchart.

Figure 22-10 Read receive FIFO



22.5.4.12 OUT data transfers

This section describes the internal data flow during data OUT and SETUP transfers, and how the application handles SETUP transfers.

(1) Setup transfers

This section describes how to handle SETUP data packets and the application's operating sequence of handling SETUP transfers. After power-on reset, the application must follow the OTGFS Initialization process to initialize the controller. Before communicating with the host, the application must initialize the endpoints based on Device Initialization, and refer to Read FIFO packets for more information.

[Application requirements]

1. To receive a SETUP packet, the SUPCNT bit (OTGFS_DOEPTSIZx) on a control OUT endpoint must be programmed to be a non-zero value. When the application sets the SUPCNT bit to a non-zero value, the controller receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status bit and EPENA bit in the OTGFS_DOEPCTLx register. The SUPCNT bit is decremented each time the control endpoint receives a SETUP packet. If the SUPCNT bit is not programmed to a proper value before receiving a SETUP packet, the controller still receives the SETUP packet and decrements the SUPCNT bit, but the application may not be able to determine the exact number of SETUP packets received in the SETUP stage of a control transfer.

- OTGFS_DOEPTSIZx.SUPCNT = 0x3

2. The application must allocate some extra space for the receive data FIFO to ensure that up to three SETUP packets can be received on a control endpoint

- The space to be reserved is 13 WORDs. Four WORDs are required for one SETUP packet, one WORD is required for the Setup stage and 8 WORDs are required to store two extra SETUP packets among all control endpoints
- Four WORDs per SETUP packet are required to store 8-byte SETUP data and 4-byte Transfer completed status and 4-byte SETUP status (SETUP packet mode). The controller must reserve this space to receive data
- FIFO is used to write SETUP data only, and never for data packets

3. The application must read 2-WORDs SETUP packet from the receive data

4. The application must read and discard the Transfer Completed status WORD from the receive FIFO, and ignore the Transfer Completed interrupt due to this read operation.

[Internal data flow]

1. When a SETUP packet is received, the controller writes the received data to the receive FIFO, without checking whether there is available space in the receive FIFO, irrespective of the NAK and Stall bits on the control endpoints.

- The controller sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.

2. For every SETUP packet received on the USB line, 3 WORDs of data are written to the receive FIFO, and the SUPCNT bit is decremented by 1 automatically.

- The first WORD contains control information used internally by the controller
- The second WORD contains the first 4 bytes of the SETUP command
- The third WORD contains the last 4 bytes of the SETUP command

3. When the SETUP stage switches to data IN/OUT stage, the controller writes a SETUP status done WORD to the receive FIFO, indicating the end of the SETUP stage.

4. The application reads the SETUP packages through the AHB bus.

5. When the application pops the Setup stage done WORD from the receive FIFO, the controller interrupts the application through the SETUP interrupt bit in the OTGFS_DOEPINTx register, indicating that the application can start processing the SETUP packet received.

6. The controller clears the endpoint enable bit for control OUT endpoints.

[Application programming process]

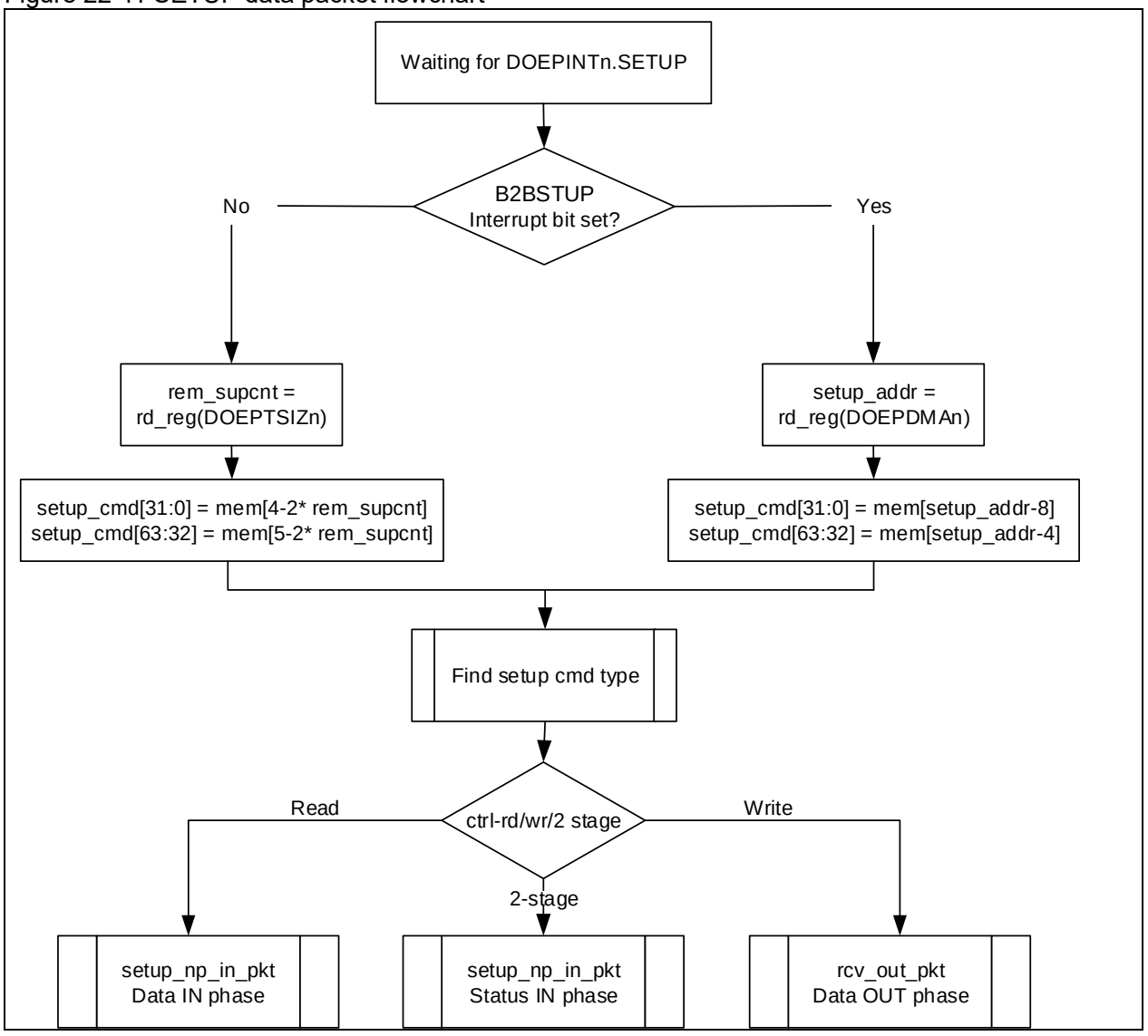
1. Program the OTGFS_DOEPTSIZx register

- OTGFS_DOEPTSIZx.SUPCNT = 0x3

2. Wait for the RXFLVL interrupt bit in the OTGFS_GINTSTS register and read and empty the data packets from the receive FIFO (Refer to Read FIFO packets for details). This operation can be repeated several times.

3. When the SETUP interrupt bit is set in the OTGFS_DOEPINTx register, it indicates that the SETUP data transfer has been completed successfully. Upon this interrupt, the application must read the OTGFS_DOEPTSIZx register to determine the number of SETUP packets received, and process the last received SETUP packet.

Figure 22-11 SETUP data packet flowchart



(2) Handling more than three consecutive SETUP packets

Per the USB 2.0 specification, typically, a host does not send more than three consecutive SETUP packets to the same endpoint during a SETUP packet error. However, the USB2.0 specification does not limit the number of consecutive SETUP packets a host can send to the same endpoint. If this condition occurs, the OTGFS controller generates an interrupt (B2BSTUP bit in the OTGFS_DOEPINTx register).

22.5.4.13 IN data transfers

This section describes the internal data flow during IN data transfers and how the application handles IN data transfers.

1. The application can either select a polling or an interrupt mode.

- In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTGFS_DTXFSTSx register to determine whether there is enough space in the data FIFO.
- In interrupt mode, the application must wait for the TXFEMP interrupt bit in the OTGFS_DIEPINTx register, and then read the OTGFS_DTXFSTSx register to determine whether there is enough space in the data FIFO.
- To write a single non-zero-length data packet, there must be enough space to write the entire data packet in the data FIFO.
- To write zero-length data packet, the application does not need to check the FIFO space.

2. Either way, when the application determines that there is enough space to write a transmit packet, the application can first write into the endpoint control register before writing the data into the data FIFO. Normally, except for setting the endpoint enable bit, the application must do a read modify write on the OTGFS_DIEPCTLx register to avoid modifying the contents of the register. If the space is enough, the application can write multiple data packets for the same endpoint into the transmit FIFO. For the periodic IN endpoints, the application must write packets for only one frame. It can write packets for the next periodic transfer only after the previous transfer has been completed.

22.5.4.14 Non-periodic (bulk and control) IN data transfers

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Device Initialization to initialize endpoints.

[Application requirements]

1. For IN transfers, the Transfer Size bit in the Endpoint Transfer Size register indicates a payload that contains multiple largest-packet-size packets and a short packet. This short packet is transmitted at the end of the transfer.

- To transmit several largest-packet-size packets and a short packet:

  Transfer size [epnum] = n * mps[epnum] + sp (n is an integer ≥ 0 and 0 ≤ sp < mps[epnum])

  If (sp > 0), then packet count [epnum] = n + 1. Otherwise, packet count [epnum] = n
- To transmit a single zero-length data packet:

  Transfer size [epnum] = 0x0

  Packet count [epnum] = 0x1
- To transmit several largest-packet-size packets and a zero-length data packet (at the end of the transfer), the application must split the transfer into two parts. First send the largest-packet-size packets and then the zero-length data packet alone.

  First transfer: Transfer size [epnum] = n * mps[epnum]; Packet count = n;

  Second transfer: Transfer size [epnum] = 0x0; Packet count = 0x1;

2. If an endpoint is enabled for data transfers, the controller updates the Transfer size register. At the end of the IN transfer (indicated by endpoint disable interrupt bit), the application must read the Transfer size register to determine how much data in the transmit FIFO have already been sent on the USB line.

3. Data fetched in the transmit FIFO = Application-programmed initial transfer size – Controller-updated final transfer size

- Data transmitted on USB = (Application-programmed initial packet count – Controller-updated final packet count) * mps[epnum]
- Data to be transmitted on USB = Application-programmed initial transfer size – Data transmitted on USB

[Internal data flow]

1. The application must set the transfer size and packet count bits in the endpoint control registers and enable the endpoint to transmit the data.

2. The application must also write the required data to the transmit FIFO of the endpoint.

3. Each time a data packet is sent to the transmit FIFO by the application the transfer size for this endpoint is decremented with the packet size. The application must continue to write data until the transfer size of the endpoint becomes 0. After writing data to the FIFO, the “packet count in the FIFO” is

incremented (this is a 3-bit count for each IN endpoint transmit FIFO data packet, which is internally maintained by the controller. For an IN endpoint FIFO, the maximum number of packets maintained by the controller at any time is 8). For non-zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.

4. After the data is written to the transmit FIFO, the controller reads them upon receiving an IN token. For each non-synchronous IN data packet transmitted with an ACK handshake signal, the number of packets for the endpoint is decremented by 1, until the packet count becomes 0. The packet count is not decremented due to a timeout.

5. For zero-length data packets (indicated by an internal zero-length flag), the controller sends zero-length packets according to the IN token, and the packet count is decremented automatically.

6. If there are no data in the FIFO on a received IN token and the packet count for the endpoint is 0, the controller generates an “IN token received when FIFO is empty” interrupt, and the NAK bit for the endpoint is not set. The controller responds with a NAK handshake signal to the non-synchronous endpoints on the USB.

7. The controller rewinds the FIFO pointers internally and no timeout interrupt is generated except for the control IN endpoints.

8. When the transfer size is 0 and the packet count is also 0, the Transfer completed interrupt is generated and the endpoint enable bit is cleared.

[Application programming sequence]

1. Program the OTGFS_DIEPTSIZx register according to the transfer size and the corresponding packet count.

2. Program the OTGFS_DIEPCTLx register according to the endpoint characteristics and set the CNAK and endpoint enable bits.

3. While sending non-zero-length data packets, the application must poll the OTGFS_DTXFSTSx register (where n is the FIFO number related to that endpoint) to determine whether there is enough space in the data FIFO. The application can also use the TXFEMP bit in the OTGFS_DIEPINTx register before writing data.

22.5.4.15 Non-synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in “OTGFS Initialization”. Before communicating with a host, the application must initialize endpoints based on the process described in “Endpoint Initialization” and by referring to “Read FIFO packets”. This section describes a regular non-synchronous OUT transfers (control, bulk or interrupt transfers).

[Application requirements]

1. For OUT data transfers, the transfer size of the endpoint transfer register must be set to a multiple of the largest packet size for the endpoint, and adjusted to the WORD boundary.

```
if (mps[epnum] mod 4) == 0
transfer size[epnum] = n * (mps[epnum]) //WORD Aligned
else
transfer size[epnum] = n * (mps[epnum] + 4 - (mps[epnum] mod 4)) //Non WORD
Aligned
packet count[epnum] = n
n > 0
```

2. When an OUT endpoint interrupt occurs, the application must read the endpoint’s transfer size register to calculate the size of the data in the memory. The received payload size must be less than the programmed transfer size.

- Payload size in memory = Application-programmed initial transfer size – Controller-updated final transfer size
- Number of USB packets the payload was received = Application-programmed initial packet count – Controller-updated final packet count

[Internal data flow]

1. The application must set the transfer size and packet count bits in the endpoint control registers, clear the NAK bit, and enable the endpoint to receive the data.

2. Once the NAK bit is cleared, the controller starts receiving data and writes it to the receive FIFO as long as there is available space in the receive FIFO. For each data packet received on the USB line, the data packet and its status are written to the receive FIFO. The packet count is decremented by 1 each time a packet (largest packet size or a short packet) is written to the receive FIFO.

- OUT data packets received with Bad Data CRC are emptied from the receive FIFO
- After sending an ACK to the data packet on the USB, the controller discards non-synchronous OUT data packets that the host (which cannot detect the ACK) re-transmits. The application does not detect multiple consecutive OUT data packets on the same endpoint with the same data PID. In this case, the packet count is not decremented.
- If there is no space in the receive FIFO, synchronous or non-synchronous data packets are ignored and not written to the receive FIFO. Besides, the non-synchronous OUT tokens receive a NAK handshake response.
- In all the above-mentioned cases, the packet count is not decremented because no data is written to the receive FIFO.

3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for the endpoint is set. Once the NAK bit is set, the synchronous or non-synchronous data packets are ignored and not written to the receive FIFO, and non-synchronous OUT tokens receive a NAK handshake response.

4. After the data is written to the receive FIFO, the application reads the data from the receive FIFO and writes it to the external memory, once packet at a time per endpoint.

5. At the end of data packet write to the external memory, the transfer size for the endpoint is decremented with the size of the written packet.

6. The OUT data transfer completed mode for an OUT endpoint is written to the receive FIFO one of the following conditions:

- The transfer size and packet count are both 0
- The last OUT data packet written to the receive FIFO is a short packet (0 ≤ data packet size < largest packet size)

7. When the application pops this entry (OUT data transfer complete), a transfer completed interrupt is generated and the endpoint enable bit is cleared.

[Application programming sequence]

1. Program the OTGFS_DOEPTSIZx register with the transfer size and the corresponding packet count.

2. Program the OTGFS_DOEPCTLx register with the endpoint characteristics, and set the endpoint enable and ClearNAK bits.

- OTGFS_DOEPCTLx.EPENA = 0x1
- OTG-DOEPCTLx.CNAK = 0x1

3. Wait for the RXFLVL interrupt in the OTGFS_GINTSTS register, and read out all data packets from the receive FIFO.

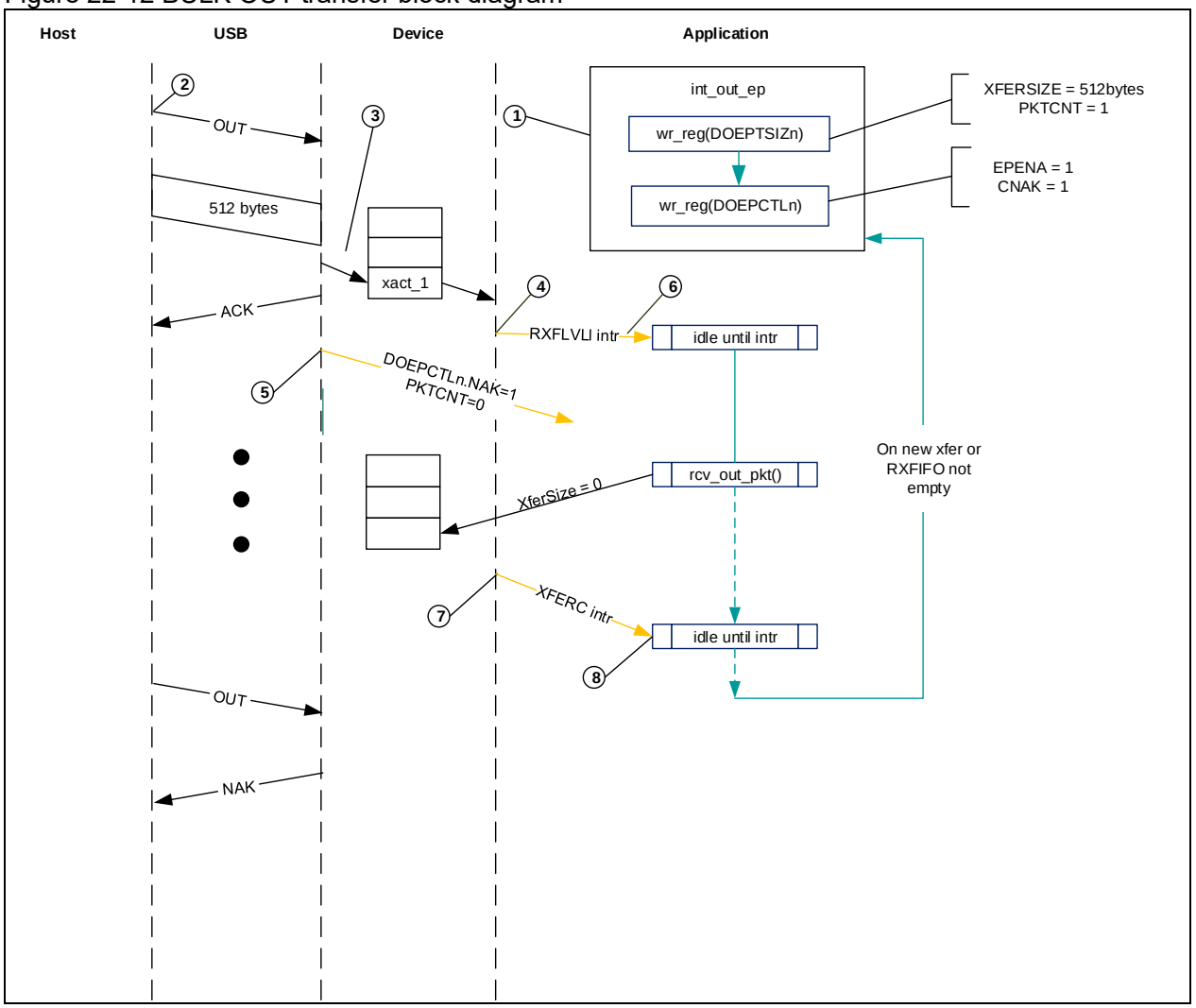
- This step can be repeated, depending on the transfer size

4. When the XFERC interrupt is set in the OTGFS_DOEPINTx register, it indicates a successful completion of the non-synchronous OUT data transfer. Read the OTGFS_DOEPTSIZx register to determine how much data has been received.

[Bulk OUT transfer]

Figure 22-12 describes the reception of a single bulk OUT data packet from the USB to the AHB and shows the events involved in the process.

Figure 22-12 BULK OUT transfer block diagram



After a SetConfiguration/SetInterface command is received, the application initializes all OUT endpoints by setting CNAK = 0x1 and EPENA = 0x1 in the OYG_DOEPCTLx register, and setting the XFERSIZE and PKTCNT bits in the OTGFS_DOEPTSIZx register.

1. The host attempts to send data (OUT token) to the endpoint

2. When the controller receives the OUT token on the USB, it stores data in the receive FIFO because the FIFO has free space.

3. Upon writing the complete data in the receive FIFO, the controller then triggers the RXFLVL interrupt bit in the OTGFS_GINTSTS register.

4. Upon receiving the packet count of USB packets, the controller internally sets the NAK bit for the endpoint to prevent it from receiving any more packets.

5. The application processes the interrupt and reads the data from the receive FIFO.

6. When the application reads all the data (equivalent to XFERSIZE), the controller generates an XFERC interrupt in the OTGFS_DOEPINTx register.

7. The application processes the interrupt and uses the XFERC bit in the OTGFS_DOEPINTx register to judge that the expected transfer is already complete.

22.5.4.16 Synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in “OTGFS Initialization”. Before communicating with a host, the application must initialize endpoints based on the process described in “Endpoint Initialization” and by referring to “Read FIFO packets”. This section describes a regular synchronous OUT transfers.

[Application requirements]

1. All the application requirements are the same as that of non-synchronous OUT data transfers.

2. For synchronous OUT data transfers, the transfer size and packet count must be set to the number of the largest-packet-size packets that can be received in a single frame and not exceed this size. Synchronous OUT data transfer cannot span more than one frame.

- 1 ≤ packet count [epnum] ≤ 3

3. If the device supports the synchronous OUT endpoints, the application must read all synchronous OUT data packets from the receive FIFO before the end of the periodic frame (EOPF interrupt in the OTGFS_GINTSTS register)

4. To receive data in the subsequent frame, a synchronous OUT endpoint must be enabled before the generation of the EOPF and SOF interrupt in the OTGFS_GINTSTS register.

[Internal data flow]

1. The internal data flow for the synchronous OUT endpoints is the same as that for the non-synchronous OUT endpoints, just for a few differences.

2. When the synchronous OUT endpoint is enabled by setting the endpoint enable bit and by clearing the NAK bit, the even/odd frame bits are also set properly. The controller can receive data on a synchronous OUT endpoint in a particular frame only when the following condition is met:

- Even/Odd microframe (OTGFS_DOEPCTLx) =SOFFN[0] (OTGFS_DSTS)

3. When the application completely reads the synchronous OUT data packet (data and status) from the receive FIFO, the controller updates the RXDPID bit in the OTGFS_DOEPTSIZx register based on the data PID of the last synchronous OUT data packet read from the receive FIFO.

[Application programming sequence]

1. Program the transfer size and the corresponding packet count of the OTGFS_DOEPTSIZx register

2. Program the OTGFS_DOEPCTLx register with the endpoint enable, ClearNAK and Even/Odd frame bits

- Endpoint enable = 0x1
- CNAK = 0x1
- Even/Odd frame = (0x0: Even; 0x1: Odd)

3. Wait for the RXFLVL interrupt in the OTGFS_GINTSTS register, and read all the data packets from the receive FIFO. See “Read FIFO” for more information

- This step can be repeated several times, depending on the transfer size

4. When the XFERC interrupt is set in the OTGFS_DOEPINTx register, it indicates the completion of the synchronous OUT data transfers. But this interrupt does not necessarily mean that the data in memory are good.

5. This interrupt signal cannot always be detected by the synchronous OUT data transfers. However, the application can detect the INCOMPISOOUT interrupt in the OTGFS_GINTSTS register. See “Incomplete synchronous OUT data transfers” for more information.

6. Read the OTGFS_DOEPTSIZx register to determine the received transfer size and to determine whether the data received in the frame are valid or not. The application must treat the data received in memory as valid only when one of the following conditions is met:

- OTGFS_DOEPTSIZx.RxDPID = 0xD0 and the USB packet count in which the payload was received =0x1
- OTGFS_DOEPTSIZx.RxDPID = 0xD1 and the USB packet count in which the payload was received =0x2
- OTGFS_DOEPTSIZx.RxDPID = 0xD2 and the USB packet count in which the payload was received =0x3

The number of USB packets in which the payload was received= Application-programed initial packet count – Controller-updated final packet count

The application discards invalid data packets.

22.5.4.17 Enable synchronous endpoints

After sending a Set interface control command to the device, a host enables the synchronous endpoints. Then the host can send the initial synchronous IN token in any frame before transmission in the sequence of BInterval.

Instead, synchronous support in the OTGFS controller is based on a single-transfer level. The application must re-configure the controller on every frame. The OTGFS controller enables the synchronous

endpoint of the frame before the frame to be transmitted.

For example, to send data on the frame n, enable the endpoint of the frame n-1. Additionally, the OTGFS controller schedules the synchronous transfers by setting Even/Odd frame bits.

[Synchronous IN transfer interrupt]

The following interrupts must be processed to ensure successful scheduling of the synchronous transfers.

- XFERC interrupt in the OTGFS_DIEPINTx register (for endpoints)
- OTG INCOMPISOIN interrupt in the OTGFS_GINTSTS register (for global interrupts)

[Handling synchronous IN transfers]

The following steps must be performed to handle a synchronous IN transfer:

1. Unmask the incompISOOUT interrupt in the OTGFS_GINTSTS register by setting the INCOMISOINMSK interrupt bit in the OTGFS_GINTMSK register

2. Unmask the XFERC interrupt in the OTGFS_DIEPINTx register by setting the XFERCMSK bit in the OTGFS_DIEPMSK register

3. Enable synchronous endpoints with the following steps:

- Program the OTGFS_DIEPTSIZx register

  OTGFS_DIEPTSIZx.XFERSIZE= n * OTGFS_DIEPCTLx.MPS + sp, where 0 <= n <= 3 and 0 <= sp <OTGFS_DIEPCTLx.MPS. When the transfer size in a frame is less than that of the MPS bit in the OTGFS_DIEPCTLx register, n=0; When the transfer size in a frame is a multiple of that of the MPS bit in the OTGFS_DIEPCTLx register, sp=0.

  OTGFS_DIEPTSIZx.PKTCNT = 0x1

  The MC bit in the OTGFS_DIEPTSIZx register is set the same value as that of the PKTCNT bit in the OTGFS_DIEPTSIZx register.

- Program the OTGFS_DIEPCTLx register

  Read the OTGFS_DSTS register to determine the current frame number

  Program the OTGFS_DIEPCTLx with the maximum packet size (MPS bit)

  Set USBACTEP = 0x1 in the OTGFS_DIEPCTLx register

  Set EPTYPE = 0x1 in the OTGFS_DIEPCTLx register, marking synchronization

  Set the FIFO number of the endpoint through the TXFNUM bit in the OTGFS_DIEPCTLx register

  Set CNAK = 0x1 in the OTGFS_DIEPCTLx register

  If.SOFFN[0] = 0x0 in OTGFS_DSTS, then SETEVENFR = 0x1 in OTGFS_DIEPCTLx (otherwise, SETEVENFR = 0x1 in OTGFS_DIEPCTLx)

  If SOFFN[0] = 0x1 in OTGFS_DSTS, then SETODDFR = 0x1 in OTGFS_DIEPCTLx (otherwise, SETODDFR = 0x0 in OTGFS_DIEPCTLx)

  Set EPENA = 0x1 in OTGFS_DIEPCTLx

4. Write endpoint data to the corresponding transmit FIFO

For example, write address ranges are as follows:

- EP1 corresponding to 0x2000 - 0x2FFC
- EP2 corresponding to 0x3000 - 0x3FFC
- EP3 corresponding to 0x3000 - 0x3FFC
- ...

5. Wait for interrupts

- When an interrupt is generated (XFERC bit in OTGFS_DIEPINTx register), clear the XFERC interrupt; For the following transaction, repeat step 3-5 until the completion of data transfers.
- When an interrupt is generated (INCOMPISOIN bit in OTGFS_GINTSTS register), clear the INCOMPISOIN interrupt; For any synchronous IN endpoint, when Odd/Even bits match the current frame number bit 0, and when the endpoint remains enabled, the controller generates an interrupt at the end of the frame. This interrupt is generated on one of the following conditions:

  (1) There is no token in a frame

  (2) Late data write to the receive FIFO. An IN token has arrived before the completion of data write

  (3) IN token error

The INCOMPISOIN interrupt in the OTGFS_GINTSTS register is a global interrupt. Therefore, when more than one synchronous endpoints are in active state, the application must determine which one of the synchronous IN endpoints has not yet completed data transfers.

To achieve this, read the DSTS and DIEPCTLx bits of all synchronous endpoints. If the current endpoint has been enabled, and the read value of the SOFFN bit in the OTGFS_DSTS register is equal to the target frame number of the endpoint, it indicates that this endpoint has not finished data transfers. The application must keep track of and update the target frame number of the synchronous endpoint.

If data transfer is not yet complete on an endpoint, then Odd/Even bits have to be toggled.

Next:

(1) When the DPID is set to 1 (an odd frame) in the OTGFS_DIEPCTLx register, write 1 to the SETD0PID bit in the OTGFS_DIEPCTLx register makes it an even frame, then data transmission starts when there is an IN token input in the next frame.

(2) When the DPID is set to 0 in the OTGFS_DIEPCTLx register, write 1 to the SETD1PID bit in the OTGFS_DIEPCTLx register makes it an odd frame, then data transmission starts when there is an IN token input in the next frame.

22.5.4.18 Incomplete synchronous OUT data transfers

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Endpoint Initialization to initialize endpoints. This section describes the application programming sequence when the controller drops synchronous OUT data packets.

[Internal data flow]

1. For synchronous OUT endpoints, the XFERC interrupt (in the OTGFS_DOEPINTx register) may not always be generated. If the controller drops synchronous OUT data packets, the application may fail to detect the XFERC interrupt in the OTGFS_DOEPINTx register.

- When the receive FIFO cannot accommodate the complete ISO OUT data packet, the controller drops the received ISO OUT data.
- When the synchronous OUT data packet is received with CRC errors.
- When the synchronous OUT token received by the controller is corrupted.
- When the application is very slow in reading the receive FIFO

2. When the controller detects the end of periodic frames before transfer complete to all synchronous OUT endpoints, an interrupt of incomplete synchronous OUT data is generated, indicating that an XFERC interrupt in the OTGFS_DOEPINTx register is not set on at least one of the synchronous OUT endpoints. At this point, the endpoint with the incomplete data transfer remains enabled, but no valid transfers are in progress on this endpoint.

[Application programming sequence]

1. The assertion of the incomplete synchronous OUT data interrupt indicates that at least one synchronous OUT endpoint has an incomplete data transfer in the current frame.

2. If this occurs because the synchronous OUT data is not completely read out from the endpoint, the application must empty all synchronous OUT data (data and status) in the receive FIFO before proceeding.

- When all data are read from the receive FIFO, the application can detect the XFERC interrupt in the OTGFS_DOEPINTx register. In this case, the application must re-enable the endpoint to receive the synchronous OUT data in the next frame by following the steps listed in “SETUP/Data IN/Status OUT”

3. When it receives an incomplete synchronous OUT data interrupt, the application must read the control registers of all synchronous OUT endpoints to determine which one of the endpoints has an incomplete data transfer in the current frame. An endpoint transfer is regarded as incomplete if both of the following conditions are met:

- OTGFS_DOEPCTLx. Even/Odd frame bit= OTGFS_DSTS.SOFFN[0]
- OTGFS_DOEPCTLx. Endpoint enable = 0x1

4. The pervious step must be performed before the SOF interrupt of the GINTSTS register is detected to ensure that the current frame number is not changed.

5. For synchronous OUT endpoints with incomplete transfers, the application must drop the data in memory, and disable the endpoint through the endpoint disable bit in the OTGFS_DOEPCTLx register.

6. Wait for the endpoint disable interrupt in the OTGFS_DOEPINTx register, and enable the endpoint to receive new data in the next frame by following the steps listed in “SETUP/Data IN/Status OUT”. Because the controller can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving wrong synchronous data.

22.5.4.19 Incomplete synchronous IN data transfers

This section describes how the application behaves on incomplete synchronous IN transfers.

[Internal data flow]

1. Synchronous IN transfers are incomplete on one of the following conditions:

- The controller receives corrupted synchronous IN tokens from more than one synchronous IN endpoints. In this case, the application can detect the incomplete synchronous IN transfer interrupt in the GINTSTS register.
- The application is slow in writing complete data to the transmit FIFO, and an IN token is received before the completion of data write. In this case, the application can detect the INTKNTXFEMP interrupt in the OTGFS_DIEPINTx register. The application ignores this interrupt, which will result in the generation of the incomplete synchronous IN transfer interrupt (in OTGFS_GINTSTS register). The controller responds to the received IN token by sending a zero-length data packet to the USB.

2. Either way, the application must stop writing the transmit FIFO as soon as possible.

3. The application must set the NAK and disable bits of the endpoints.

4. The controller disables the endpoint, clears the disable bit, and triggers the endpoint disable interrupt.

[Application programming sequence]

1. When the transmit FIFO becomes empty, the application ignores the INTKNTXFEMP interrupt (in the OTGFS_DIEPINTx register) from any synchronous IN endpoint because this can trigger the incomplete synchronous IN interrupt.

2. The incomplete synchronous IN transfer interrupt (in the OTGFS_GINTSTS register) indicates that at least one synchronous IN endpoint is with incomplete synchronous IN transfers.

3. The application must read the endpoint control registers of all synchronous IN endpoints to determine which one is with incomplete synchronous IN transfers.

4. The application must write data to the periodic transmit FIFO of the endpoint.

5. Disable theses endpoints by setting the following bits in the OTGFS_DIEPCTLx register

- OTGFS_DIEPCTLx.SETNAK = 0x1
- OTGFS_DIEPCTLx. endpoint enable = 0x1

6. The endpoint disable interrupt in the DIEPINTx register indicates that the controller has disabled the endpoint.

7. At this point, the application must empty the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next frame. The application must refresh the data through the OTGFS_GRSTCTL register.

22.5.4.20 Periodic IN (interrupt and synchronous) data transfers

This section describes a typical periodic IN data transfer.

To initialize the controller after power-on reset, the application must perform the steps list in OTGFS Initialization. Before communicating with a host, the controller must follow the steps defined in Endpoint Initialization to initialize endpoints.

[Application requirements]

1. Application requirements in “Non-periodic (bulk and control) IN data transfers” also apply to periodic IN data transfers, except for a slight difference of requirement 2.

- The application can only transmit multiples of largest-packet-size data packets, and a short packet. To transmit several largest-packet-size data packets and a short packet, the following conditions must be met:

Transfer size [epnum] = n * mps[epnum] + sp (where n and i are integers ≥ 0, and 0 ≤ sp < mps[epnum])

If (sp > 0), packet count [epnum] = n + 1. Otherwise, packet count [epnum] = n, mc[epnum] = packet count [epnum]

- The application cannot transmit a zero-length data packet at the end of a transfer. But it can transmit a single zero-length data packet in itself, provided packet count [epnum] = 1, mc[epnum] = packet count [epnum]

2. The application can only schedule data transfers of one frame at a time
- (OTGFS_DIEPTSIZx.MC – 1) * OTGFS_DIEPCTLx.MPS ≤OTGFS_DIEPTSIZx.XFERSIZ ≤OTGFS_DIEPTSIZx.MC * OTGFS_DIEPCTLx.MPS
- OTGFS_DIEPTSIZx.PKTCNT = OTGFS_DIEPTSIZx.MC
- If OTGFS_DIEPTSIZx.XFERSIZ < OTGFS_DIEPTSIZx.MC * OTGFS_DIEPCTLx.MPS, the last data packet of the transfer is a short packet.

3. For periodic IN endpoints, one-frame data must be prefetched before the data transfer in the next frame. This can be done by enabling periodic IN endpoint 1 frame before the scheduling of the frame to be transmitted.

4. The complete data to be transmitted in a frame must be written to the transmit FIFO by the application before the periodic IN token is received. Even when one-WORD data to be transmitted per frame is missing in the transmit FIFO while the periodic IN token is received, the controller behaves as when the FIFO is empty. When the transmit FIFO is empty, a zero-length data packet would be transmitted on the USB, and An NAK handshake signal would be transmitted for INTR IN endpoints.

[Internal data flow]

1. The application must set the transfer size and packet count bits of the endpoint registers, and enable the endpoint to transmit the data.

2. The application must also write the required data to the associated transmit FIFO.

3. Each time the application writes a packet to the transmit FIFO, the transfer size for the endpoint is decremented by the packet size. Continue to write data until the transfer size for the endpoint becomes 0

4. When an IN token for a periodic endpoint is received, the application writes the data to the FIFO (If any). If the complete data for the frame is not present in the FIFO, the controller generates an INTKNTXFEMP interrupt.
- A zero-length data packet is transmitted on the USB for synchronous IN endpoints
- An NAK handshake signal is transmitted on the USB for interrupt IN endpoints.

5. The packet count for the endpoints is decremented by one under the following conditions:
- For synchronous endpoints, when a zero-or non-zero-length data packet is transmitted
- For interrupt endpoints, when an ACK handshake is transmitted
- When the transfer size and packet count are both 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable bit is cleared.

6. In the “Periodic frame interval” (by the PERFRINT bit in the OTGFS_DCFG register), when the controller finds non-empty any one of the IN endpoint FIFOs scheduled for the current frame non-empty, the controller generates an INCOMPISOIN interrupt in the OTGFS_GINTSTS register.

[Application programming sequence (frame transfers)]

1. Program the OTGFS_DIEPTSIZx register

2. Program the OTGFS_DIEPCTLx register based on endpoint characteristics, and set the CNAK and endpoint enable bits

3. Write the data to be transmitted into the transmit FIFO.

4. The assertion of the INTKNTXFEMP interrupt indicates that the application has not yet written all data to be transferred into the transmit FIFO.

5. If the interrupt endpoint is already enabled while this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint to transmit data on the next IN token. If it is enabled while the interrupt is detected, refer to “Incomplete synchronous IN data transfers”.

6. When the interrupt IN endpoint is set as a periodic endpoint, the controller internally can process the timeout on the interrupt IN endpoint, without the need of the application intervention. Therefore, the application can never detect the TIMEOUT interrupt (in the OTGFS_DIEPINTx register) on the periodic interrupt IN endpoints.

7. The assertion of the XFERC interrupt in the OTGFS_DIEPINTx register but without the INTKNTXFEMP interrupt indicates the successful completion of a synchronous IN transfer. When reading the OTGFS_DIEPTSIZx register, only transfer size =0 and packet count =0 indicate that all data

are transmitted on the USB line.

8. The assertion of the XFERC interrupt in the OTGFS_DIEPINTx register, with or without the INTKNTXFEMP interrupt, indicates the successful completion of an interrupt IN transfer. When reading the OTGFS_DIEPTSIZx register, only transfer size =0 and packet count =0 indicate that all data are transmitted on the USB line.

9. The assertion of the INCOMPISOIN interrupt but without the above-mentioned interrupts indicates that the controller did not receive at least one periodic IN token in the current frame. Refer to “Incomplete synchronous IN data transfers” for more information on synchronous IN endpoints.

22.6 OTGFS control and status registers

The application controls the OTGFS controller by reading from and writing to the control and status registers (CSRx) through the AHB slave interface. These registers are accessible by 32 bits, and the addresses are 32-bit aligned.

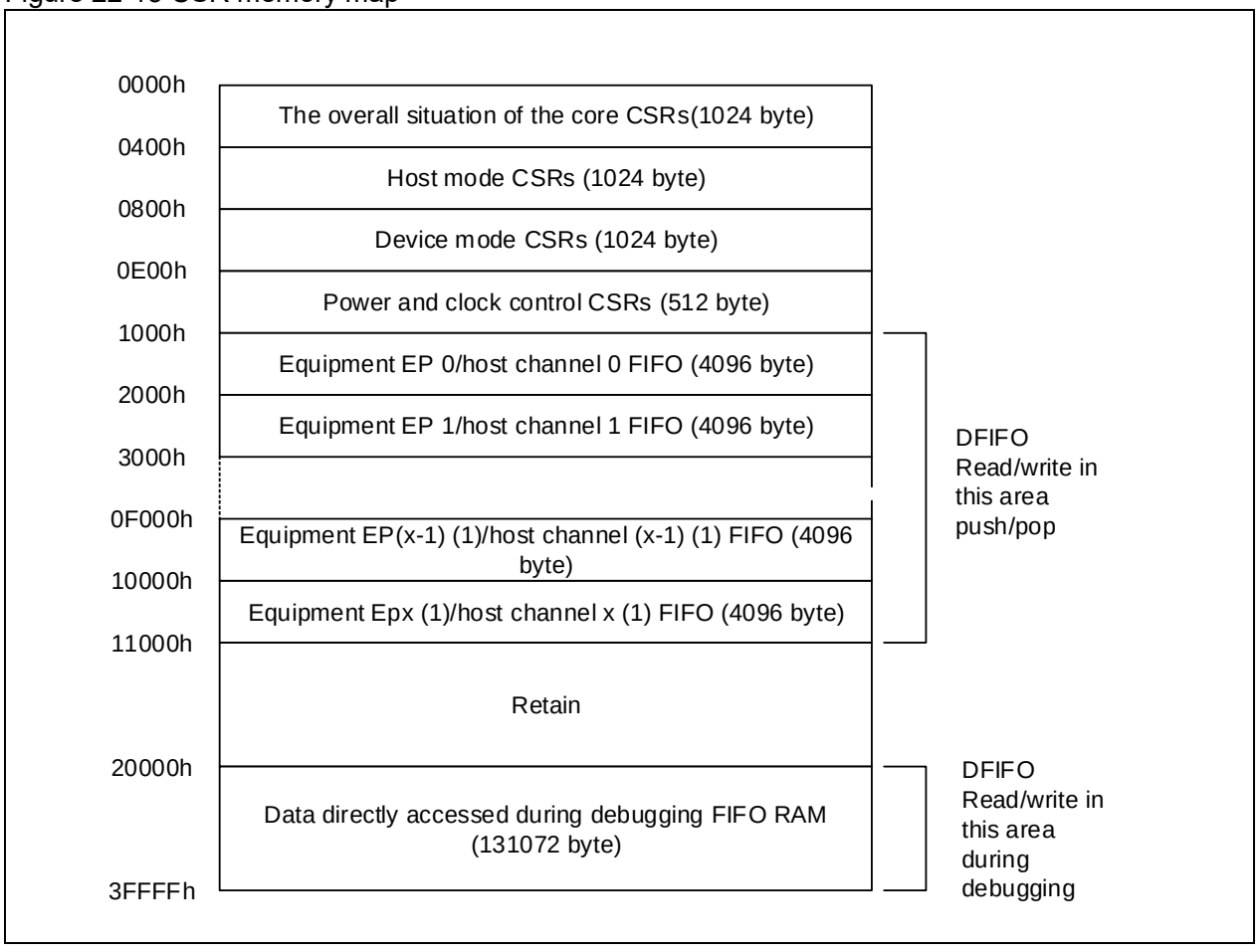
Only the controller global, power and clock control, data FIFO access and host port control and status registers are active in both host and device modes. When the OTGFS controller operates in either host or device mode, the application must not access the register group from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and the MODMIS bit (in the OTGFS_GINTSTS register) is affected.

When the controller switches from one mode to the other, the registers in the new mode must be re-initialized as they are after a power-on reset. These peripheral registers must be accessed by words (32-bit)

22.6.1 CSR register map

The host and device mode registers occupy different addresses. All registers are located in the AHB clock domain

Figure 22-13 CSR memory map



x = 7 in device mode, x =15 in host mode.

The OTGFS control and status registers contain OTGFS global register, host mode register, device mode register, data FIFO register, power and clock control register.

1. OTGFS global registers: They are active in both host and device modes. The register acronym is G.
2. Host-mode registers: They must be programmed every time the controller changes to host mode, The register acronym is H.
3. Device-mode registers: They must be programmed every time the controller changes to device mode, The register acronym is D.
4. Data FIFO access registers: These registers are valid in both in host and device modes, and are used to read or write the FIFO for a specific endpoint or channel in a given direction. If a host channel is of type IN, the FIFO can only be read. Similarly, if a host channel is of type OUT, the FIFO can only be written.
5. Power and clock control register: There is only one register for power and clock control. It is valid in both host and device modes.

22.6.2 OTGFS register address map

Table 22-4 shows the USB OTG register map and their reset values.

These peripheral registers must be accessed by words (32-bit).

Table 22-4 OTGFS register map and reset values

Register	Offset	Reset value
OTGFS_GOTGCTL	0x000	0x0001 0000
OTGFS_GOTGINT	0x004	0x0000 0000
OTGFS_GAHBCFG	0x008	0x0000 0000
OTGFS_GUSBCFG	0x00C	0x0000 1440
OTGFS_GRSTCTL	0x010	0x8000 0000
OTGFS_GINTSTS	0x014	0x0400 0020
OTGFS_GINTMSK	0x018	0x0000 0000
OTGFS_GRXSTSR	0x01C	0x0000 0000
OTGFS_GRXSTSP	0x020	0x0000 0000
OTGFS_GRXFSIZ	0x024	0x0000 0200
OTGFS_GNPTXFSIZ/ OTGFS_DIEPTXF0	0x028	0x0200 0200
OTGFS_GNPTXSTS	0x02C	0x0008 0200
OTGFS_GCCFG	0x038	0x0000 0000
OTGFS_GUID	0x03C	0x0000 1000
OTGFS_HPTXFSIZ	0x100	0x0200 0400
OTGFS_DIEPTXF1	0x104	0x0000 0000
OTGFS_DIEPTXF2	0x108	0x0000 0000
OTGFS_DIEPTXF3	0x10C	0x0000 0000
OTGFS_DIEPTXF4	0x110	0x0000 0000
OTGFS_DIEPTXF5	0x114	0x0000 0000
OTGFS_DIEPTXF6	0x118	0x0000 0000
OTGFS_DIEPTXF7	0x11C	0x0000 0000
OTGFS_HCFG	0x400	0x0000 0000
OTGFS_HFIR	0x404	0x0000 EA60
OTGFS_HFNUM	0x408	0x0000 3FFF

OTGFS_HPTXSTS	0x410	0x0008 0100
OTGFS_HAINT	0x414	0x0000 0000
OTGFS_HAINTMSK	0x418	0x0000 0000
OTGFS_HPRT	0x440	0x0000 0000
OTGFS_HCCHAR0	0x500	0x0000 0000
OTGFS_HCINT0	0x508	0x0000 0000
OTGFS_HCINTMSK0	0x50C	0x0000 0000
OTGFS_HCTSIZ0	0x510	0x0000 0000
OTGFS_HCCHAR1	0x520	0x0000 0000
OTGFS_HCINT1	0x528	0x0000 0000
OTGFS_HCINTMSK1	0x52C	0x0000 0000
OTGFS_HCTSIZ1	0x530	0x0000 0000
OTGFS_HCCHAR2	0x540	0x0000 0000
OTGFS_HCINT2	0x548	0x0000 0000
OTGFS_HCINTMSK2	0x54C	0x0000 0000
OTGFS_HCTSIZ2	0x550	0x0000 0000
OTGFS_HCCHAR3	0x560	0x0000 0000
OTGFS_HCINT3	0x568	0x0000 0000
OTGFS_HCINTMSK3	0x56C	0x0000 0000
OTGFS_HCTSIZ3	0x570	0x0000 0000
OTGFS_HCCHAR4	0x580	0x0000 0000
OTGFS_HCINT4	0x588	0x0000 0000
OTGFS_HCINTMSK4	0x58C	0x0000 0000
OTGFS_HCTSIZ4	0x590	0x0000 0000
OTGFS_HCCHAR5	0x5A0	0x0000 0000
OTGFS_HCINT5	0x5A8	0x0000 0000
OTGFS_HCINTMSK5	0x5AC	0x0000 0000
OTGFS_HCTSIZ5	0x5B0	0x0000 0000
OTGFS_HCCHAR6	0x5C0	0x0000 0000
OTGFS_HCINT6	0x5C8	0x0000 0000
OTGFS_HCINTMSK6	0x5CC	0x0000 0000
OTGFS_HCTSIZ6	0x5D0	0x0000 0000
OTGFS_HCCHAR7	0x5E0	0x0000 0000
OTGFS_HCINT7	0x5E8	0x0000 0000
OTGFS_HCINTMSK7	0x5EC	0x0000 0000
OTGFS_HCTSIZ7	0x5F0	0x0000 0000
OTGFS_HCCHAR8	0x600	0x0000 0000
OTGFS_HCINT8	0x608	0x0000 0000
OTGFS_HCINTMSK8	0x60C	0x0000 0000
OTGFS_HCTSIZ8	0x610	0x0000 0000
OTGFS_HCCHAR9	0x620	0x0000 0000

OTGFS_HCINT9	0x628	0x0000 0000
OTGFS_HCINTMSK9	0x62C	0x0000 0000
OTGFS_HCTSIZ9	0x630	0x0000 0000
OTGFS_HCCHAR10	0x640	0x0000 0000
OTGFS_HCINT10	0x648	0x0000 0000
OTGFS_HCINTMSK10	0x64C	0x0000 0000
OTGFS_HCTSIZ10	0x650	0x0000 0000
OTGFS_HCCHAR11	0x660	0x0000 0000
OTGFS_HCINT11	0x668	0x0000 0000
OTGFS_HCINTMSK11	0x66C	0x0000 0000
OTGFS_HCTSIZ11	0x670	0x0000 0000
OTGFS_HCCHAR12	0x680	0x0000 0000
OTGFS_HCINT12	0x688	0x0000 0000
OTGFS_HCINTMSK12	0x68C	0x0000 0000
OTGFS_HCTSIZ12	0x690	0x0000 0000
OTGFS_HCCHAR13	0x6A0	0x0000 0000
OTGFS_HCINT13	0x6A8	0x0000 0000
OTGFS_HCINTMSK13	0x6AC	0x0000 0000
OTGFS_HCTSIZ13	0x6B0	0x0000 0000
OTGFS_HCCHAR14	0x6C0	0x0000 0000
OTGFS_HCINT14	0x6C8	0x0000 0000
OTGFS_HCINTMSK14	0x6CC	0x0000 0000
OTGFS_HCTSIZ14	0x6D0	0x0000 0000
OTGFS_HCCHAR15	0x6E0	0x0000 0000
OTGFS_HCINT15	0x6E8	0x0000 0000
OTGFS_HCINTMSK15	0x6EC	0x0000 0000
OTGFS_HCTSIZ15	0x6F0	0x0000 0000
OTGFS_DCFG	0x800	0x0820 0000
OTGFS_DCTL	0x804	0x0000 0002
OTGFS_DSTS	0x808	0x0000 0010
OTGFS_DIEPMSK	0x810	0x0000 0000
OTGFS_DOEPMSK	0x814	0x0000 0000
OTGFS_DAINT	0x818	0x0000 0000
OTGFS_DAINTMSK	0x81C	0x0000 0000
OTGFS_DIEPEMPMSK	0x834	0x0000 0000
OTGFS_DIEPCTL0	0x900	0x0000 0000
OTGFS_DIEPINT0	0x908	0x0000 0080
OTGFS_DIEPTSIZ0	0x910	0x0000 0000
OTGFS_DTXFSTS0	0x918	0x0000 0200
OTGFS_DIEPCTL1	0x920	0x0000 0000
OTGFS_DIEPINT1	0x928	0x0000 0080

OTGFS_DIEPTSIZ1	0x930	0x0000 0000
OTGFS_DTXFSTS1	0x938	0x0000 0200
OTGFS_DIEPCTL2	0x940	0x0000 0000
OTGFS_DIEPINT2	0x948	0x0000 0080
OTGFS_DIEPTSIZ2	0x950	0x0000 0000
OTGFS_DTXFSTS2	0x958	0x0000 0200
OTGFS_DIEPCTL3	0x960	0x0000 0000
OTGFS_DIEPINT3	0x968	0x0000 0080
OTGFS_DIEPTSIZ3	0x970	0x0000 0000
OTGFS_DTXFSTS3	0x978	0x0000 0200
OTGFS_DIEPCTL4	0x980	0x0000 0000
OTGFS_DIEPINT4	0x988	0x0000 0080
OTGFS_DIEPTSIZ4	0x990	0x0000 0000
OTGFS_DTXFSTS4	0x998	0x0000 0200
OTGFS_DIEPCTL5	0x9A0	0x0000 0000
OTGFS_DIEPINT5	0x9A8	0x0000 0080
OTGFS_DIEPTSIZ5	0x9B0	0x0000 0000
OTGFS_DTXFSTS5	0x9B8	0x0000 0200
OTGFS_DIEPCTL6	0x9C0	0x0000 0000
OTGFS_DIEPINT6	0x9C8	0x0000 0080
OTGFS_DIEPTSIZ6	0x9D0	0x0000 0000
OTGFS_DTXFSTS6	0x9D8	0x0000 0200
OTGFS_DIEPCTL7	0x9E0	0x0000 0000
OTGFS_DIEPINT7	0x9E8	0x0000 0080
OTGFS_DIEPTSIZ7	0x9F0	0x0000 0000
OTGFS_DTXFSTS7	0x9F8	0x0000 0200
OTGFS_DOEPCTL0	0xB00	0x0000 8000
OTGFS_DOEPINT0	0xB08	0x0000 0000
OTGFS_DOEPTSIZ0	0xB10	0x0000 0000
OTGFS_DOEPCTL1	0xB20	0x0000 0000
OTGFS_DOEPINT1	0xB28	0x0000 0000
OTGFS_DOEPTSIZ1	0xB30	0x0000 0000
OTGFS_DOEPCTL2	0xB40	0x0000 0000
OTGFS_DOEPINT2	0xB48	0x0000 0000
OTGFS_DOEPTSIZ2	0xB50	0x0000 0000
OTGFS_DOEPCTL3	0xB60	0x0000 0000
OTGFS_DOEPINT3	0xB68	0x0000 0000
OTGFS_DOEPTSIZ3	0xB70	0x0000 0000
OTGFS_DOEPCTL4	0xB80	0x0000 0000
OTGFS_DOEPINT4	0xB88	0x0000 0000
OTGFS_DOEPTSIZ4	0xB90	0x0000 0000

OTGFS_DOEPCTL5	0xBA0	0x0000 0000
OTGFS_DOEPINT5	0xBA8	0x0000 0000
OTGFS_DOEPTSIZ5	0xBB0	0x0000 0000
OTGFS_DOEPCTL6	0xBC0	0x0000 0000
OTGFS_DOEPINT6	0xBC8	0x0000 0000
OTGFS_DOEPTSIZ6	0xBD0	0x0000 0000
OTGFS_DOEPCTL7	0xBE0	0x0000 0000
OTGFS_DOEPINT7	0xBE8	0x0000 0000
OTGFS_DOEPTSIZ7	0xBF0	0x0000 0000
OTGFS_PCGCCTL	0xE00	0x0000 0000

22.6.3 OTGFS global registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between two modes.

22.6.3.1 OTGFS status and control register (OTGFS_GOTGCTL)

This register controls the OTG function and reflects its status.

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x0000	resd	Kept at its default value.
Bit 21	CURMOD	0x0	ro	Current Mode of Operation Accessible in both host and device modes This bit indicates the current operation mode. 0: Device mode 1: Host mode
Bit 20: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	CONIDSTS	0x1	ro	Accessible in both host and device modes Connector ID status This bit indicates the connecter ID status. 0: OTGFS controller is in A-device mode 1: OTGFS controller is in B-device mode
Bit 15: 0	Reserved	0x0000	resd	Kept at its default value.

22.6.3.2 OTGFS interrupt status control register (OTGFS_GOTGINT)

The application reads this register to know about which kind of OTG interrupt is generated, and writes this register to clear the OTG interrupt.

Bit	Name	Reset value	Type	Description
Bit 31: 3	Reserved	0x0000	resd	Kept at its default value.
Bit 2	SESENDDET	0x0	rw1c	Available in both host and device modes Session end detected The controller sets this bit when a Bvalid (Vbus) signal is disconnected. This register can only be set by hardware. Writing 1 by software clears this bit.
Bit 1: 0	Reserved	0x0000	resd	Kept at its default value.

22.6.3.3 OTGFS AHB configuration register (OTGFS_GAHBCFG)

This register is used to configure the controller after power-on or mode change. This register mainly contains AHB-related parameters. Do not change this register after the initial configuration. The application must configure this register before starting transmission on either the AHB or USB.

Bit	Name	Reset value	Type	Description
Bit 31: 9	Reserved	0x000000	resd	Kept at its default value.
Bit 8	PTXFEMPLVL	0x0	rw	Accessible in host mode only Periodic TxFIFO empty level It indicates when the periodic TxFIFO empty interrupt bit in the GINTSTS register is triggered.

				0: PTXFEMP (GINTSTS) interrupt indicates that the periodic TxFIFO is half empty 1: PTXFEMP (GINTSTS) interrupt indicates that the periodic TxFIFO is fully empty
Bit 7	NPTXFEMPLVL	0x0	rw	Accessible in both host mode and device modes Non-Periodic TxFIFO empty level In host mode, this bit indicates when the non-periodic TxFIFO empty interrupt (NPTXFEMP in GINTSTS) is triggered. In device mode, this bit indicates when the IN endpoint TxFIFO empty interrupt (TXFEMP bit in DIEPINTn) is triggered. 0: The TxFEMP (in DIEPINTn) interrupt indicates that the IN endpoint TxFIFO is half empty 1: The TxFEMP (in DIEPINTn) interrupt indicates that the IN endpoint TxFIFO is fully empty
Bit 6: 1	Reserved	0x00	resd	Kept at its default value.
Bit 0	GLBINTMSK	0x0	rw	Accessible in both host mode and device modes Global interrupt mask The application uses this bit to mask or unmask the interrupts sent by the interrupt line to itself. 0: Mask the interrupts sent to the application 1: Unmask the interrupts sent to the application

22.6.3.4 OTGFS USB configuration register (OTGFS_GUSBCFG)

This register is used to configure the controller after power-on or a change between host mode and device mode. This register contains USB and USB-PHY related parameters. The application must program the register before handling any transaction on either the AHB or USB. Do not change this register after the initial configuration.

Bit	Name	Reset value	Type	Description
Bit 31	COTXPKT	0x0	rw	Accessible in both host mode and device modes Corrupt Tx packet This bit is for debug purpose only. Do not set this bit to 1.
Bit 30	FDEVMODE	0x0	rw	Accessible in both host mode and device modes Force device mode Writing 1 to this bit forces the controller to go into device mode, irrespective of the status of the ID input point. 0: Normal mode 1: Force device mode After setting this bit, the application must wait at least 25ms before the configuration takes effect.
Bit 29	FHSTMODE	0x0	rw	Accessible in both host mode and device modes Force host mode Writing 1 to this bit forces the controller to go into host mode, irrespective of the status of the ID input point. 0: Normal mode 1: Force host mode After setting this bit, the application must wait at least 25ms before the configuration takes effect.
Bit 28: 15	Reserved	0x0000	resd	Kept at its default value.
Bit 14	Reserved	0x0	resd	Kept at its default value.
Bit 13: 10	USBTRDTIM	0x5	rw	Accessible in device mode USB Turnaround Time This field sets the turnaround time in PHY clocks. It defines the response time when the MAC sends a request to the packet FIFO controller (PFC) to fetch data from the DFIFO (SPRAM). These bits must be configured as follows: 0101: When the MAC interface is 16-bit UTMI+ 1001: When the MAC interface is 8-bit UTMI+ Note: The aforementioned values are calculated based on a minimum of 30MHz AHB frequency. The USB turnaround time is critical for certifications with long cables and 5-Hub. If you want the AHB to run below 30

				MHz, and don't care about the USB turnaround time, you can set larger values for these bits.
Bit 9: 3	Reserved	0x08	resd	Kept at its default value.
Bit 2: 0	TOUTCAL	0x0	rw	Accessible in both host mode and device modes FS Timeout calibration The number of PHY clocks that the application programs in these bits is added to the full-speed interpacket timeout duration in order to compensate for any additional latency introduced by the PHY. This action can be required, because the delay triggered by the PHY while generating the line state condition can vary from one PHY to another. In full-speed mode, the USB standard timeout value is 16~18 (inclusive) bit times. The application must program these bits based on the enumeration speed. The number of bit times added per PHY clock is 0.25 bit times.

22.6.3.5 OTGFS reset register (OTGFS_GRSTCTL)

The application resets various hardware modules in the controller through this register.

Bit	Name	Reset value	Type	Description
Bit 31	AHBIDLE	0x1	ro	Accessible in both host mode and device modes AHB master Idle This bit indicates that the AHB master state machine is in idle condition.
Bit 30: 11	Reserved	0x000	resd	Kept at its default value.
Bit 10: 6	TXFNUM	0x00	rw	Accessible in both host mode and device modes TxFIFO number This field indicates the FIFO number that must be refreshed through the TxFIFO Flush bit. Do not make changes to this field until the controller clears the TxFIFO Flush bit. 00000: - Non-periodic TxFIFO in host mode - Tx FIFO 0 in device mode 00001: - Periodic TxFIFO in host mode - TXFIFO 1 in device mode 00010: - TXFIFO 2 in device mode ... 01111: - TXFIFO 15 in device mode 10000: - Refresh all the transmit FIFOs in device or host mode
Bit 5	TXFFLSH	0x0	rw1s	Accessible in both host mode and device modes TxFIFO Flush This bit selectively refreshes a single or all transmit FIFOs, but can do so when the controller is not in the process of a transaction. The application must write this bit only after checking that the controller is neither writing to nor reading from the TxFIFO. Verify using these registers: Read: NAK effective interrupt (NAK Effective Interrupt) ensures that the controller is not reading from the FIFO Write: AHBIDLE bit in GRSTCTL ensures that the controller is not writing to the FIFO. For FIFO reprogramming, it is usually recommended to carry out flushing operation. In device endpoint disable state, it is also advised to use FIFO flushing operation. The application must wait until the controller clears this bit, before performing other operations. It takes 8 clocks to clear this bit (slowest of phy_clk or hclk)
Bit 4	RXFFLSH	0x0	rw1s	Accessible in both host mode and device modes

				RxFIFO flush The application can refresh the entire RxFIFO using this bit, but must first ensure that the controller is not in the process of a transaction. The application must only write to this bit after checking that the controller is neither reading from nor writing to the RxFIFO. The application must wait until the controller clears this bit, before performing other operations. It takes 8 clocks to clear this bit (slowest of PHY or AHB)
Bit 3	Reserved	0x0	resd	Kept at its default value.
Bit 2	FRMCNTRST	0x0	rw1s	Accessible in both host mode and device modes Host frame counter reset The application uses this bit to reset the frame number counter inside the controller. After the frame counter is reset, the subsequent SOS sent out by the controller has a frame number of 0. If the application writes 1 to this bit, it may not be able to read the value, because this bit is cleared after a few clock cycles by the controller
Bit 1	PIUSFTRST	0x0	rw1s	Accessible in both host mode and device modes PIU FS dedicated controller soft reset This bit is used to reset PIU full-speed dedicated controller All state machines in the PIU full-speed dedicated controller are reset to the idle state. When the PHY remains in the receive state for more than one-frame time due to PHY errors (such as operation interrupted or babble), this bit can be used to reset the PIU full-speed dedicated controller. This is can be cleared automatically, the controller this clear this bit after all the necessary logic is reset in the controller.
Bit 0	CSFTRST	0x0	rw1s	Accessible in both host mode and device modes Controller soft reset Resets the hclk and phy_clock domain as follows: Clears all interrupts and CSR registers except for the following bits: - HCFG.FSLSPCS - DCFG.DECSPD - DCTL.SFTDIS Resets all state machines (except AHB slave) to the idle state, and clears all the transmit and receive FIFOs. All transactions on the AHB master are terminated as soon as possible after completing the last phase of an AHB data transfer. All transactions on the USB are terminated immediately. The application can write to this bit at any time to reset the controller. This is can be cleared automatically, the controller this clear this bit after all the necessary logic is reset in the controller. The controller could take several clocks to clear this bit, depending on the current state of the controller. Once this bit is cleared, the application must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). Additionally, the application must ensure that the bit 31 in this register is set (AHB master is in idle state) before performing other operations. Typically, the software set is used during software development and also when the user dynamically changes the PHY selection bits in the above-listed USB configuration registers. To change the PHY, the corresponding PHY clock is selected and used in the PHY domain. After a new clock is selected, the PHY domain has to be reset for normal operation.

22.6.3.6 OTGFS interrupt register (OTGFS_GINTSTS)

This register interrupts the application due to system-level events in the current mode (device or host mode), as shown in Figure 22-2.

Some of the bits in this register are valid only in host mode, while others are valid in device mode only. Besides, this register indicates the current mode.

The FIFO status interrupts are read-only. The FIFO interrupt conditions are cleared automatically as soon as the software reads from or writes to the FIFO while processing these interrupts.

The application must clear the GINTSTS register at initialization before enabling an interrupt bit to avoid any interrupt generation prior to initialization.

Bit	Name	Reset value	Type	Description
Bit 31	WKUPINT	0x0	rw1c	Accessible in both host mode and device modes Resume/Remote wakeup detected interrupt) In device mode, this interrupt is generated only when a resume signal (triggered by host) is detected on the USB bus. In host mode, this interrupt is generated only when a remote wakeup signal (triggered by device) is detected on the USB bus.
Bit 30	Reserved	0x0	resd	Kept at its default value.
Bit 29	DISCONINT	0x0	rw1c	Accessible in host mode only Disconnect detected interrupt The interrupt is generated when a device disconnect is detected.
Bit 28	CONIDSCHG	0x0	rw1c	Accessible in both host mode and device modes Connector ID status change This bit is set by the controller when there is a change in connector ID status.
Bit 27	Reserved	0x0	resd	Kept at its default value.
Bit 26	PTXFEMP	0x1	ro	Accessible in host mode only Periodic TxFIFO Empty The interrupt is generated when the Periodic Transmit FIFO is either half or completely empty and there is space for a request to be written in the periodic request queue. The half or completely empty status depends on the periodic transmit FIFO empty level bit in the AHB configuration register.
Bit 25	HCHINT	0x0	ro	Host channel interrupt The controller sets this bit to indicate that an interrupt is pending on one of the channels in the controller (in host mode). The application must read the Host All Channels Interrupt register to determine the exact number of the channel on which the interrupt occurred, and then read the Host Channel-n Interrupt register to determine the interrupt event source. The application must clear the corresponding status bit in the HCINTn (Host All Channels Interrupt) register to clear this bit.
Bit 24	PRTINT	0x0	ro	Host port interrupt The controller sets this bit to indicate a change in port status one of the ports. The application must read the Host Port Control and Status register to determine the exact event source. The application must clear the Host Port Control and Status register to clear this bit.
Bit 23: 22	Reserved	0x0	resd	Kept at its default value.
Bit 21	INCOMPIP INCOMPISOOUT	0x0	rw1c	Incomplete periodic transfer Accessible in host mode only In host mode, the controller sets this interrupt bit when there are incomplete periodic transfers still pending in the current frame. Incomplete Isochronous OUT Transfer Accessible in device mode only In device mode, the controller sets this interrupt bit to

				indicate that there is at least one synchronous OUT endpoint with incomplete transfers in the current frame. This interrupt is generated along with the End of Periodic Frame Interrupt bit in this register.
Bit 20	INCOMPISOIN	0x0	rw1c	Accessible in device mode only Incomplete Isochronous IN Transfer The controller sets this interrupt to indicate that there is at least one synchronous IN endpoint with incomplete transfers in the current frame. This interrupt is generated along with the End of Periodic Frame Interrupt bit in this register.
Bit 19	OEPTINT	0x0	ro	Accessible in device mode only OUT endpoints interrupt The controller sets this bit to indicate that an interrupt is pending on one of the OUT endpoints in the controller. The application must read the Device All Endpoints Interrupt register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-n Interrupt register to determine the exact source of the interrupt. The application must clear the corresponding status bit in the corresponding Device OUT Endpoint-n Interrupt register to clear this bit.
Bit 18	IEPTINT	0x0	ro	Accessible in device mode only IN Endpoints interrupt The controller sets this bit to indicate that an interrupt is pending one of the IN endpoints in the controller (in device mode). The application must read the Device All Endpoints Interrupt register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding Device IN Endpoint-n Interrupt register to determine the exact source of the interrupt. The application must clear the corresponding status bit in the corresponding Device IN Endpoint-n Interrupt register to clear this bit.
Bit 17: 16	Reserved	0x0	resd	Kept at its default value.
Bit 15	EOPF	0x0	rw1c	Accessible in device mode only End of periodic frame interrupt This bit indicates that the period programmed in the periodic frame interval bit of the Device Configuration register has been reached in the current frame.
Bit 14	ISOOUTDROP	0x0	rw1c	Accessible in device mode only Isochronous OUT packet dropped interrupt) The controller sets this bit on the following condition: the controller fails to write a synchronous OUT packet into the receive FIFO because the receive FIFO does not have enough space to accommodate a maximum size packet for the synchronous OUT endpoint.
Bit 13	ENUMDONE	0x0	rw1c	Accessible in device mode only Enumeration done The controller sets this bit to indicate that speed enumeration is done. The application must read the Device Status register to obtain the enumeration speed.
Bit 12	USBRST	0x0	rw1c	Accessible in device mode only USB Reset The controller sets this bit to indicate that a reset is detected on the USB bus.
Bit 11	USBSUSP	0x0	rw1c	Accessible in device mode only USB Suspend The controller sets this bit to indicate that a suspend is detected on the USB bus. The controller enters the Suspend state when there is no activity on the bus for a long period of time.

Bit 10	ERLYSUSP	0x0	rw1c	Accessible in device mode only Early suspend The controller sets this bit to indicate that the idle state has been detected on the USB bus for 3 ms.
Bit 9: 8	Reserved	0x0	resd	Kept at its default value.
Bit 7	GOUTNAKEFF	0x0	ro	Accessible in device mode only Global OUT NAK effective This bit indicates that the Set Global OUT NAK bit in the Device Control register (set by the application) has taken effect. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register.
Bit 6	GINNAKEFF	0x0	ro	Accessible in device mode only Global IN Non-periodic NAK effective This bit indicates that the Set Global Non-periodic IN NA bit in the Device Control register (set by the application) has taken effect. That is, the controller has sampled the Global IN NAK bit set by the application. This bit can be cleared by writing the Clear Global Non-periodic IN NA bit in the Device Control register. This interrupt does not necessarily mean that a NAK handshake signal is sent out on the USB bus. The STALL bit has priority over the NAK bit.
Bit 5	NPTXFEMP	0x1	ro	Accessible in both host and device modes Non-periodic TxFIFO empty This interrupt is generated when the Non-periodic TxFIFO is either half or completely empty and there is enough space for at least one request to be written to the Non-periodic Transmit Request Queue. The half or completely empty depends on the Non-periodic TxFIFO Empty Level bit in the Core AHB Configuration register.
Bit 4	RXFLVL	0x0	ro	Accessible in both host and device modes RxFIFO Non-Empty Indicates that there is at least one packet to be read from the receive FIFO.
Bit 3	SOF	0x0	rw1c	Accessible in both host and device modes Start of Frame In host mode, the controller sets this bit to indicate that an SOF (full-speed) or Keep-Alive (low-speed) is transmitted on the USB bus. The application must set this bit to 1 to clear this interrupt. In device mode, the controller sets this bit to indicate that an SOF token has been received on the USB bus. The application must read the Device Status register to get the current frame number. This interrupt can be generated only when the controller is running in FS mode. This bit is set by the controller. The application must write 1 to clear this bit. Note: Reading this register immediately after power-on reset may return the value 0x1. If this register is read as 0x1 immediately after power-on reset, it does not mean that an SOF has been transmitted (in host mode) or received (in device mode). The reading of this register is valid only when an effective connection has been established between the host and the device. If this bit is set after power-on reset, the application can clear this bit.
Bit 2	OTGINT	0x0	ro	Accessible in both host and device modes OTG interrupt The controller sets this bit to indicate that an OTG protocol event is generated. The application must read the OTGFS_GOTGINT register to determine the exact source that caused this interrupt. The application must clear the corresponding status bit in the OTGFS_GOTGINT register to clear this bit.
Bit 1	MODEMIS	0x0	rw1c	Accessible in both host and device modes

				Mode mismatch interrupt The controller sets this bit when the application is attempting to access: A host-mode register, when the controller is running in device mode A device-mode register, when the controller is running in host mode An OKAY response occurs when the register access is completed on the AHB, but it is ignored by the controller internally, and does not affect the operation of the controller. This bit can be set by the controller only. The application must write 1 to clear this bit.
Bit 0	CURMOD	0x0	ro	Accessible in both host and device modes Current mode of operation This bit indicates the current mode. 0: Device mode 1: Host mode

22.6.3.7 OTGFS interrupt mask register (OTGFS_GINTMSK)

This register works with the Interrupt Register to interrupt the application. When an interrupt bit is masked, the interrupt related to this interrupt bit is not generated. However, the Interrupt Register bit corresponding to this interrupt is still set.

Interrupt mask: 0

Interrupt unmask: 1

Bit	Name	Reset value	Type	Description
Bit 31	WKUPINTMSK	0x0	rw	Accessible in both host and device modes Resume/Remote wakeup detected interrupt mask
Bit 30	Reserved	0x0	resd	Kept at its default value.
Bit 29	DISCONINTMSK	0x0	rw	Accessible in both host and device modes Disconnect detected interrupt mask
Bit 28	CONIDSCHGMSK	0x0	rw	Accessible in both host and device modes Connector ID status change mask
Bit 27	Reserved	0x0	resd	Kept at its default value.
Bit 26	PTXFEMPMSK	0x0	rw	Accessible in host mode only Periodic TxFIFO empty mask
Bit 25	HCHINTMSK	0x0	rw	Accessible in host mode only Host channels interrupt mask
Bit 24	PRTINTMSK	0x0	ro	Accessible in host mode only Host port interrupt mask
Bit 23: 22	Reserved	0x0	resd	Kept at its default value.
Bit 21	INCOMPIPMSK INCOMPISOOUTMSK	0x0	rw	Incomplete periodic transfer mask Accessible in host mode only Incomplete isochronous OUT transfer mask Accessible in device mode only
Bit 20	INCOMISOINMSK	0x0	rw	Accessible in device mode only Incomplete isochronous IN transfer mask
Bit 19	OEPTINTMSK	0x0	rw	Accessible in device mode only OUT endpoints interrupt mask
Bit 18	IEPTINTMSK	0x0	rw	Accessible in device mode only IN endpoints interrupt mask
Bit 17	Reserved	0x0	rw	Kept at its default value.
Bit 16	Reserved	0x0	resd	Kept at its default value.
Bit 15	EOPFMSK	0x0	rw	Accessible in device mode only End of periodic frame interrupt mask
Bit 14	ISOOUTDROPMSK	0x0	rw	Device only isochronous OUT packet dropped interrupt mask
Bit 13	ENUMDONEMSK	0x0	rw	Accessible in device mode only Enumeration done mask
Bit 12	USBRSTMSK	0x0	rw	Accessible in device mode only USB Reset mask
Bit 11	USBSUSPMSK	0x0	rw	Accessible in device mode only

				USB suspend interrupt mask
Bit 10	ERLYSUSPMSK	0x0	rw	Accessible in device mode only Early suspend interrupt mask
Bit 9: 8	Reserved	0x0	resd	Kept at its default value.
Bit 7	GOUTNAKEFFMSK	0x0	rw	Accessible in device mode only Global OUT NAK effective mask
Bit 6	GINNAKEFFMSK	0x0	rw	Accessible in device mode only Global Non-periodic IN NAK effective mask
Bit 5	NPTXFEMPMSK	0x0	rw	Accessible in both host and device modes Non-periodic TxFIFO empty mask
Bit 4	RXFLVLMSK	0x0	rw	Accessible in both host and device modes Receive FIFO Non-empty mask
Bit 3	SOFMSK	0x0	rw	Accessible in both host and device modes Start of Frame mask
Bit 2	OTGINTMSK	0x0	rw	Accessible in both host and device modes OTG interrupt mask
Bit 1	MODEMISMSK	0x0	rw	Accessible in both host and device modes Mode mismatch interrupt mask
Bit 0	Reserved	0x0	resd	Kept at its default value.

22.6.3.8 OTGFS receive status debug read/OTG status read and POP registers (OTGFS_GRXSTSR / OTGFS_GRXSTSP)

A read to the Receive Status Debug Read register returns the data of the top of the Receive FIFO. A read to the Receive Status Read and Pop register pops the data of the top of the Receive FIFO.

The receive status contents are interpreted differently in host and device modes. Then controller ignores the receive status pop/read when the receive FIFO is empty and returns the value of 0x0000 0000. The application can only pop the receive status FIFO when the receive FIFO non-empty bit of the Core Interrupt register is set.

Host mode:

Bit	Name	Reset value	Type	Description
Bit 31: 21	Reserved	0x000	resd	Kept at its default value.
Bit 20: 17	PKTSTS	0x0	ro	Packet status Indicates the status of the received data packet. 0010: IN data packet received 0011: IN transfer completed (triggers an interrupt) 0101: Data toggle error (triggers an interrupt) 0111: Channel halted (triggers an interrupt) Others: Reserved Reset value: 0
Bit 16: 15	DPID	0x0	ro	Data PID Indicates the data PID of the received data packet. 00: DATA0 10: DATA1 01: DATA2 11: MDATA Reset value: 0
Bit 14: 4	BCNT	0x000	ro	Byte count Indicates the byte count of the received IN data packet.
Bit 3: 0	CHNUM	0x0	ro	Channel number Indicates the channel number to which the currently received data packet belongs.

Device mode:

Bit	Name	Reset value	Type	Description
Bit 31: 25	Reserved	0x00	resd	Kept at its default value.
Bit 24: 21	FN	0x0	ro	Frame number Indicates the least significant 4 bits of the frame number of the data packet received on the USB bus. This field is applicable only when the synchronous OUT endpoints are supported.

Bit 20: 17	PKTSTS	0x0	ro	Packet status Indicates the status of the received data packet. 0001: Global OUT NAK (triggers an interrupt) 0010: OUT data packet received 0011: OUT transfer completed (triggers an interrupt) 0100: SETUP transaction completed (triggers an interrupt) 0110: SETUP data packet received Others: Reserved
Bit 16: 15	DPID	0x0	ro	Data PID Indicates the data PID of the received OUT data packet. 00: DATA0 10: DATA1 01: DATA2 11: MDATA
Bit 14: 4	BCNT	0x000	ro	Byte count Indicates the byte count of the received data packet.
Bit 3: 0	EPTNUM	0x0	ro	Endpoint number Indicates the endpoint number to which the currently received data packet belongs.

22.6.3.9 OTGFS receive FIFO size register (OTGFS_GRXFSIZ)

The application can program the SRAM size that must be allocated to the receive FIFO.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	RXFDEP	0x0200	ro/rw	RxFIFO Depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 512 The power-on reset value of this register is defined as the largest receive data FIFO depth during the configuration.

22.6.3.10 OTGFS non-periodic Tx FIFO size (OTGFS_GNPTXFSIZ)/Endpoint 0 Tx FIFO size registers (OTGFS_DIEPTXF0)

The application can program the SRAM size and start address of the non-periodic transmit FIFO. The fields of this register varies with host mode or device mode.

Host:

Bit	Name	Reset value	Type	Description
Bit 31: 16	NPTXFDEP	0x0000	ro/rw	Non-periodic TxFIFO depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 256
Bit 15: 0	NPTXFSTADDR	0x0200	ro/rw	Non-periodic transmit SRAM start address This field contains the memory start address of the Non-periodic Transmit FIFO SRAM.

Device:

Bit	Name	Reset value	Type	Description
Bit 31: 16	INEPT0TXDEP	0x0000	ro/rw	N Endpoint TxFIFO 0 depth This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 256
Bit 15: 0	INEPT0TXSTADDR	0x0200	ro/rw	IN Endpoint FIFO0 transmit SRAM start address This field contains the memory start address of the IN Endpoint FIFO0 transmit SRAM.

22.6.3.11 OTGFS non-periodic Tx FIFO size/request queue status register (OTGFS_GNPTXSTS)

This register is valid in host mode only. It is a read-only register that contains the available space information for the Non-periodic TxFIFO and the Non-periodic Transmit Request Queue.

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30: 24	NPTXQTOP	0x00	ro	Top of the Non-periodic transmit request queue Indicates that the MAC is processing the request from the non-periodic transmit request queue. Bit [30: 27]: Channel/Endpoint number Bit [26: 25]: 00: IN/OUT token 01: Zero-length transmit packet (device IN/host OUT) 10: PING/CSPLIT token 11: Channel halted command Bit [24]: Terminate (last request for the selected channel/endpoint)
Bit 23: 16	NPTXQSPCAVAIL	0x08	ro	Non-periodic transmit request queue space available Indicates the amount of space available in the non-periodic transmit request queue. This queue supports both IN and OUT requests in host mode. 00: Non-periodic transmit request queue is full 01: 1 location available 02: 2 locations available N: n locations available (0 ≤ n ≤ 8) Others: Reserved Reset value: Configurable
Bit 15: 0	NPTXFSPCAVAIL	0x0200	ro	Non-periodic TxFIFO space available Indicates the amount of space available in the non-periodic TxFIFO. Values are in terms of 32-bit words. 00: Non-periodic transmit FIFO is full 01: 1 location available 02: 2 locations available N: n locations available (0 ≤ n ≤ 256) Others: Reserved Reset value: Configurable

22.6.3.12 OTGFS general controller configuration register (OTGFS_GCCFG)

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x00	resd	Kept at its default value.
Bit 21	VBUSIG	0x0	rw	VBUS ignored When this bit is set, the OTGFS controller does not monitor the Vbus pin voltage, and assumes that the Vbus is always active in both host and device modes, and leaves the Vbus pin for other purposes. 0: Vbus is not ignored 1: Vbus is ignored, and is deemed as always active
Bit 20	SOFOUTEN	0x0	rw	SOF output enable 0: No SOF pulse output 1: SOF pulse output on PIN
Bit 19: 18	Reserved	0x0	resd	Kept at its default value.
Bit 17	LP_MODE	0x0	rw	Low-power mode This bit is used to control the OTG PHY consumption. When this bit is set to 1 by software, the OTG PHY enters low-power mode; when this bit is cleared by software, the OTG PHY operates in normal mode. 0: Non-low-power mode 1: Low-power mode
Bit 16	PWRDOWN	0x0	rw	Power down This bit is used to activate the transceiver in transmission/reception. It must be pre-configured to allow

				USB communication. 0: Power down enable 1: Power down disable (Transceiver active)
Bit 15: 0	Reserved	0x0000	resd	Kept at its default value.

22.6.3.13 OTGFS controller ID register (OTGFS_GUID)

This is a read-only register containing the production ID.

Bit	Name	Reset value	Type	Description
31: 0	USERID	0x0000 1000	rw	Product ID field The application can program the ID field.

22.6.3.14 OTGFS host periodic Tx FIFO size register (OTGFS_HPTXFSIZ)

This register contains the size and memory start address of the periodic transmit FIFO.

Bit	Name	Reset value	Type	Description
Bit 31: 16	PTXFSIZE	0x02000	ro/rw	Host periodic TxFIFO depth Values are in terms of 32-bit words. Minimum value is 16 Maximum value is 512
Bit 15: 0	PTXFSTADDR	0x0600	ro/rw	Host Periodic TxFIFO start address The power-on reset value of this register is the sum of the largest receive FIFO depth and the largest non-periodic transmit FIFO depth.

22.6.3.15 OTGFS device IN endpoint Tx FIFO size register (OTGFS_DIEPTXFn) (x=1...7, where n is the FIFO number)

This register holds the depth and memory start address of the IN endpoint transmit FIFO in device mode. Each of the FIFOs contains an IN endpoint data. This register can be used repeatedly for instantiated IN endpoint FIFO1~15. The GNPTXFSIZ register is used to program the depth and memory start address of the IN endpoint FIFO 0.

Bit	Name	Reset value	Type	Description
Bit 31: 16	INEPTXFDEP	0x0200	ro/rw	IN Endpoint TxFIFO depth Values are in terms of 32-bit words. Minimum value is 16 Maximum value is 512 The reset value is the maximum possible IN endpoint transmit FIFO depth
Bit 15: 0	INEPTXFSTADDR	0x0400	ro/rw	IN Endpoint FIFOn transmit SRAM start address This field contains the SRAM start address of the IN endpoint n transmit FIFO

22.6.4 Host-mode registers

Host-mode registers affect the operation of the controller in host mode. Host-mode register are not accessible in device mode (as the results are undefined in device mode). Host-mode registers contain as follows:

22.6.4.1OTGFS host mode configuration register (OTGFS_HCFG)

This register is used to configure the controller after power-on. Do not change this register after initialization.

Bit	Name	Reset value	Type	Description
Bit 31: 3	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 2	FSLSSUPP	0x0	ro	FS- and LS-only support The application uses this bit to control the controller's enumeration speed. With this bit, the application can make the controller enumerate as a full-speed host mode, even if the connected device supports high-speed communication. Do not change this bit after initial programming. 0: FS/LS, depending on the largest speed supported by the connected device.

				1: FS/LS-only, even if the connected device supports high-speed.
Bit 1: 0	FSLSPCLKSEL	0x0	rw	FS/LS PHY clock select When the controller is in FS host mode: 01: PHY clock is running at 48MHz Others: Reserved When the controller is in LS host mode: 00: Reserved 01: PHY clock is running at 48 MHz 10: PHY clock is running at 6 MHz. If 6 MHz clock is selected, reset must be done by software. 11: Reserved

22.6.4.2 OTGFS host frame interval register (OTGFS_HFIR)

This register is used to program the current

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	HFIRRLDCTRL	0x0	rw	Reload control This bit is used to disable/enable dynamic reload for the host frame register at runtime. 1: Reload control disable 0: Reload control enable This bit must be configured at initialization. Do not change its value at runtime.
Bit 15: 0	FRINT	0xEA60	rw	Frame interval The application uses this filed to program the interval between two consecutive SOFs (full speed) The number of PHY locks in this field indicates the frame interval. The application can write a value to the host frame interval register only after the port enable bit in the host port control and status register has been set. If no value is programmed, the controller calculates the value based on the PHY clock frequency defined in the FS/LS PHY clock select bit of the host configuration register. Do not change the value of this field after initial configuration. 1 ms * (FS/LS PHY clock frequency)

22.6.4.3 OTGFS host frame number/frame time remaining register (OTGFS_HFNUM)

This register indicates the current frame number, and also the time remaining in the current frame (in terms of the number of PHY clocks).

Bit	Name	Reset value	Type	Description
Bit 31: 16	FTREM	0x0000	ro	Frame time remaining Indicates the time remaining in the current frame (FS/HS), in terms of the number of PHY clocks. This field decrements with the number of PHY clocks. When it reaches zero, this filed is reloaded with the value of the frame interval register, and a new SOF is transmitted on the USB bus.
Bit 15: 0	FRNUM	0x3FFF	ro	Frame number This field increments every time a new SOP is transmitted on the USB bus, and is cleared to 0 when the value reaches 16'h3FFF.

22.6.4.4 OTGFS host periodic Tx FIFO/request queue register (OTGFS_HPTXSTS)

This is a read-only register containing the free space information of the periodic Tx FIFO and the periodic transmit request queue.

Bit	Name	Reset value	Type	Description
Bit 31: 24	PTXQTOP	0x00	ro	Top of the periodic transmit request queue) Indicates that the MAC is processing the request from the periodic transmit request queue. This register is used for debugging. Bit [31]: Odd/Even frame 0: Transmit in even frame 1: Transmit in odd frame Bit [30: 27]: Channel/Endpoint number Bit [26: 25]: Type 00: IN/OUT 01: Zero-length packet 10: Reserved 11: Channel command disable Bit [24]: Terminate (last request for the selected channel or endpoint)
Bit 23: 16	PTXQSPCAVAIL	0x08	ro	Periodic transmit request queue space available Indicates the number of free space available to be written in the periodic transmit request queue. This queue contains both IN and OUT requests. 00: Periodic transmit request queue is full 01: 1 space available 10: 2 space available N: n space available (0 ≤ n ≤ 8) Others: Reserved
Bit 15: 0	PTXFSPCAVAIL	0x0100	rw	Periodic transmit data FIFO space available Indicates the number of free space available to be written in the periodic transmit FIFO, in terms of 32-bit words. 0000: Periodic transmit FIFO is full 0001: 1 space available 0010: 2 space available N: n space available (0 ≤ n ≤ 512) Others: Reserved

22.6.4.5 OTGFS host all channels interrupt register (OTGFS_HAINT)

When a flag event occurs on a channel, the host all channels interrupt register interrupts the application through the host channels interrupt bit of the controller interrupt register, as shown in Figure 22-2. There is one interrupt bit for each channel, up to 16 bits. The application sets or clears this register by setting or clearing the appropriate bit in the corresponding host channel-n interrupt register.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	HAINT	0x0000	ro	Channel interrupts One bit per channel: bit 0 for channel 0, bit 15 for channel 15.

22.6.4.6 OTGFS host all channels interrupt mask register (OTGFS_HAINTMSK)

The host all channels interrupt mask register works with the host all channels interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per one channel, 16 bits in total.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	HAINTMSK	0x0000	rw	Channel interrupt mask One bit per channel: bit 0 for channel 0, bit 15 for channel 15.

22.6.4.7 OTGFS host port control and status register (OTGFS_HPRT)

This register is valid only in host mode. Currently, the OTG host supports only one port.

This register contains USB port-related information such as USB reset, enable, suspend, resume, connect status and test mode, as show in Figure 22-2. The register of type rw1c can interrupt the application through the host port interrupt bit in the controller interrupt register. Upon a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the register of type rw1c, the application must write 1 to clear the interrupt.

Bit	Name	Reset value	Type	Description
Bit 31: 19	Reserved	0x0000	resd	Kept at its default value.
Bit 18: 17	PRTSPD	0x0	ro	Port speed Indicates the speed of the device connected to this port. 00: Reserved 01: Full speed 10: Low speed 11: Reserved
Bit 16: 13	PRTTSTCTL	0x0	rw	Port test control The application writes a non-zero value to this field to put the port into test mode, and the port gives a corresponding signal. 0000: Test mode disabled 0001: Test_J mode 0010: Test_K mode 0011: Test_SE0_NAK mode 0100: Test_Packet mode 0101: Test_Force_Enable Others: Reserved
Bit 12	PRTPWR	0x0	rw	Port power The application uses this bit to control power supply to this port (by writing 1 or 0) 0: Power off 1: Power on Note: This bit is not associated with interfaces. The application must follow the programming manual to set this bit for various interfaces.
Bit 11: 10	PRTLNSTS	0x0	ro	Port line status Indicates the current logic status of the USB data lines. Bit [10]: Logic level of D+ Bit [11]: Logic level of D–
Bit 9	Reserved	0x0	resd	Kept at its default value.
Bit 8	PRTRST	0x0	rw	Port reset When this bit is set by the application, a reset sequence is started on this port. The application must calculate the time required for the reset sequence, and clear this bit after the reset sequence is complete. 0: Port not in reset 1: Port in reset The application must keep this bit set for a minimum duration defined in Section 7.1.7.5 of USB 2.0 specification to start a reset on the port. In addition to this, the application can make this bit set for another 10 ms to the minimum duration, before clearing this bit. There is no maximum limit set by the USB standard.
Bit 7	PRTSUSP	0x0	rw1s	Port suspend The application sets this bit to put this port in suspend mode. In this case, the controller only stops sending SOF. The application must set the port clock stop bit in order to disable the PHY clock. The read value of this bit reflects the current suspend status of the port. This bit is cleared by the controller when a remote wakeup signal is detected or when the application sets the port reset bit or port resume bit in this register, or sets the

				resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the controller interrupt register. The controller can still clear this bit, even if the device is disconnected with the host. 0: Port not in suspend mode 1: Port in suspend mode
Bit 6	PRTRES	0x0	rw	Port resume The application sets this bit to drive resume signaling on the port. The controller continues to trigger the resume signal until the application clears this bit. If the controller detects a USB remote wakeup sequence (as indicated by the port resume/remote wakeup detected interrupt bit of the controller interrupt register), the controller starts driving resume signaling without the intervention of the application. The read value of this bit indicates whether the controller is currently driving resume signaling. 0: No resume triggered 1: Resume triggered
Bit 5	PRTOVRCCHNG	0x0	rw1c	Port overcurrent change The controller sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes. This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 4	PRTOVRCACT	0x0	ro	Port overcurrent active Indicates the overcurrent status of the port. 0: No overcurrent 1: Overcurrent condition
Bit 3	PRTENCHNG	0x0	rw1c	Port enable/disable change The controller sets this bit when the status of the port enable bit 2 in this register changes. This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 2	PRTENA	0x0	rw1c	Port enable A port is enabled only by the controller after a reset sequence. This port is enabled by an overcurrent condition, a disconnected condition ro by the application. The application cannot set this bit by a register write operation. It can only clear this bit to disable the port. This bit does not trigger any interrupt. 0: Port disabled 1: Port enabled
Bit 1	PRTCONDET	0x0	rw1c	Port connect detected On a device connection detected, the controller sets this bit using the host port interrupt bit in the controller register. This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 0	PRTCONSTS	0x0	ro	Port connect status 0: No device is connected to the port 1: A device is connected to the port

22.6.4.8 OTGFS host channel x characteristics register (OTGFS_HCCHARx) (x = 0...15, where x= channel number)

Bit	Name	Reset value	Type	Description
Bit 31	CHENA	0x0	rw1s	Channel enable This bit is set by the application and cleared by the OTG host. 0: Channel disabled 1: Channel enabled
Bit 30	CHDIS	0x0	rw1s	Channel disable The application sets this bit to stop transmitting or receiving data on a channel, even before the transfer on

				that channel is complete. The application must wait for the generation of the channel disabled interrupt before treating the channel as disabled.
Bit 29	ODDFRM	0x0	rw	Odd frame This bit is set / cleared by the application to indicate that the OTG host must perform a transfer in an odd frame. This bit is applicable for periodic transfers (synchronous and interrupt) only. 0: Even frame 1: Odd frame
Bit 28: 22	DEVADDR	0x00	rw	Device address This field is used to select the device that can serve as the data source or receiver.
Bit 21: 20	MC	0x0	rw	Multi count (MC) This field indicates to the host the number of transfers that must be performed per frame for the periodic endpoint. 00: Reserved. This field generates undefined results. 01: 1 transaction 10: 2 transactions per frame 11: 3 transactions per frame This field must be set to at least 0x01.
Bit 19: 18	EPTYPE	0x0	rw	Endpoint type Indicates the transfer type selected. 00: Control transfer 01: Synchronous transfer 10: Bulk transfer 11: Interrupt transfer
Bit 17	LSPDDEV	0x0	rw	Low-speed device The application sets this bit to indicate that this channel is communicating to a low-speed device.
Bit 16	Reserved	0x0	resd	Kept at its default value.
Bit 15	EPTDIR	0x0	rw	Endpoint direction Indicates whether the transfer is in IN or OUT. 0: OUT 1: IN
Bit 14: 11	EPTNUM	0x0	rw	Endpoint number Indicates the endpoint number on the device (serving as data source or receiver)
Bit 10: 0	MPS	0x000	rw	Maximum packet size Indicates the maximum packet size of the corresponding port.

22.6.4.9 OTGFS host channelx interrupt register (OTGFS_HCINTx) (x = 0...15, where x= channel number)

This register contains the status of a channel related to USB and AHB events, as shown in Figure 22-2. The application must read this register when the host channels interrupt bit is set in the controller interrupt register. Before reading this register, the application must read the host all channels interrupt register to get the exact channel number of the host channel-n interrupt register. The application must clear the corresponding bit in this register to clear the corresponding bits in the OTGFS_HAIN and OTGFS_GINTSTS registers.

Bit	Name	Reset value	Type	Description
Bit 31: 11	Reserved	0x000000	resd	Kept at its default value.
Bit 10	DTGLERR	0x0	rw1c	Data toggle error This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 9	FRMOVRUN	0x0	rw1c	Frame overrun This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 8	BBLERR	0x0	rw1c	Babble error This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 7	XACTERR	0x0	rw1c	Transaction error Indicates one of the following errors occurred on the USB

				bus: CRC check failure Timeout Bit stuffing error EOP error This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 6	Reserved	0x0	resd	Kept at its default value.
Bit 5	ACK	0x0	rw1c	ACK response received/Transmitted interrupt This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 4	NAK	0x0	rw1c	NAK response received interrupt This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 3	STALL	0x0	rw1c	STALL response received interrupt This bit can only be set by the controller. The application must write 1 to clear this bit.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	CHHLTD	0x0	rw1c	Channel hated Indicates that the transfer completed abnormally either because of any transfer error or in response to a disable request by the application.
Bit 0	XFERC	0x0	rw1c	Transfer completed Transfer completed normally, without any error. This bit can only be set by the controller. The application must write 1 to clear this bit.

22.6.4.10 OTGFS host channelx interrupt mask register (OTGFS_HCINTMSKx) (x = 0...15, where x= channel number)

This register is used to mask the channels described in the previous section.

Bit	Name	Reset value	Type	Description
Bit 31: 11	Reserved	0x000000	resd	Kept at its default value.
Bit 10	DTGLERRMSK	0x0	rw	Data toggle error mask
Bit 9	FRMOVRUNMSK	0x0	rw	Frame overrun mask
Bit 8	BBLERRMSK	0x0	rw	Babble error mask
Bit 7	XACTERRMSK	0x0	rw	Transaction error mask
Bit 6	NYETMSK	0x0	rw	NYET response received interrupt mask
Bit 5	ACKMSK	0x0	rw	ACK response received/transmitted interrupt mask
Bit 4	NAKMSK	0x0	rw	NAK response received interrupt mask
Bit 3	STALLMSK	0x0	rw	STALL response received interrupt mask
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	CHHLTDMSK	0x0	rw	Channel halted mask
Bit 0	XFERCMSK	0x0	rw	Transfer completed mask

22.6.4.11 OTGFS host channelx transfer size register (OTGFS_HCTSIZx) (x = 0...15, where x= channel number)

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30: 29	PID	0x0	rw	PID (Pid) The application programs this field with the type of PID used for the initial transfer. The host controls this filed for the rest of transfers. 00: DATA0 01: DATA2 10: DATA1 11: MDATA(non-control)/SETUP(control)
Bit 28: 19	PKTCNT	0x000	rw	Packet count The application programs this field with the expected number of packets to be transmitted or received. The host decrements the packet count on every successful transmission or reception of an OUT/IN packet. When this count reaches zero, the application is interrupted to

				indicate normal completion of the transfer.
Bit 18: 0	XFERSIZE	0x00000	rw	Transfer size For an OUT transfer, this field indicates the number of data bytes the host sends during a transfer. For an IN transfer, this field indicates the buffer size that the application has reserved for the transfer. For an IN transfer (periodic and non-periodic), the application must program this field as an integer multiple of the maximum packet size.

22.6.5 Device-mode registers

These registers are applicable in device mode only. They are not supported in host mode due to unknown access results. Some of the registers affect all the endpoints, while some affect only one endpoint.

22.6.5.1 OTGFS device configure register (OTGFS_DCFG)

This register configures the controller in device mode after power-on or after certain control commands or enumeration. Do not change this register after initial programming.

Bit	Name	Reset value	Type	Description
Bit 31: 13	Reserved	0x0110	resd	Kept at its default value.
Bit 12: 11	PERFRINT	0x0	rw	Periodic frame interval This field indicates the time within a frame at which the periodic frame end interrupt is generated. The application can use this interrupt to determine if the synchronous transfer has been completed in a frame. 00: 80% of the frame interval 01: 85% of the frame interval 10: 90% of the frame interval 11: 95% of the frame interval
Bit 10: 4	DEVADDR	0x00	rw	Device address The application must program this field every time a SetAddress command is received.
Bit 3	Reserved	0x0	resd	Kept at its default value.
Bit 2	NZSTSOUTHSHK	0x0	rw	Non-zero-length status OUT handshake The application can use this field to select the handshake the controller sends on receiving a non-zero-length data packet during a control transfer' status stage. 1: Send a STALL handshake on a non-zero-length status OUT transfer and do not send the received OUT packet to the application 0: Send the received OUT packet to the application (zero-length or non-zero-length), and send a handshake based on the NAK and STALL bits in the device endpoint control register.
Bit 1: 0	DEVSPD	0x0	rw	Device speed This field indicates the speed at which the application needs the controller to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the entire sequence is complete, and is based on the speed of the USB host to which the controller is connected. 00: Reserved 01: Reserved 10: Reserved 11: Full speed (USB1.1 transceiver, clock is 48MHz)

22.6.5.2 OTGFS device control register (OTGFS_DCTL)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value.
Bit 11	PWROPRGDNE	0x0	wo	Power-on programming done The application uses this bit to indicate that the register configuration is complete after a wakeup from power-down mode.
Bit 10	CGOUTNAK	0x0	wo	Clear global OUT NAK

				Writing 1 to this bit clears the global OUT NAK.
Bit 9	SGOUTNAK	0x0	wo	Set global OUT NAK Writing to this bit sets the global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after checking that the global OUT NAK effective bit in the controller interrupt register is cleared.
Bit 8	CGNPINNAK	0x0	wo	Clear Global Non-periodic IN NAK Writing to this bit clears the global Non-periodic OUT NAK.
Bit 7	SGNPINNAK	0x0	wo	Set global Non-periodic IN NAK Writing to this bit sets the global Non-periodic OUT NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The application must set this bit only after checking that the global IN NAK effective bit in the controller interrupt register is cleared.
Bit 6: 4	TSTCTL	0x0	rw	Test control 000: Test mode disabled 001: Test_J mode 010: Test_K mode 011: Test_SE0_NAK mode 100: Test_Packet mode 101: Test_Force_Enable Others: Reserved
Bit 3	GOUTNAKSTS	0x0	ro	Global OUT NAK status 0: A handshake is sent based on the FIFO status, NAK and STALL bit settings. 1: No data is written to the receive FIFO, irrespective of space availability. Sends a NAK handshake on all packets (except on SETUP transfers). Drops all synchronous OUT packets.
Bit 2	GNPINNAKSTS	0x0	ro	Global Non-periodic IN NAK status 0: A handshake is sent based on the data status in the transmit FIFO 1: A NAK handshake is sent on all non-periodic IN endpoints, irrespective of the data status in the transmit FIFO.
Bit 1	SFTDISCON	0x1	rw	Software disconnect The application uses this bit to indicate the OTGFS controller to perform software disconnected. Once this bit is set, the host finds the device disconnected, and the device does not receive signals on the USB bus. The controller stays in the disconnected state until the application clears this bit. 0: Normal operation. When this bit is cleared after a software disconnect, the controller issues a device connect event to the host. Then the USB host restarts device enumeration.
Bit 0	RWKUPSIG	0x0	rw	Remote wakeup signaling When this bit is set by the application, the controller initiates a remote signal to wakeup the USB host. The application must set this bit to indicate the controller to exit the suspend mode. Per USB2.0 standards, the application must clear this bit 1-15 ms after setting it.

Table 22-5 lists the minimum duration at which the software disconnect bit must be set in various states for the USB host to detect a device disconnect. To accommodate clock jitter, it is advised that the application adds some extra delay to the specified minimum duration.

Table 22-5 Minimum duration for software disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspend	1ms + 2.5us
Full speed	Idle	2.5us
Full speed	No idle or suspend (performing transfers)	2.5us

22.6.5.3 OTGFS device status register (OTGFS_DSTS)

This register indicates the status of the controller related to OTGFS events. It must be read on interrupt events from the device all interrupts register (OTGFS_DAINT).

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value.
Bit 21: 8	SOFFN	0x0000	ro	Frame number of the received SOF Note: The read value of this field immediately after power-on reset reflects a non-zero value. If a non-zero value is returned after reading this field immediately after power-on reset, it does not mean that the host has received a SOP. The read value of this field is valid only when the host is connected to the device.
Bit 7: 4	Reserved	0x1	resd	Kept at its default value.
Bit 3	ETICERR	0x0	ro	Erratic error This error causes the controller to enter suspend mode, and interrupt is generated with the early suspend bit of the controller interrupt register. If the early suspend is asserted due to an erratic error, the application can only perform a software disconnect recover.
Bit 2: 1	ENUMSPD	0x0	ro	Enumerated speed Indicates the speed at which the controller has determined after speed detection through a sequence. 01: Reserved 10: Reserved 11: Full speed (PHY clock is running at 48MHz) Others: Reserved
Bit 0	SUSPSTS	0x0	ro	Suspend status In device mode, this bit is set as long as a suspend condition is detected on the USB bus. The controller enters the suspend state when there is no activity on the USB bus. The controller exits the suspend state on the following conditions: When there is an activity on the USB bus When the application writes to the remote wakeup signal bit in the device control register.

22.6.5.4 OTGFS device OTGFSIN endpoint common interrupt mask register (OTGFS_DIEPMSK)

This register works with each of the device IN endpoint interrupt register for all endpoints to generate an IN endpoint interrupt. The IN endpoint interrupt for a specific status in the OTGFS_DIEPINTx register can be masked by writing to the corresponding bit in the OTGFS_DIEPMSK register. Status bits are masked by default.

Bit	Name	Reset value	Type	Description
Bit 31: 10	Reserved	0x000000	resd	Kept at its default value.
Bit 9	BNAINMSK	0x0	rw	BNA interrupt mask 0: Interrupt masked 1: Interrupt unmasked
Bit 8	TXFIFOUDRMSK	0x0	rw	FIFO underrun mask 0: Interrupt masked 1: Interrupt unmasked
Bit 7	Reserved	0x0	resd	Kept at its default value.

Bit 6	INEPTNAKMSK	0x0	rw	IN endpoint NAK effective mask 0: Interrupt masked 1: Interrupt unmasked
Bit 5	INTKNEPTMISMSK	0x0	rw	IN token received with EP mismatch mask 0: Interrupt masked 1: Interrupt unmasked
Bit 4	INTKNTXFEMPMSK	0x0	rw	IN token received when TxFIFO empty mask 0: Interrupt masked 1: Interrupt unmasked
Bit 3	TIMEOUTMSK	0x0	rw	Timeout condition mask (Non-isochronous endpoints)) 0: Interrupt masked 1: Interrupt unmasked
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	EPTDISMSK	0x0	rw	Endpoint disabled interrupt mask 0: Interrupt masked 1: Interrupt unmasked
Bit 0	XFERCMSK	0x0	rw	Transfer completed interrupt mask 0: Interrupt masked 1: Interrupt unmasked

22.6.5.5 OTGFS device OUT endpoint common interrupt mask register (OTGFS_DOEPMSK)

This register works with each of the OTGFS_DOEPINTx registers for all endpoints to generate an OUT endpoint interrupt. Each of the bits in the OTGFS_DOEPINTx registers can be masked by writing to the register. All interrupts are masked by default.

Bit	Name	Reset value	Type	Description
Bit 31:10	Reserved	0x000000	resd	Kept at its default value.
Bit 9	BNAOUTMSK	0x0	rw	BNA interrupt mask 0: Interrupt masked 1: Interrupt unmasked
Bit 8	OUTPERRMSK	0x0	rw	OUT packet error mask 0: Interrupt masked 1: Interrupt unmasked
Bit 7	Reserved	0x0	resd	Kept at its default value.
Bit 6	B2BSETUPMSK	0x0	rw	Back-to-back SETUP packets received mask 0: Interrupt masked 1: Interrupt unmasked
Bit 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	OUTTEPDMSK	0x0	rw	OUT token received when endpoint disabled mask 0: Interrupt masked 1: Interrupt unmasked
Bit 3	SETUPMSK	0x0	rw	SETUP phase done mask Applies to control endpoints only. 0: Interrupt masked 1: Interrupt unmasked
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	EPTDISMSK	0x0	rw	Endpoint disabled interrupt mask 0: Interrupt masked 1: Interrupt unmasked
Bit 0	XFERCMSK	0x0	rw	Transfer completed interrupt mask 0: Interrupt masked 1: Interrupt unmasked

22.6.5.6 OTGFS device all endpoints interrupt mask register (OTGFS_DAINT)

When an event occurs on an endpoint, The IN/OUT endpoint interrupt bits in the OTGS_DAINT register can be used to interrupt the application. There is one interrupt pit per endpoint, up to 8 interrupt bits for OUT endpoints and 8 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used at the same time. The corresponding bits in this register are set and cleared when the application sets and clears the bits in the corresponding device endpoint-x interrupt register.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x0000	resd	Kept at its default value.
Bit 23: 16	OUTEPTINT	0x0000	ro	OUT endpoint interrupt bits One OUT endpoint per bit. Bit 16 for OUT endpoint 0, bit 18 for OUT endpoint 2.
Bit 15: 8	Reserved	0x0000	resd	Kept at its default value.
Bit 7: 0	INEPTINT	0x0000	ro	IN endpoint interrupt bits One IN endpoint per bit. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7.

22.6.5.7 OTGFS all endpoints interrupt mask register (OTGFS_DAINTMSK)

When an event occurs on a device endpoint, the device endpoint interrupt mask register works with the device endpoint interrupt register to interrupt the application. However, the device all endpoints interrupt register corresponding to this interrupt is still set.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x0000	resd	Kept at its default value.
Bit 23: 16	OUTEPTMSK	0x0000	rw	OUT EP interrupt mask bits One OUT endpoint per bit. Bit 16 for OUT endpoint 0, bit 18 for OUT endpoint 2. 0: Interrupt masked 1: Interrupt unmasked
Bit 15: 8	Reserved	0x0000	resd	Kept at its default value.
Bit 7: 0	INEPTMSK	0x0000	rw	IN EP interrupt mask bits One IN endpoint per bit. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7. 0: Interrupt masked 1: Interrupt unmasked

22.6.5.8 OTGFS device IN endpoint FIFO empty interrupt mask register (OTGFS_DIEPEMPMSK)

This register works with the TXFE_OTGFS_DIEPINTx register to generate an interrupt.

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x0000	resd	Kept at its default value.
Bit 7: 0	INEPTXFEMSK	0x0000	rw	IN endpoint Tx FIFO empty interrupt mask bits These bits serve as mask bits for the device IN endpoint interrupt register. A transmit FIFO empty interrupt bit per IN endpoint. Bit 0 for IN endpoint 0, bit 7 for IN endpoint 7. 0: Interrupt masked 1: Interrupt unmasked

22.6.5.9 OTGFS device control IN endpoint 0 control register (OTGFS_DIEPCTL0)

This section describes the control IN endpoint 0 control register. Nonzero control endpoint uses registers for endpoints 1-7.

Bit	Name	Reset value	Type	Description
Bit 31	EPTENA	0x0	rw1s	Endpoint enable The application sets this bit to start data transmission on the endpoint 0. The controller clears this bit before generating the following interrupts:

				Endpoint disabled Transfer completed.
Bit 30	EPTDIS	0x0	ro	Endpoint disable The application sets this bit to stop data transmission on an endpoint. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint is enabled.
Bit 29: 28	Reserved	0x0	resd	Kept at its default value.
Bit 27	SNAK	0x0	wo	Set NAK A write to this bit sets the NAK bit of the endpoint. The application can use this bit to control the transmission of NAK handshakes on the endpoint. The controller also sets this bit when a SETUP data packet is received on the endpoint.
Bit 26	CNAK	0x0	wo	Clear NAK A write to this bit clears the NAK bit for the endpoint.
Bit 25: 22	TXFNUM	0x0	rw	TxFIFO number The endpoint 0 can only use FIFO0.
Bit 21	STALL	0x0	rw1s	STALL handshake The application sets this bit, and the controller clears this bit when a SETUP token is received. If a NAK bit, a global non-periodic IN NAK or global OUT NAK bit is set along with this bit, the STALL bit has priority.
Bit 20	Reserved	0x0	resd	Kept at its default value.
Bit 19: 18	EPTYPE	0x0	ro	Endpoint type Set to 0 by hardware for control endpoints.
Bit 17	NAKSTS	0x0	ro	NAK status Indicates the following: 0: The controller is transmitting non-NAK handshakes based on the FIFO status 1: The controller is transmitting NAK handshakes on this endpoint When this bit is set, either by the application or controller, the controller stops transmitting data, even if there are space available in the receive FIFO. The controller always responds to SETUP data packets with an ACK handshake, irrespective of this bit's setting.
Bit 16	Reserved	0x0	resd	Kept at its default value.
Bit 15	USBACEPT	0x0	ro	USB active endpoint This bit is always set to 1, indicating that the control endpoint 0 is always active in all configurations and interfaces.
Bit 14: 2	Reserved	0x0000	resd	Kept at its default value.
Bit 1: 0	MPS	0x0	rw	Applies to IN and OUT endpoints The application uses this bit to program the maximum packet size for the current logical endpoint. 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

22.6.5.10 OTGFS device IN endpoint-x control register (OTGFS_DIEPCTLx) (x=x=1...7, where x is endpoint number)

The application uses this register to control the behavior of the endpoints other than endpoint 0.

Bit	Name	Reset value	Type	Description
Bit 31	EPTENA	0x0	rw1s	Endpoint enable The application sets this bit to start transmitting data on an endpoint. The controller clears this bit before the generation one of the following interrupts on this endpoint: SETUP stage done Endpoint disabled Transfer completed
Bit 30	EPTDIS	0x0	rw1s	Endpoint disable The application sets this bit to stop transmitting data on an endpoint, even if the transfer on that endpoint is incomplete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint enabled set.
Bit 29	SETD1PID/ SETODDFR	0x0	wo	Set DATA1 PID Applies to interrupt/bulk IN endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA1. Set odd frame Applies to synchronous IN endpoints only. Writing to this bit sets the Even/Odd frame to odd frame. 0: Disabled Set DATA1 PID disabled or Do not force odd frame 1: Set DATA1 PID enabled or forced odd frame
Bit 28	SETD0PID/ SETEVENFR	0x0	rw	Set DATA0 PID Applies to interrupt/bulk IN endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA0. Set Even frame Applies to synchronous IN endpoints only. Writing to this bit sets the Even/Odd frame to even frame. 0:Disabled Set DATA0 PID disabled or Do not force even frame 1: Set DATA0PID or set the EOFRNUM to even frame
Bit 27	SNAK	0x0	wo	Set NAK A write to this bit sets the NAK bit for the endpoint. The application uses this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a Transfer completed interrupt or after receiving a SETUP packet. Values: 0: Do not set NAK 1: Set NAK
Bit 26	CNAK	0x0	wo	Clear NAK A write to this bit clears the NAK bit for this endpoint. 0: Not clear NAK 1: Clear NAK
Bit 25: 22	TXFNUM	0x0	rw	TxFIFO number Allocate FIFO number to the corresponding endpoint. A separate FIFO number is allocated to each valid IN endpoint. This bit applies to IN endpoints only.
Bit 21	STALL	0x0	rw	STALL handshake Applies to non-control, non-synchronous IN and OUT endpoints. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, global non-periodic IN NAK bit or global OUT NAK bit is set along with this bit, the STALL bit has priority. Only the application can clear this bit, but the controller never.

				0: Stall all invalid tokens 1: Stall all valid tokens
Bit 20	Reserved	0x0	resd	Kept at its default value.
Bit 19: 18	EPTYPE	0x0	rw	Endpoint type This is the transfer type supported by this logical endpoint. 00: Control 01: Synchronous 10: Bulk 11: Interrupt
Bit 17	NAKSTS	0x0	ro	NAK status Indicates the following status: 0: The controller is sending non-NAK handshakes based on the FIFO status 1: The controller is sending NAK handshakes When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO to accommodate the incoming data packets. For non-synchronous IN endpoints: the controller stops transmitting data on the endpoint, even if there is data pending in the transmit FIFO. For synchronous IN endpoints: the controller sends a zero-length data packet, even if there is space in the transmit FIFO. The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not.
Bit 16	DPID/ EOFRNUM	0x0	ro	Endpoint data PID Applies to interrupt/bulk IN endpoints only. This bit contains the PID of the packet to be transmitted on this endpoint. The application must program the PID of the initial data packet to be received or transmitted on this endpoint, after the endpoint is enabled. The application programs DATA0 or DATA1 PID through the SetD1PID and SetD0PID of this register. 0: DATA0 1: DATA1 Even/Odd frame Applies to synchronous IN endpoints only. Indicates the frame number in which the controller transmits synchronous data on this endpoint. The application must program the even/odd frame number in which it tends to transmit or receive synchronous data through the SETEVNFR and SETODDFR bits in this register. 0: Even frame 1: Odd frame
Bit 15	USBACEPT	0x0	rw	USB active endpoint Indicates whether this endpoint is active in the current configuration and interface. The controller clears this bit for all endpoints except for endpoint 0 after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program the endpoint registers and set this bit. 0: Inactive 1: Active
Bit 14: 11	Reserved	0x0	resd	Kept at its default value.
Bit 10: 0	MPS	0x000	rw	Maximum packet size The application uses this field to set the maximum packet size for the current logical endpoint. The values are in bytes.

22.6.5.11 OTGFS device control OUT endpoint 0 control register (OTGFS_DOEPCTL0)

This section describes the control OUT endpoint 0 control register. Non-zero control endpoints use registers for endpoints 1-7.

Bit	Name	Reset value	Type	Description
Bit 31	EPTENA	0x0	rw1s	Endpoint enable The application sets this bit to start transmitting data on endpoint 0.The controller clears this bit before setting any one of the following interrupts on this endpoint: SETUP stage done Endpoint disabled Transfer completed
Bit 30	EPTDIS	0x0	ro	Endpoint disable The application cannot disable control OUT endpoint 0.
Bit 29: 28	Reserved	0x0	resd	Kept at its default value.
Bit 27	SNAK	0x0	wo	Set NAK A write to this bit sets the NAK bit for this endpoint. The application can use this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a transfer completed interrupt or when a SETUP data packet is received.
Bit 26	CNAK	0x0	wo	Clear NAK A write to this bit clears the NAK for the endpoint.
Bit 25: 22	Reserved	0x0	resd	Kept at its default value.
Bit 21	STALL	0x0	rw1s	STALL handshake The application sets this bit and the controller clears this bit when a SETUP token is received for this endpoint. If a NAK bit, global non-periodic OIT NAK bit is set along with this bit, the STALL bit has priority. The controller always responds to SETUP data packets, regardless of whether this bit is set or not.
Bit 20	SNP	0x0	rw	Snoop mode This bit configures the endpoint to Snoop mode. In this mode, the controller does not check the correctness of OUT packets before transmitting OUT packets to the application memory.
Bit 19: 18	EPTYPE	0x0	ro	Endpoint type Hardware sets this bit to 0 to control endpoint type.
Bit 17	NAKSTS	0x0	ro	NAK status Indicates the followings: 0: The controller is sending non-NAK handshakes based on the FIFO status 1: The controller is sending NAK handshakes When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO. The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not.
Bit 16	Reserved	0x0	resd	Kept at its default value.
Bit 15	USBACEPT	0x1	ro	USB active endpoint This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.
Bit 14: 2	Reserved	0x0000	resd	Kept at its default value.
Bit 1: 0	MPS	0x0	ro	Maximum packet size The maximum packet size of the control OUT endpoint 0 is the same as that of the control IN endpoint 0. 00: 64 bytes 01: 32 bytes 10: 16 bytes 11: 8 bytes

22.6.5.12 OTGFS device control OUT endpoint-x control register (OTGFS_DOEPCTLx) (x= x=1...7, where x if endpoint number)

This application uses this register to control the behavior of all endpoints other than endpoint 0.

Bit	Name	Reset value	Type	Description
Bit 31	EPTENA	0x0	rw1s	Endpoint enable Indicates that the descriptor structure and data buffer for data reception has been configured. The controller clears this bit before setting any one of the following interrupts on this endpoint: – SETUP stage done – Endpoint disabled – Transfer completed
Bit 30	EPTDIS	0x0	ro	Endpoint disable The application sets this bit to stop transmitting data on an endpoint, even if the transfer on that endpoint is incomplete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The controller clears this bit before setting the endpoint disabled interrupt. The application must set this bit only when the endpoint enabled set. 0: No effect 1: Endpoint disabled
Bit 29	SETD1PID/ SETODDFR	0x0	rw	Set DATA1 PID Applies to interrupt/bulk OUT endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA1. Set odd frame Applies to synchronous OUT endpoints only. Writing to this bit sets the Even/Odd frame to odd frame. 0: Disabled Set DATA1 PID disabled or Do not force odd frame 1: Set DATA1 PID enabled or forced odd frame
Bit 28	SETD0PID/ SETEVENFR	0x0	rw	Set DATA0 PID Applies to interrupt/bulk OUT endpoints only. Writing to this bit sets the endpoint data PID bit in this register to DATA0. Set Even frame Applies to synchronous OUT endpoints only. Writing to this bit sets the Even/Odd frame to even frame. 0:Disabled Set DATA0 PID disabled or Do not force even frame 1: Set DATA0PID or set the EOFRNUM to even frame
Bit 27	SNAK	0x0	wo	Set NAK A write to this bit sets the NAK bit for the endpoint. The application uses this bit to control the transmission of NAK handshakes on an endpoint. The controller sets this bit on a Transfer completed interrupt or after receiving a SETUP packet. Values: 0: Do not set NAK 1: Set NAK
Bit 26	CNAK	0x0	wo	Clear NAK A write to this bit clears the NAK bit for the endpoint. 0: Not clear NAK 1: Clear NAK
Bit 25: 22	Reserved	0x0	resd	Kept at its default value.
Bit 21	STALL	0x0	rw	Applies to non-control, non-synchronous IN and OUT endpoints. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, global non-periodic IN NAK bit or global OUT NAK bit is set along with this bit, the STALL bit has priority. Only the application can clear

				this bit, but the controller never.
Bit 20	SNP	0x0	rw	Snoop mode This bit configures the endpoint to Snoop mode. In this mode, the controller does not check the correctness of OUT packets before transmitting OUT packets to the application memory.
Bit 19: 18	EPTYPE	0x0	rw	Endpoint type This is the transfer type supported by this logical endpoint. 00: Control 01: Synchronous 10: Bulk 11: Interrupt
Bit 17	NAKSTS	0x0	ro	NAK status Indicates the followings: 0: The controller is sending non-NAK handshakes based on the FIFO status 1: The controller is sending NAK handshakes When this bit is set (either by the application or the controller), the controller stops receiving any data on an OUT endpoint, even if there is space in the receive FIFO to accommodate the incoming data packets. For non-synchronous IN endpoints: the controller stops transmitting data on the endpoint, even if there is data pending in the transmit FIFO. For synchronous IN endpoints: the controller sends a zero-length data packet, even if there is space in the transmit FIFO. The controller always responds to SETUP data packets with an ACK handshake, regardless of whether this bit is set or not.
Bit 16	DPID/ EOFRNUM	0x0	ro	Endpoint data PID Applies to interrupt/bulk OUT endpoints only. This bit contains the PID of the packet to be transmitted on this endpoint. The application must program the PID of the initial data packet to be received or transmitted on this endpoint, after the endpoint is enabled. The application programs DATA0 or DATA1 PID through the SetD1PID and SetD0PID of this register. 0: DATA0 1: DATA1 Even/Odd frame Applies to synchronous OUT endpoints only. Indicates the frame number in which the controller transmits synchronous data on this endpoint. The application must program the even/odd frame number in which it tends to transmit or receive synchronous data through the SETEVNFR and SETODDFR bits in this register. 0: Even frame 1: Odd frame
Bit 15	USBACEPT	0x0	rw	USB active endpoint Indicates whether this endpoint is active in the current configuration and interface. The controller clears this bit for all endpoints except for endpoint 0 after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program the endpoint registers and set this bit. 0: Inactive 1: Active
Bit 14: 11	Reserved	0x0	resd	Kept at its default value.
Bit 10: 0	MPS	0x000	rw	Maximum packet size The application uses this field to set the maximum packet size for the current logical endpoint. The values are in bytes.

22.6.5.13 OTGFS device IN endpoint-x interrupt register (OTGFS_DIEPINTx) (x=0...7, where x if endpoint number)

This register indicates the status of an endpoint when USB and AHB-related events occurs, as shown in Figure 22-2 When the IEPINT bit of the OTGFS_GINTSTS register is set, the application must first read the OTGFS_DAINT register to get the exact endpoint number in which the event occurs, before reading the endpoint interrupt registers. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTGFS_DAINT and OTGFS_GINTST registers.

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value.
Bit 7	TXFEMP	0x0	ro	Transmit FIFO empty This interrupt is generated when the transmit FIFO for this endpoint is half or completely empty. The half or completely empty status depends on the transmit FIFO empty level bit in the controller AHB configuration register.
Bit 6	INEPTNAK	0x0	rw1c	IN endpoint NAK effective This bit can be cleared by writing 1 to the CNAK bit in the DIEPCTLx register. This interrupt indicates that the IN endpoint NAB bit set by the application has taken effect. This interrupt does not guarantee that a NAK handshake is sent on the USB line. A STALL bit has priority over a NAK bit. This bit applies to the scenario only when the endpoint is enabled.
Bit 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	INTKNTXFEMP	0x0	rw1c	N token received when TxFIFO is empty Indicates that an IN token was received when the associated transmit FIFO (periodic or non-periodic) was empty. An interrupt is generated on the endpoint for which an IN token was received.
Bit 3	TIMEOUT	0x0	rw1c	Timeout condition Applies to control IN endpoints only. This bit indicates that the controller has detected a timeout condition for the last IN token on this endpoint.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	EPTDISD	0x0	rw1c	Endpoint disabled interrupt This bit indicates that the endpoint is disabled according to the application's request.
Bit 0	XFERC	0x0	rw1c	Transfer completed interrupt Indicates that the programmed transfers are complete on the AHB and on the USB for this endpoint.

22.6.5.14 OTGFS device OUT endpoint-x interrupt register (OTGFS_DOEPINTx) (x=0...7, where x if endpoint number)

This register indicates the status of an endpoint with respect to USB and AHB-related events, as shown in Figure 22-2. When the OEPINT bit of the OTGFS_GINTSTS register is set, the application must first read the OTGFS_DAINT register to get the exact endpoint number in which the event occurs, before reading the endpoint interrupt registers. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTGFS_DAINT and OTGFS_GINTST registers.

Bit	Name	Reset value	Type	Description
Bit 31: 7	Reserved	0x0000001	resd	Kept at its default value.
Bit 6	B2BSTUP	0x0	rw1c	Back-to-back SETUP packets received Indicates that more than three back-to-back SETUP packets are received.
Bit 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	OUTTEPD	0x0	rw1c	OUT token received when endpoint disabled Applies to control OUT endpoints only. Indicates that an OUT token was received when the endpoint has not yet been enabled. An interrupt is generated on the endpoint for which an OUT token was received.

Bit 3	SETUP	0x0	rw1c	SETUP phase done Applies to control OUT endpoints only. Indicates that the SETUP stage for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. Upon this interrupt, the application can decode the received SETUP data packets.
Bit 2	Reserved	0x0	resd	Kept at its default value.
Bit 1	EPTDISD	0x0	rw1c	Endpoint disabled interrupt Indicates that the endpoint is disabled according to the application's request.
Bit 0	XFERC	0x0	rw1c	Transfer completed interrupt Indicates that the programmed transfers are complete on the AHB and on the USB for this endpoint.

22.6.5.15 OTGFS device IN endpoint 0 transfer size register (OTGFS_DIEPTSIZ0)

The application must set this register before enabling endpoint 0. Once the endpoint 0 is enabled using the endpoint enable pin in the device endpoint 0 control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

Bit	Name	Reset value	Type	Description
Bit 31: 21	Reserved	0x000	resd	Kept at its default value.
Bit 20: 19	PKTCNT	0x0	rw	Packet count Indicates the total number of USB packets that constitute the transfer size of data for the endpoint 0. This field is decremented every time a packet is read from the transmit FIFO (maximum packet size or short packet)
Bit 18: 7	Reserved	0x000	resd	Kept at its default value.
Bit 6: 0	XFERSIZE	0x00	rw	Transfer size Indicates the transfer size (in bytes) for the endpoint 0. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint at the end of each packet. The controller decrements this field every time a packet from the external memory is written to the transmit FIFO.

22.6.5.16 OTGFS device OUT endpoint 0 transfer size register (OTGFS_DOEPTSIZ0)

The application must set this register before enabling endpoint 0. Once the endpoint 0 is enabled using the endpoint enable pin in the device endpoint 0 control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30: 29	SUPCNT	0x0	rw	SETUP packet count Indicates the number of back-to-back SETUP data packets the endpoint can receive. 01: 1 packet 10: 2 packets 11: 3 packets
Bit 28: 20	Reserved	0x000	resd	Kept at its default value.
Bit 19	PKTCNT	0	rw	Packet count This bit is decremented to 0 after a packet is written to the receive FIFO.
Bit 18: 7	Reserved	0x000	resd	Kept at its default value.
Bit 6: 0	XFERSIZE	0x00	rw	Transfer size Indicates the transfer size (in bytes) for the endpoint 0. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The controller decrements this field every time a packet from the external memory is written to the transmit FIFO.

				The controller decrements this field every time a packet from the receive FIFO is written to the external memory.

22.6.5.17 OTGFS device IN endpoint-x transfer size register (OTGFS_DIEPTSIZx) (x=1...7, where x is endpoint number)

The application must set this register before enabling endpoint x. Once the endpoint x is enabled using the endpoint enable pin in the device endpoint x control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30: 29	MC	0x0	rw	Multi count For periodic IN endpoints, this field indicates the number of packets to be transmitted on the USB for each frame. The controller uses this field to calculate the data PID transmitted on synchronous IN endpoints. 01: 1 packet 10: 2 packets 11: 3 packets
Bit 28: 19	PKTCNT	0x000	rw	Packet count Indicates the total number of USB packets (data transfer size on the endpoint) this field is decremented every time a packet is read from the transmit FIFO (maximum packet size and short packet).
Bit 18: 0	XFERSIZE	0x00000	rw	Transfer Size Indicates the transfer size (in bytes) for the current endpoint. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The controller decrements this field every time a packet from the external memory is written to the transmit FIFO.

22.6.5.18 OTGFS device IN endpoint transmit FIFO status register (OTGFS_DTXFSTSx) (x=1...7, where x is endpoint number)

This is a read-only register containing the free space information for the device IN endpoint transmit FIFO.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	INEPTXFSAV	0x0200	ro	IN endpoint TxFIFO space available Indicates the amount of free space in the endpoint transmit FIFO. Values are in terms of 32-bit words. 0x0: Endpoint transmit FIFO is full 0x1: 1 word available 0x02: 2 words available 0xn: n words available (0 < n < 512) 0x200: Remaining 512 words Others: Reserved

22.6.5.19 OTGFS device OUT endpoint-x transfer size register (OTGFS_DOEPTSIZx) (x=1...7, where x is endpoint number)

The application must set this register before enabling endpoint x. Once the endpoint x is enabled using the endpoint enable pin in the device endpoint x control register, the controller modifies this register. The application can only read this register as long as the controller clears the endpoint enable bit.

Bit	Name	Reset value	Type	Description
Bit 31	Reserved	0x0	resd	Kept at its default value.
Bit 30: 29	RXDPID	0x0	ro	Received data PID Applies to synchronous OUT endpoints only. This is the data PID received in the last packet. 00: DATA0 01: DATA2 10: DATA1 11: MDATA SETUP packet count Applies to synchronous OUT endpoints only. Indicates the number of back-to-back SETUP data packets the endpoint can receive. 01: 1 packet 10: 2 packets 11: 3 packets
Bit 28: 19	PKTCNT	0x000	rw	Packet count Indicates the number of USB packets transmitted on the endpoint. This field is decremented every time a packet is written to the receive FIFO (maximum packet size and short packet)
Bit 18: 0	XFERSIZE	0x00000	rw	Transfer size Indicates the transfer size (in bytes) for the current endpoint. The controller interrupts the application when the transfer size becomes 0. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The controller decrements this field every time a packet is read from the receive FIFO and written to the external memory.

22.6.6 Power and clock control registers

22.6.6.1 OTGFS power and clock gating control register (OTGFS_PCGCCTL)

This register is available in host and device modes.

Bit	Name	Reset value	Type	Description
Bit 31: 5	Reserved	0x0000000	resd	Kept at its default value.
Bit 4	SUSPENDM	0x0	ro	PHY suspend Indicates that the PHY has been suspended.
Bit 3: 1	Reserved	0x0	resd	Kept at its default value.
Bit 0	STOPPCLK	0x0	rw	Stop PHY clock The application uses this bit to stop PHY clock when the USB is suspended, session is invalid or device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

23 HICK auto clock calibration (ACC)

23.1 ACC introduction

HICK auto clock calibration (HICK ACC), which uses the SOF signal (1 ms of period) generated as a reference signal, implements the sampling and calibration for the HICK clocks.

The main purpose of this module is to provide a clock of 48MHz±0.25% for the USB device.

It is able to make the calibrated frequency as close to the target frequency as possible by means of “cross and return” algorithm.

23.2 Main features

- Programmable center frequency
- Programmable boundary frequency that triggers calibration function
- Center frequency precision ±0.25%
- Status detection flags
  - Calibration ready flag
- Error detection flags
  - Reference signal lost error flag
- Two interrupt source flag
  - Calibration ready flag
  - Reference signal lost error flag
- Two calibration modes: coarse calibration and fine calibration

23.3 Interrupt requests

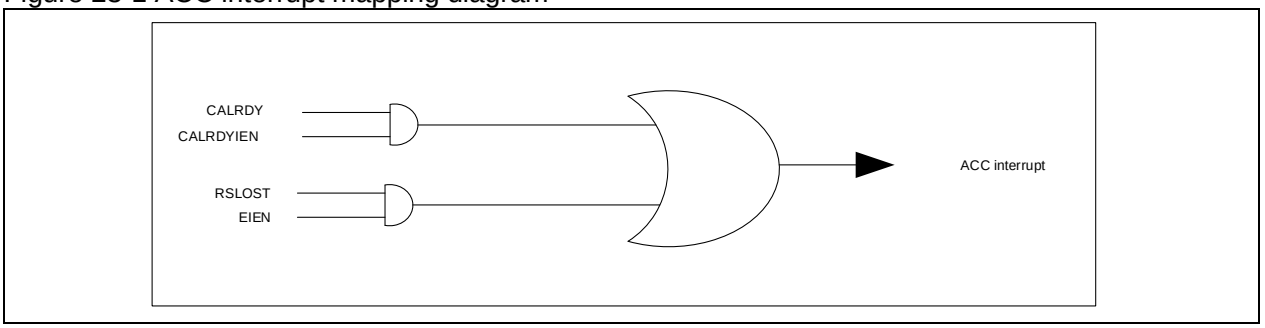
Table 23-1 ACC interrupt requests

Interrupt event	Event flag	Enable bit
Calibration ready	CALRDY	CALRDYIEN
Reference signal lost	RSLOST	EIEN

ACC interrupt events are linked to the same interrupt vector (see Figure 23-1). Interrupt events include:

- During calibration process: When the calibration gets ready or reference signal lost occurs, the corresponding interrupt will be generated if the corresponding enable bit is enabled.

Figure 23-1 ACC interrupt mapping diagram



23.4 Functional description

Auto clock calibration (HICK ACC), which uses the SOF signal (1 ms of period) generated as a reference signal, implements the sampling and calibration for the HICK clocks. In particular, the HICK clock frequency can be calibrated to a precision of ±0.25% so as to meet the needs of the high-precision clock applications such as USB.

The signals of the module are connected to the CRM and HICK inside the microcontroller instead of being connected to the pins externally.

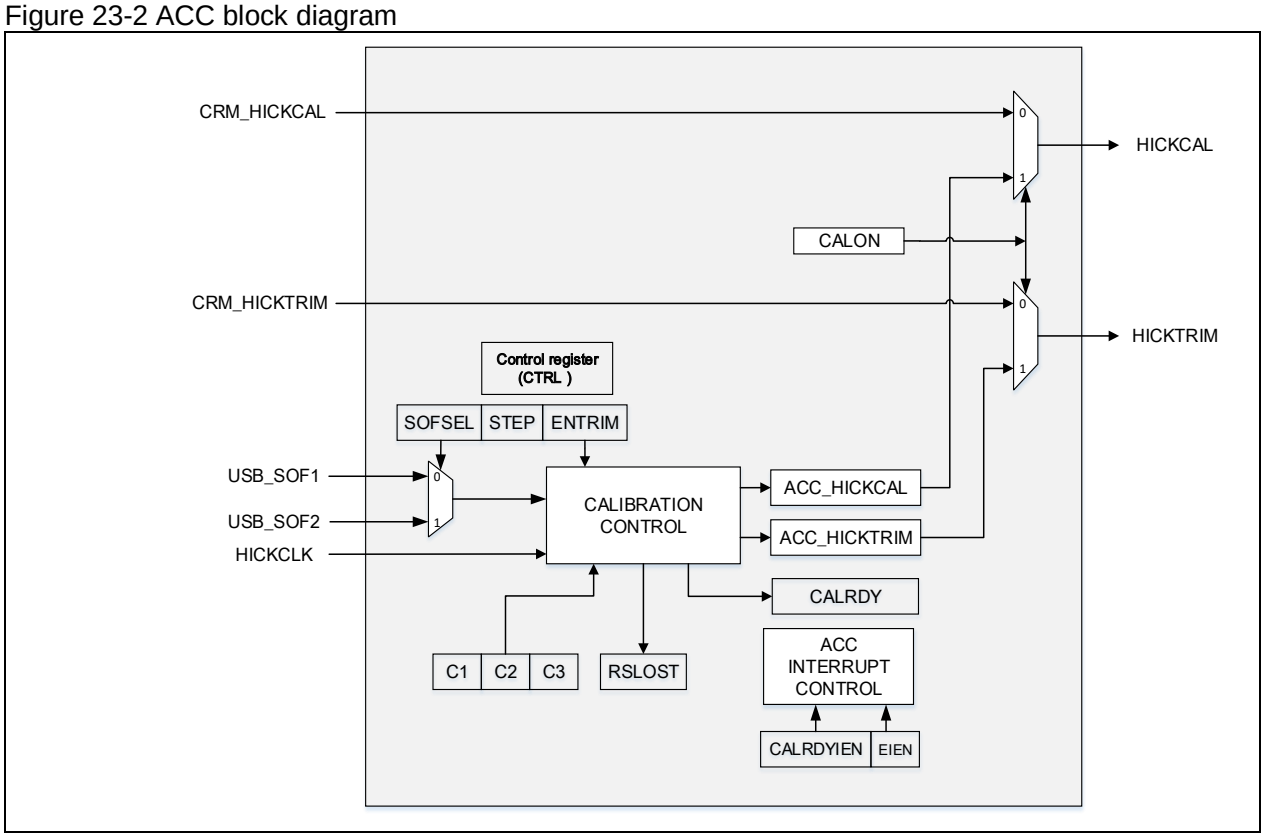
- CRM_HICKCAL: the HICKCAL bit in the CRM module. This signal is used to calibrate the HICK in bypass mode. The value is defined by the HICKCAL[7: 0] in the CRM_CTRL register.
- CRM_HICKTRIM: the HICKTRIM bit in the CRM module. This signal is used to calibrate the HICK in bypass mode. The value is defined by the HICKTRIM[5: 0] in the CRM_CTRL register.

The default value of the HICK is 32, which can be calibrated to 8MHz±0.25%. The HICK frequency can be adjusted by 20kHz (design value) each time when the CRM_HICKTRIM value changes. In other words, the HICK output frequency will increase by 20kHz each time the CRM_HICKTRIM value is decremented by one; the HICK output frequency will reduce by 20kHz each time the CRM_HICKTRIM value is decremented by one.

- USB_SOF: USB Start-of-Frame signal given by the USB device. Its high-level width is 12 system clock cycles, a pulse signal of 1 ms.
- HICKCLK: HICK clock. The original HICK output frequency is 48MHz, but the sampling clock used by the HICK calibration module is frequency divider (1/6) clock, about 8MHz.
- HICKCAL: HICK module calibration signal. For the HICK clock after frequency division (1/6), the HICK clock frequency will change by 40KHz (design value) each time the HICKCAL changes, which is positively correlated. In other words, the HICK clock frequency will increase by 40KHz (design value) each time the HICKCAL is incremented by one; the HICK clock frequency will reduce by 40KHz each time the HICKCAL is decremented by one.
- HICKTRIM: HICK module calibration signal. For the HICK clock after frequency division (1/6), the HICK clock frequency will change by 20KHz (design value) each time the HICKCAL changes, which is positively correlated.

Refer to Section 23.6 for more information about the bit definition in the registers.

Figure 23-2 ACC block diagram

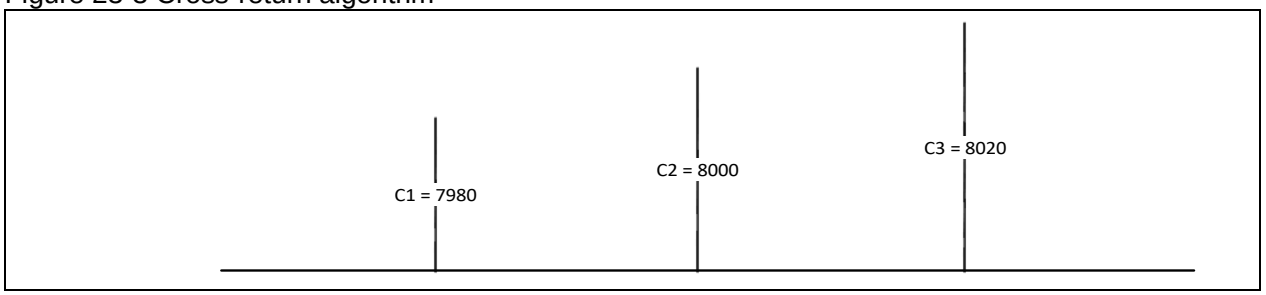


23.5 Principle

USB_SOF period signal: 1ms of period must be accurate, which is a prerequisite of the normal operation of an auto calibration module.

cross-return algorithm: This is used to calculate a calibration value closest to the theoretic value. In theory, the actual frequency after calibration can be adjusted to be within an accuracy range of about 0.5 steps from the target frequency (8MHz).

Figure 23-3 Cross-return algorithm



From the above figure, auto calibration function will adjust the HICKCAL or HICKTRIM according to the specified step as soon as the condition for triggering auto calibration is reached.

Cross:

If the auto calibration condition is met, the actual sampling data in the first 1ms period will be either less than C2, or greater than C2.

When this value is less than C2, the auto calibration module will start increasing either the HICKCAL or HICKTRIM according to the step definition until the actual sampling value is greater than C2. In this way, the actual value will cross over C2 from small to large.

When this value is greater than C2, the auto calibration module will start decrease either the HICKCAL or HICKTRIM according to the step definition until the actual sampling value become less than C1. In this way, the actual value will cross over C2 from large to small.

Return:

After cross operation is completed, the actual value closest to C2 can be obtained by comparing the difference (calculated as absolute value) between the actual sampling value and C2 before and after crossing C2 so as to get the best calibration value HICKCAL or HICKTRIM.

If the difference after crossing is less than the one before crossing C2, the calibration value after crossing prevails, and stops the calibration process until the next condition for auto calibration appears.

If the difference after crossing is greater than the one before crossing C2, the calibration value before crossing prevails, and it will return by one step to the one before crossing, and stops the calibration process until the next condition for auto calibration appears.

According to the cross-return strategy, in theory, it is possible to get the frequency accuracy that is 0.5 steps away from the center frequency.

Four conditions for enabling auto calibration function are as follows:

1. The rising edge of the CANLON (from 0 to 1)
2. When CALON=1, reference signal is lost and restored
3. When the sample counter is less than C1
4. When the sample counter is greater than C3

Even though the sampling counter is between C1 and C3, at the rising edge the CANLON, the auto calibration module can also be activated so that the HICK frequency can be adjusted to be within a range of 0.5 steps of the center frequency as soon as the CANLON is enabled.

Under one of the above-mentioned circumstances, the HICK frequency can be calibrated to be within 0.5 steps of the center frequency. To achieve the best calibration accuracy, it is recommended to remain step as 1 (default value). If the step is set to 0, either HICKCAL or HICKTRIM will not be able to be calibrated.

23.6 Register description

Refer to the list of abbreviations used in register descriptions.

These peripheral registers must be accessed by words (32 bits).

23.6.1 ACC register map

Table 23-2 ACC register map and reset values

Register	Offset	Reset value
ACC_STS	0x00	0x0000 000
ACC_CTRL1	0x04	0x0000 0100
ACC_CTRL2	0x08	0x0000 2080
ACC_C1	0x0C	0x0000 1F2C
ACC_C2	0x10	0x0000 1F40
ACC_C3	0x14	0x00000 1F54

23.6.2 Status register (ACC_STS)

Bit	Name	Reset value	Type	Description
Bit 31: 9	Reserved	0x0000000	resd	Kept at its default value.
Bit 1	RSLOST	0x0	ro	Reference Signal Lost 0: Reference Signal is not lost 1: Reference Signal is lost Note: During the calibration, when the sample counter of the calibration module is twice that of C2, if a SOF reference signal is not detected, it means that the reference signal is lost. The internal statue machine will move to the idle state unless another SOF signal is detected; otherwise, the internal clock sample counter remains 0. The RSLOST bit is immediately cleared after the CALON bit is cleared or when the RSLOST is written with 0. Reference signal detection occurs only when CALON=1.
Bit 0	CALRDY	0x0	ro	Internal high-speed clock calibration ready 0: Internal 8MHz oscillator calibration is not ready 1: Internal 8MHz oscillator calibration is ready Note: This bit is set by hardware to indicate that internal 8MHz oscillator has been calibrated to the frequency closest to 8MHz. The CALRDY is immediately cleared after the CALON bit is cleared or when the CALRDY is written with 0.

23.6.3 Control register 1 (ACC_CTRL1)

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Forced by hardware to 0
Bit 11: 8	STEP	0x1	rw	Calibrated step This field defines the value after each calibration. Note: It is recommended to set the step bit in order to get a more accurate calibration result. While ENTRIM=0, only the HICKCAL is calibrated. If the step is incremented or decremented by one, the HICKCAL will be incremented or decremented by one accordingly, and the HICK frequency will increase or decrease by 40KHz (design value). This is a positive relationship. While ENTRIM=1, only the HICKTRIM is calibrated. If the step is incremented or decremented by one, the HICKTRIM will be incremented or decremented by one accordingly, and the HICK frequency will increase or decrease by 20KHz (design value). This is a positive relationship.
Bit 7: 6	Reserved	0x0	rw	Forced by hardware to 0
Bit 5	CALRDYIEN	0x0	rw	CALRDY interrupt enable This bit is set or cleared by software.

				0: Interrupt generation disabled 1: ACC interrupt is generated when CALRDY=1 in the ACC_STS register
Bit 4	EIEN	0x0	rw	RSLOST error interrupt enable This bit is set or cleared by software. 0: Interrupt generation disabled 1: ACC interrupt is generated when RSLOST=1 in the ACC_STS register
Bit 3	Reserved	0x0	rw	Forced by hardware to 0
Bit 2	SOFSEL	0x0	rw	SOF select This bit is set or cleared by software. 0: USB1 as SOF signal source 1: USB2 as SOF signal source
Bit 1	ENTRIM	0x0	rw	Enable trim This bit is set or cleared by software. 0: HICKCAL is calibrated. 1: HICKTRIM is calibrated. Note: It is recommended to set ENTRIM=1 in order to get higher calibration accuracy.
Bit 0	CALON	0x0	rw	Calibration on This bit is set or cleared by software. 0: Calibration disabled 1: Calibration enabled, and starts searching for a pulse on the USB_SOF. Note: This module cannot be used without the USB_SOF reference signal. If there are no requirements on the accuracy of the HICK clock, it is unnecessary to enable this module.

23.6.4 Control register 2 (ACC_CTRL2)

Bit	Name	Reset value	Type	Description
Bit 31: 14	Reserved	0x00000	resd	Forced to 0 by hardware
Bit 13: 8	HICKTRIM	0x20	ro	Internal high-speed auto clock trimming This field is read only, but not written. Internal high-speed clock is adjusted by ACC module, which is added to the ACC_HICKCAL[7: 0] bit. These bits allow the users to input a trimming value to adjust the frequency of the HICKRC oscillator according to the variations in voltage and temperature. The default value is 32, which can trim the HICK to 8MHz±0.25%. The trimming value is 20kHz (design value) between two consecutive ACC_HICKTRIM steps.
Bit 7: 0	HICKCAL	0x80	ro	Internal high-speed auto clock calibration This field is read only, but not written. Internal high-speed clock is adjusted by ACC module. These bits allow the users to input a trimming value to adjust the frequency of the HICKPC oscillator according to the variations in voltage and temperature. The default value is 128, which can trim the HICK to 8MHz±0.25%. The trimming value is 40kHz (design value) between two consecutive ACC_HICKCAL steps.

23.6.5 Compare value 1 (ACC_C1)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Forced to 0 by hardware
Bit 15: 0	C1	0x1F2C	rw	Compare 1 This value is the lower boundary for triggering calibration, and its default value is 7980. When the number of clocks sampled by ACC in 1ms period is less than or equal to C1, auto calibration is triggered automatically. When the actual sampling value (number of clocks in 1ms) is greater than C1 but less than C3, auto calibration is not enabled.

23.6.6 Compare value 2 (ACC_C2)

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Forced to 0 by hardware
Bit 15: 0	C2	0x1F40	rw	Compare 2 This value defines the number of clocks sampled for 8MHz (ideal frequency) clock in 1ms period , and its default value is 8000 (theoretical value). As a center point of cross-return strategy, this value is used to calculate the calibration value closest to the theoretical value. In theory, the actual frequency after calibration can be trimmed to be within an accuracy of 0.5 steps from the target frequency (8MHz).

23.6.7 Compare value 3 (ACC_C3)

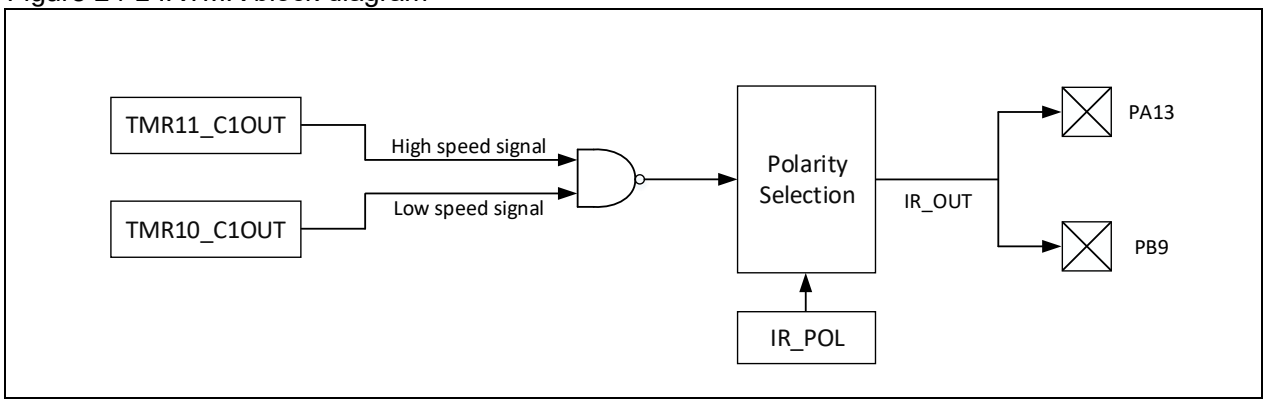
Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Forced to 0 by hardware
Bit 15: 0	C3	0x1F54	rw	Compare 3 This value is the upper boundary for triggering calibration. When the number of clock sampled by ACC in 1ms period is greater than or equal to C3, auto calibration is triggered automatically. When the actual sampling value (number of clocks in 1ms period) is greater than C1 but less than C3, auto calibration is not enabled.

24 Infrared timer (IRTMR)

The IRTMR (Infrared Timer) is used to generate the IR_OUT signal that drives the infrared LED so as to achieve infrared control.

The IR_OUT signals consists of a low-frequency modulation envelope and high-frequency carrier signals. The low-frequency modulation envelope signal comes from TMR10_C1OUT, while the high-frequency carrier signal is provided by the TMR11_C1OUT register. The IR_POL bit in the SCFG_CFG1 register controls whether the IR_OUT output is reversed or not. The IR_OUT signal is output through multiplexed function via PB9 or PA13 (multiplexed mode needs to be configured in advance).

Figure 24-1 IRTMR block diagram



25 External memory controller (XMC)

25.1 XMC introduction

XMC peripheral block can translate the AHB signals into the external memory signals and vice versa.The supported external memories include SRAM, NOR Flash, PSRAM and SDRAM.

25.2 XMC main features

NOR/PSRAM has the following features:

- Chip-select signal supports 4 external memories, each of which has their own control register
- Support access to static memory devices, including
  - Static random access memory (SRAM)
  - NOR Flash
  - PSRAM
- 8-bit or 16-bit wide memory
- Various timing mode selection
  - Two modes with the same timings for read and write
  - Four modes with different timings for read and write
  - Multiplexed address/data mode
  - Synchronous mode
- Programmable timing control registers
- Translate the AHB data size into the appropriate external memory data size

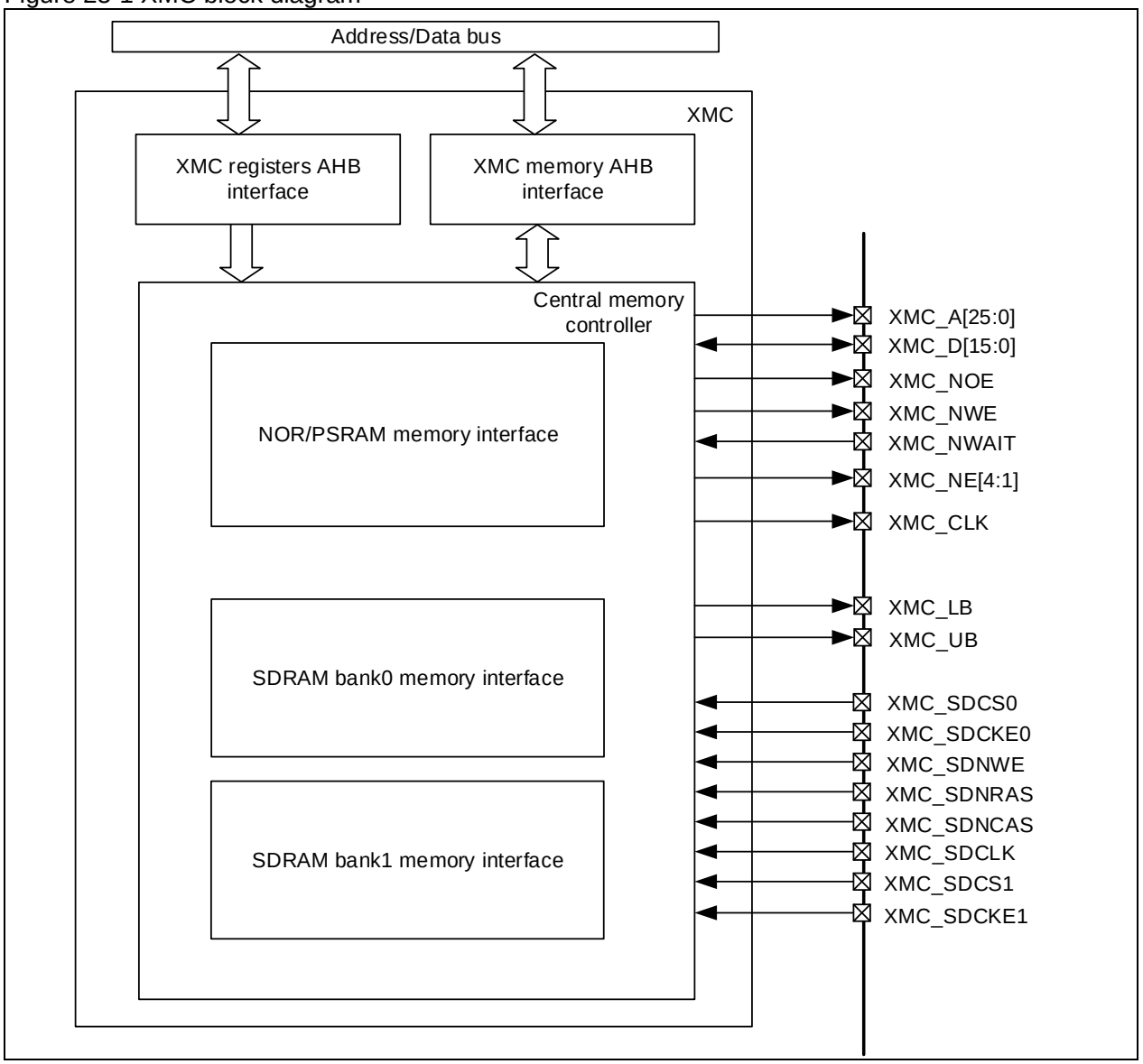
SDRAM card has the following features:

- Connected to 2 independent SDRAM devices externally
- 8-bit or 16-bit wide SDRAM data bus
- Supports up to 13-bit address row, 11-bit address column and 4 internal memory areas (4x16Mx16bit :128 MB and 4x16Mx8bit: 64 MB)
- SDRAM clock sources: HCLK, HCLK/2, HCLK/3 or HCLK/4
- Supports AHB byte, halfword and word accesses
- Programmable SDRAM access timing parameters
- Automatic row and memory boundary management
- Multi-memory area ping-pong access
- Automatic refresh operation of software-programmable refresh rate
- Supports self-refresh mode
- Supports power-down mode
- SDRAM power-on initialization through software
- CAS latency can be configured 1/2/3 SDRAM clock cycles

25.3 XMC architecture

25.3.1 Block diagram

Figure 25-1 XMC block diagram



While interfacing to the external memory, NOR/PSRAM, NAND, PC card and SDRAM uses different pins respectively, as shown in Table 25-1 and Table 25-2.

Table 25-1 NOR/PSRAM pins

Pin name	I/O	Description
XMC_CLK	Output	Clock
XMC_NE[x], x=1,4	Output	Chip select
XMC_NADV	Output	Address latch or address valid (NL) signal
XMC_A[x]	Output	Address bus
XMC_NOE	Output	Output enable signal
XMC_NWE	Output	Write enable signal
XMC_LB and XMC_UB	Output	Byte select signal
XMC_D[15: 0]	Read input/write output	Data bus/multiplexed address data
XMC_NWAIT	Input	Wait signal

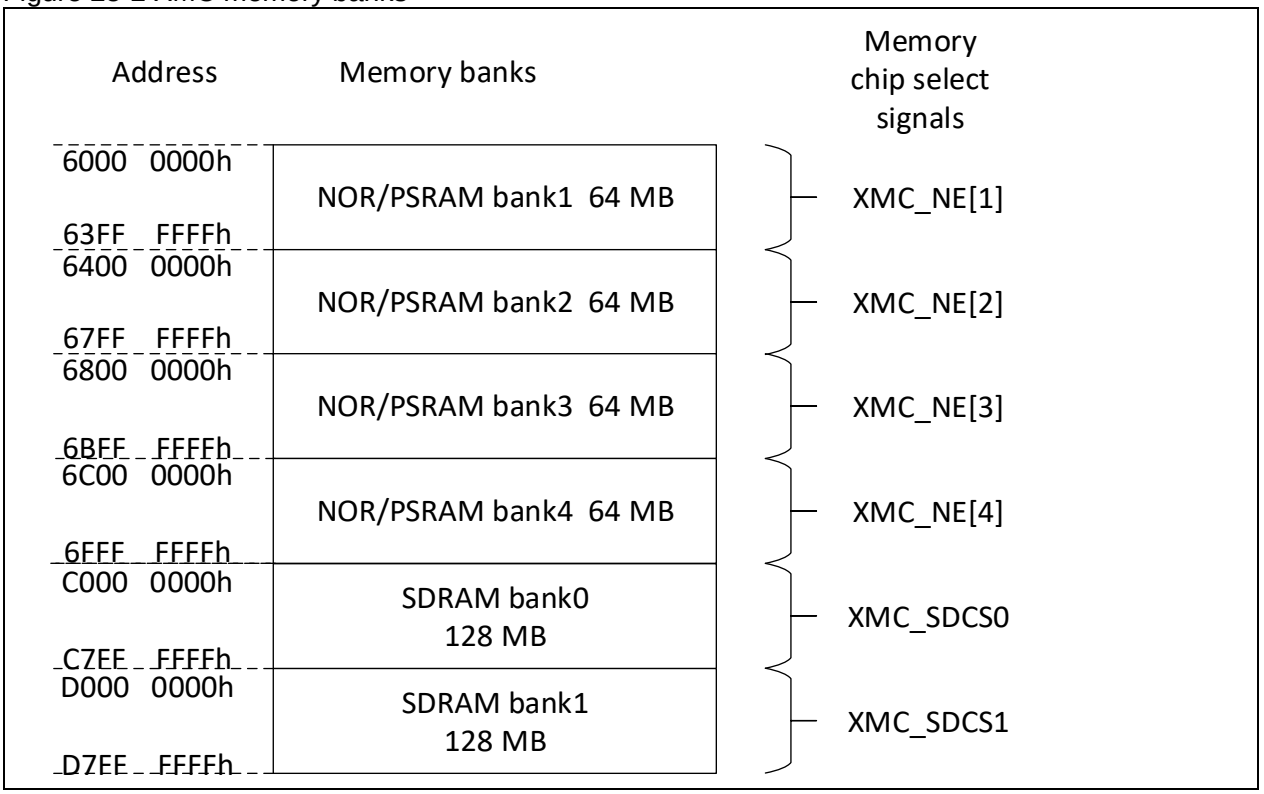
Table 25-2 SDRAM pins

Pin name	I/O	Description
XMC_SDCKE0	Output	DEVICE0 clock enable
XMC_SDCKE1	Output	DEVICE1 clock enable
XMC_SDCS0	Output	DEVICE0 chip-select
XMC_SDCS1	Output	DEVICE1 chip-select
XMC_SDCLK	Output	Clock signal
XMC_SDNRAS	Output	Row select
XMC_SDNCAS	Output	Column select
XMC_SDNWE	Output	Write enable
XMC_LB, XMC_UB	Output	Byte select
XMC_A[12:0]	Output	Address bus
XMC_A[14]	Output	BANK address low
XMC_A[15]	Output	B BANK address high
XMC_D[15:0]	Read input/write output	Data bus

25.3.2 Address mapping

XMC address is divided into multiple memory banks, as shown below.

Figure 25-2 XMC memory banks



Some HADDR bits are used to select which bank to access to, as shown in Table 25-3.

Table 25-3 Memory bank selection

HADDR[31：28]	**HADDR[27：26]**
0110：NOR/PSRAM	00：bank1
	01：bank2
	10：bank3
	11：bank4
HADDR[31：28]	**HADDR[26:0]**
1100：SDRAM BANK1	BANK and row/column address mapping
HADDR[31：28]	**HADDR[26:0]**
1101：SDRAM BANK2	BANK and row/column address mapping

Table 25-4 8-bit SDRAM address mapping

Row size	HADDR (**AHB internal address line**) 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0			
11-bit	Reserved	Bank [1:0]	Row[10:0]	Column[7:0]
	Reserved	Bank [1:0]	Row[10:0]	Column[8:0]
	Reserved	Bank [1:0]	Row[10:0]	Column[9:0]
	Reserved	Bank [1:0]	Row[10:0]	Column[10:0]
12-bit	Reserved	Bank [1:0]	Row[11:0]	Column[7:0]
	Reserved	Bank [1:0]	Row[11:0]	Column[8:0]
	Reserved	Bank [1:0]	Row[11:0]	Column[9:0]
	Reserved	Bank [1:0]	Row[11:0]	Column[10:0]
13-bit	Reserved	Bank [1:0]	Row[12:0]	Column[7:0]
	Reserved	Bank [1:0]	Row[12:0]	Column[8:0]
	Reserved	Bank [1:0]	Row[12:0]	Column[9:0]
	Reserved	Bank [1:0]	Row[12:0]	Column[10:0]

Table 25-5 16-bit SDRAM address mapping

Row size	HADDR (AHB internal address line) 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
11-bit	Reserved | Bank [1:0] | Row[10:0] | Column[7:0]
	Reserved | Bank [1:0] | Row[10:0] | Column[8:0]
	Reserved | Bank [1:0] | Row[10:0] | Column[9:0]
	Reserved | Bank [1:0] | Row[10:0] | Column[10:0]
12-bit	Reserved | Bank [1:0] | Row[11:0] | Column[7:0]
	Reserved | Bank [1:0] | Row[11:0] | Column[8:0]
	Reserved | Bank [1:0] | Row[11:0] | Column[9:0]
	Reserved | Bank [1:0] | Row[11:0] | Column[10:0]
13-bit	Reserved | Bank [1:0] | Row[12:0] | Column[7:0]
	Reserved | Bank [1:0] | Row[12:0] | Column[8:0]
	Reserved | Bank [1:0] | Row[12:0] | Column[9:0]
	Reserved | Bank [1:0] | Row[12:0] | Column[10:0]

25.4 NOR/PSRAM

NOR/PSRAM offers multiple access modes with different timings to drive multiple memories including NOR Flash, SRAM, PSRAM or Cellular RAM.

There are two banks, from bank 1 to bank 4, with independent control registers. They can be accessed with different timings and different chip-select signals.

25.4.1 Operating mode

Pin function:

Pin signals vary from external memory to external memory. Table 24-8 lists typical pin signals.

Table 25-6 Pin signals for NOR and PSRAM

XMC pin name	NOR Flash	PSRAM
XMC_CLK	Clock (synchronous mode)	Clock (synchronous mode)
XMC_NE[x]	Chip-select	Chip-select
XMC_NADV	Address latch or valid address signal	Address latch or valid address signal
XMC_A[25:0]	Address bus	Address bus
XMC_NOE	Output enable	Output enable
XMC_NWE	Write enable	Write enable
XMC_LB, XMC_UB	Without using XMC_LB, XMC_UB	XMC_LB: lower byte XMC_UB: upper byte
XMC_D[15: 0]	Data bus multiplexed address data bus (multiplex and synchronous mode)	Data bus multiplexed address data bus (multiplex and synchronous mode)
XMC_NWAIT	NOR Flash wait request	PSRAM wait request

Note: If the memory data size is 8-bit, the typical data bus is XMC_D[7: 0].

Access address

The upper bytes of the HADDR bit are used to select a memory bank while the lower bytes to data memory address. HADDR is a byte address whereas the XMC supports memory addressed in words or half words. Address translation between them is shown in Table 25-7. As long as read/write access to a specific address occurs, the XMC will enable chip-select signals and write/read operation to the external memories according to the HADDR bit.

Table 25-7 Address translation between HADDR and external memory

External memory data width	Address connection	Accessible maximum memory space (bits)
8-bit	HADDR[25: 0] is linked to XMC_A[25: 0]. In multiplexed and synchronous mode, HADDR[15: 0] is connected to XMC_D[15: 0] during address latch period.	64 Mbyte x8 =512 Mbit
16-bit	HADDR[26: 1] is connected to XMC_A[25: 0]. In multiplexed and synchronous mode, HADDR[16: 1] is connected to XMC_D[15: 0] during address latch period	(64 Mbyte x 16)/2=512 Mbit

Data access

In case that the AHB data width is not equal to that of the memories, the XMC will make appropriate arrangement according to the typical signals of the external memories. Table 25-8 lists the operation modes supported by XMC.

Table 25-8 Data access width vs. external memory data width

Memory	Mode	AHB data width	Memory width	Description
SRAM	Asynchronous read/write	8/16/32	8	One-time access, or split into 2 or 4 accesses
	Asynchronous read/write	8/16/32	16	XMC_LB and XMC_UB, One-time, or split into two access
PSRAM	Asynchronous read	8	16	
	Asynchronous write	8	16	Use XMC_LB and XMC_UB
	Asynchronous read/write	16	16	
	Asynchronous read/write	32	16	Split into 2 XMC accesses
	Synchronous write	8	16	Use XMC_LB and XMC_UB
	Synchronous read/write	16	16	
	Synchronous read/write	32	16	Split into 2 XMC accesses
NOR Flash	Asynchronous read	8	16	
	Asynchronous read/write	16	16	
	Asynchronous read/write	32	16	Split into 2 XMC accesses
	Synchronous read	16	16	
	Synchronous read	32	16	Split into 2 XMC accesses

25.4.2 Access mode

The XMC offers various access modes. Each access mode is operated based on the timing parameters, as shown in Table 25-9.Users can perform programming operations according to the specifications of the external memory and application needs.

Access modes available in the XMC:

- Read/write operation with the same timings: Mode 1 and Mode 2
- Read/write operation with different timings: Mode A, B, C and D
- Multiplexed address data lines
- Clock-based synchronous mode

Table 25-9 NOR/PSRAM parameter registers

Parameter register	Function	Access mode	Unit
ADDRST	Address set-up time	1, 2, A, B, C, D and multiplexed	HCLK cycle
ADDRHT	Address-hold time	D and multiplexed	HCLK cycle
DTST	Data set-up time	1, 2, A, B, C, D and multiplexed	HCLK cycle
DTLAT	Data latency time	Synchronous	XMC_CLK cycle
CLKPSC	Clock prescaler	Synchronous	HCLK cycle

In addition to timing parameter registers for timing control, if the wait enable bit (NWASEN or NWSEN) is enabled, the XMC will start to check whether the XMC_NWAIT signal is in wait request state during data set time. If so, the XMC will wait until the XMC_NWAIT returns to the ready state before data transfer.

25.4.2.1 Read/write operation with same timings

The timings of read and write operation in mode 1 and mode 2 are based on the XMC_BK1TMG register configuration.

Mode 1

As configured in Table 25-10 and Table 25-11, the XMC uses mode 1 to access the external memory. The timing of read operation is shown in Figure 25-3. The timing of write operation is shown in Figure 25-4.

Table 25-10 Mode 1— SRAM/NOR Flash chip select control register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS: CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications
Bit 14	RWTD: Read-write timing different	0x0
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR flash access enable	0x0
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications
Bit 3: 2	DEV: Memory device type	Configure according to memory specifications. It is valid except 0x2 (NOR Flash)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-11 Mode 1— SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-3 and Figure 25-4. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-3 and Figure 25-4. Configure according to needs and memory specifications.

Figure 25-3 NOR/PSRAM mode 1 read access

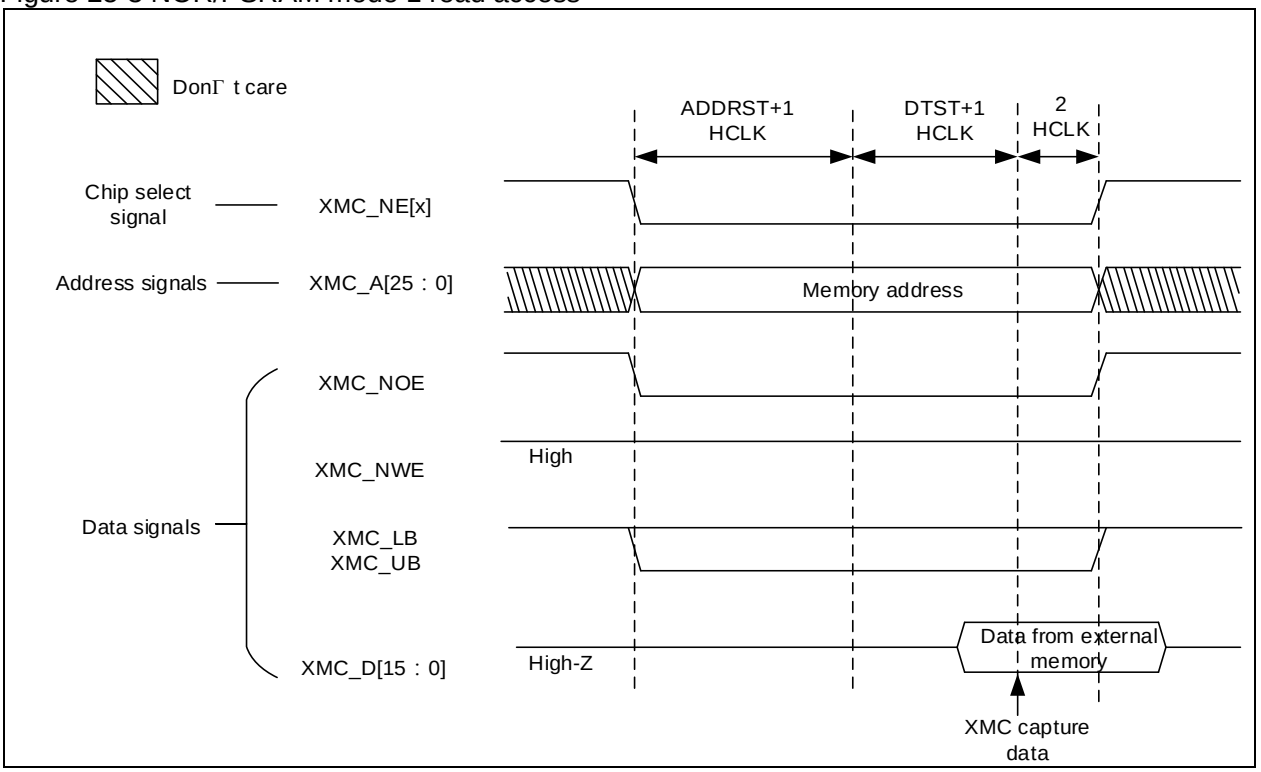
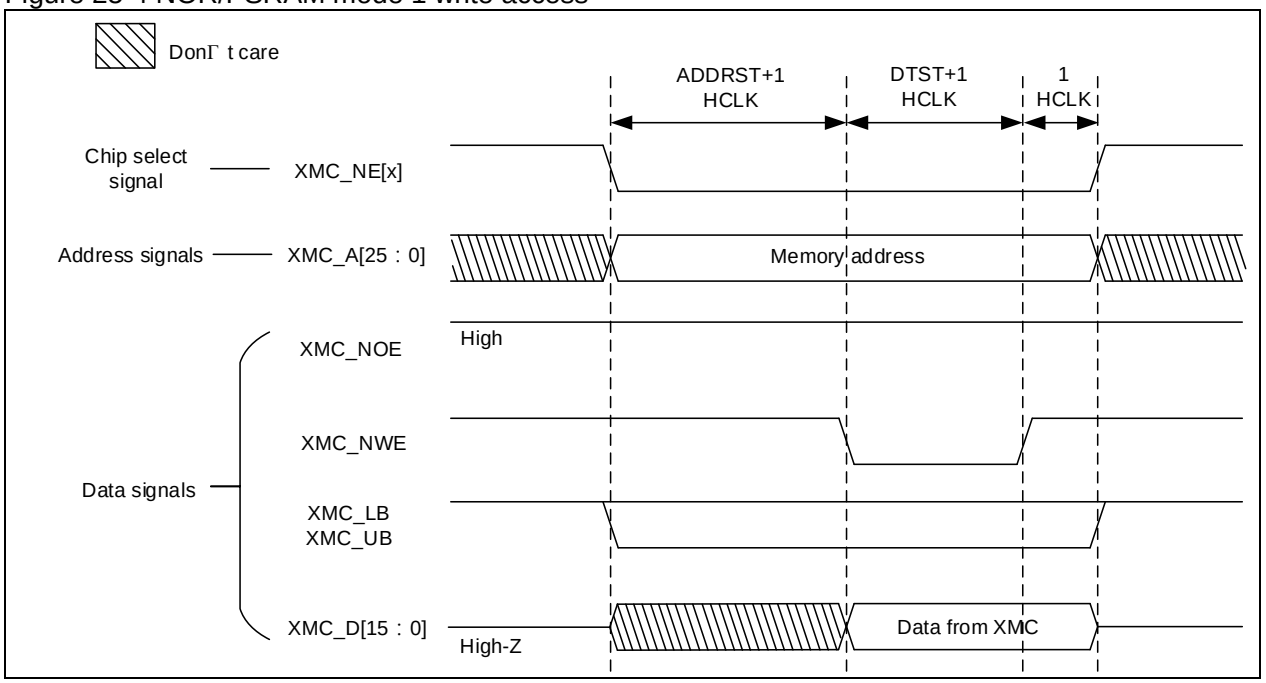


Figure 25-4 NOR/PSRAM mode 1 write access



Mode 2

As configured in Table 25-12 and Table 25-13, the XMC uses mode 2 to access the external memory. The timing of read operation is shown in Figure 25-5. The timing of write operation is shown in Figure 25-6.

Table 25-12 Mode 2 — SRAM/NOR Flash chip select control register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x0
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	0x1
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	0x2 (NOR Flash)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-13 Mode 2 — SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-5 and Figure 25-6. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-5 and Figure 25-6. Configure according to needs and memory specifications.

Figure 25-5 NOR/PSRAM mode 2 read access

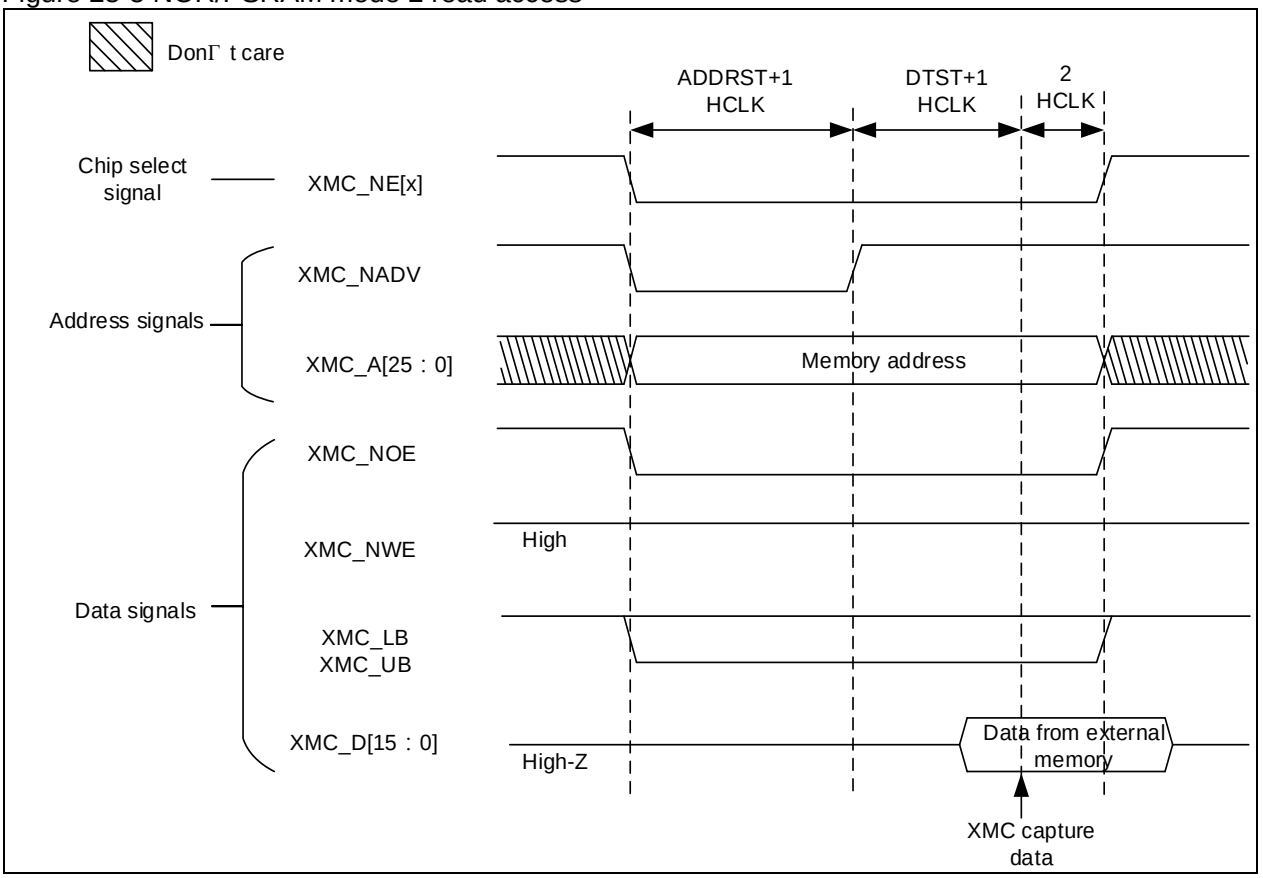
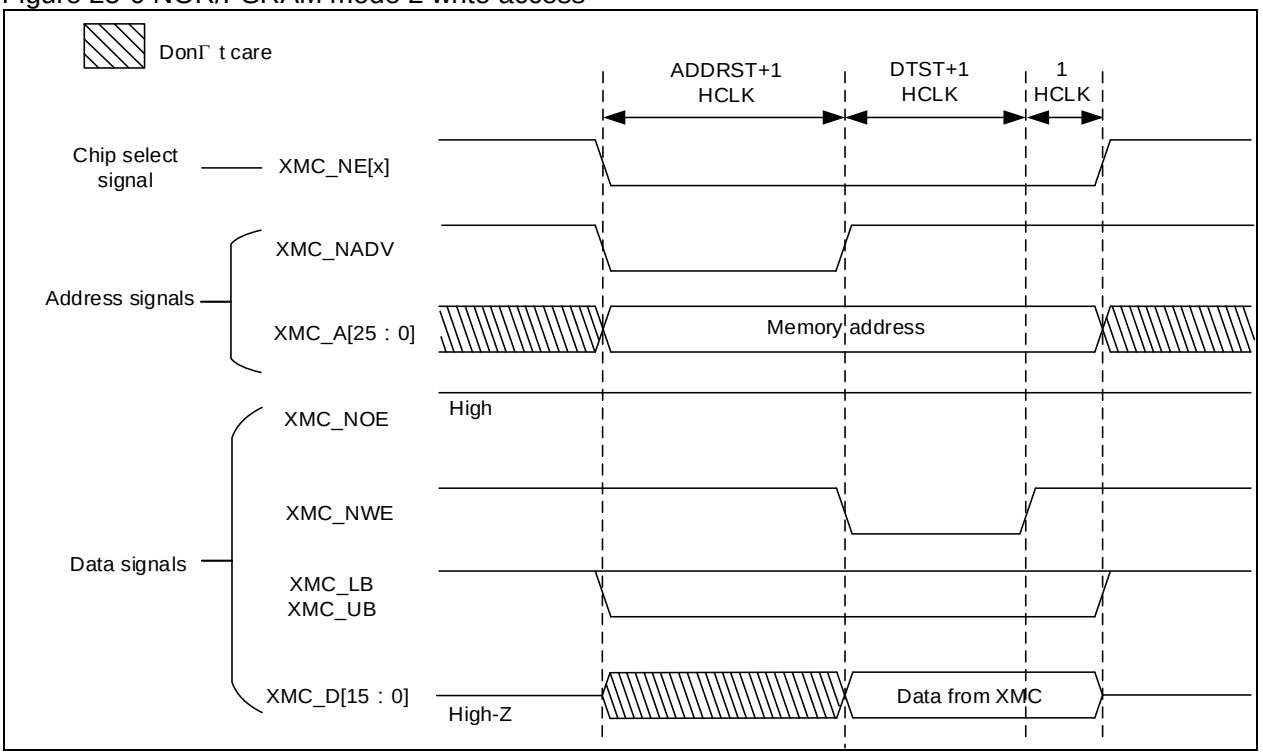


Figure 25-6 NOR/PSRAM mode 2 write access



25.4.2.2 Read/write operation with different timings

The timing of read operation in mode A/B/C/D is based on the SRAM/NOR Flash chip select timing register (XMC_BK1TMG). The timing of write operation is based on SRAM/NOR Flash write timing register (XMC_BK1TMGWR). In addition to this, it is possible to mix A, B, C and D modes for read and write operations.

Mode A

As configured in Table 25-14 Table 25-15, and Table 25-15, the XMC uses mode A to access the external memory. The timing of read operation is shown in Figure 25-7. The timing of write operation is shown in Figure 25-8.

Table 25-14 Mode A— SRAM/NOR Flash chip select control register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x1
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	0x0
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	Configure according to memory specifications. It is valid except 0x2 (NOR Flash)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-15 Mode A— SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0 (Mode A)
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-7.Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-7.Configure according to needs and memory specifications.

Table 25-16 Mode A— SRAM/NOR Flash write timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0 (Mode A)
Bit 27: 20	Reserved	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-8. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-8. Configure according to needs and memory specifications.

Figure 25-7 NOR/PSRAM mode A read access

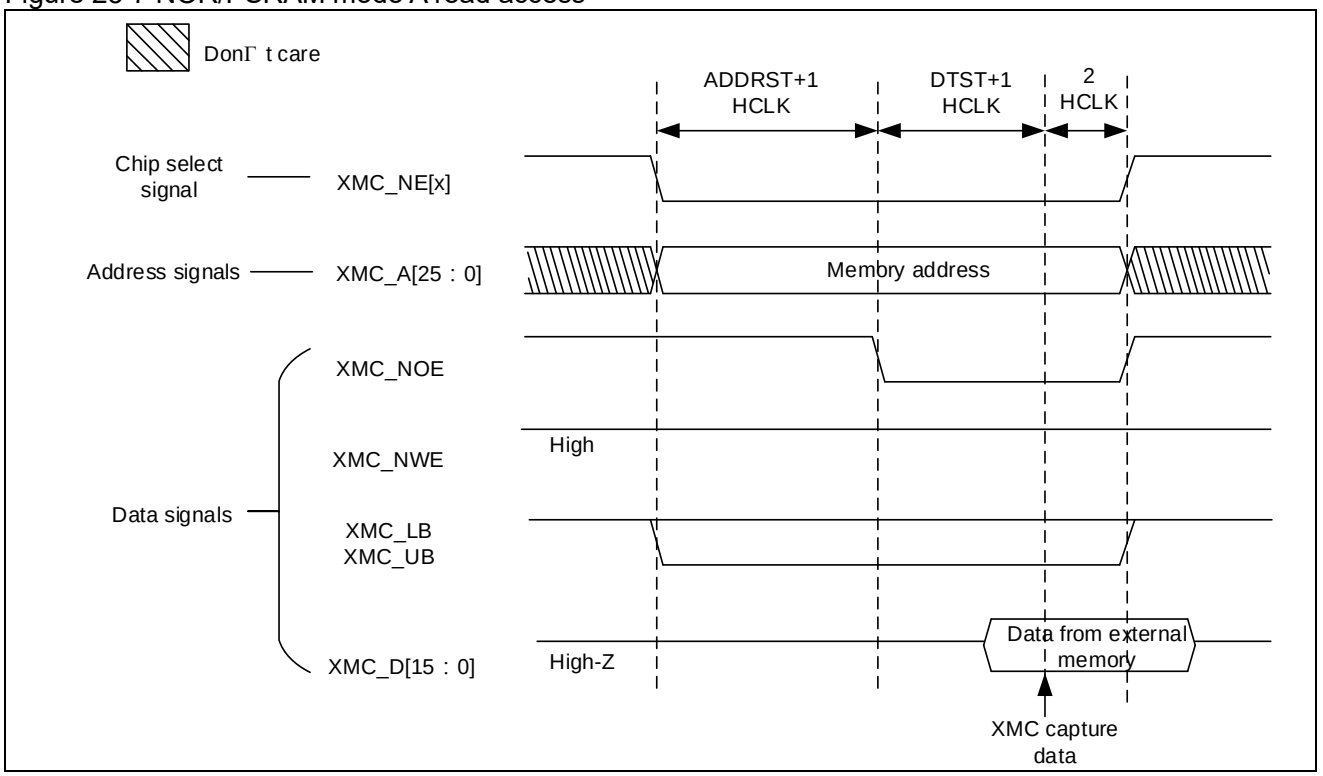
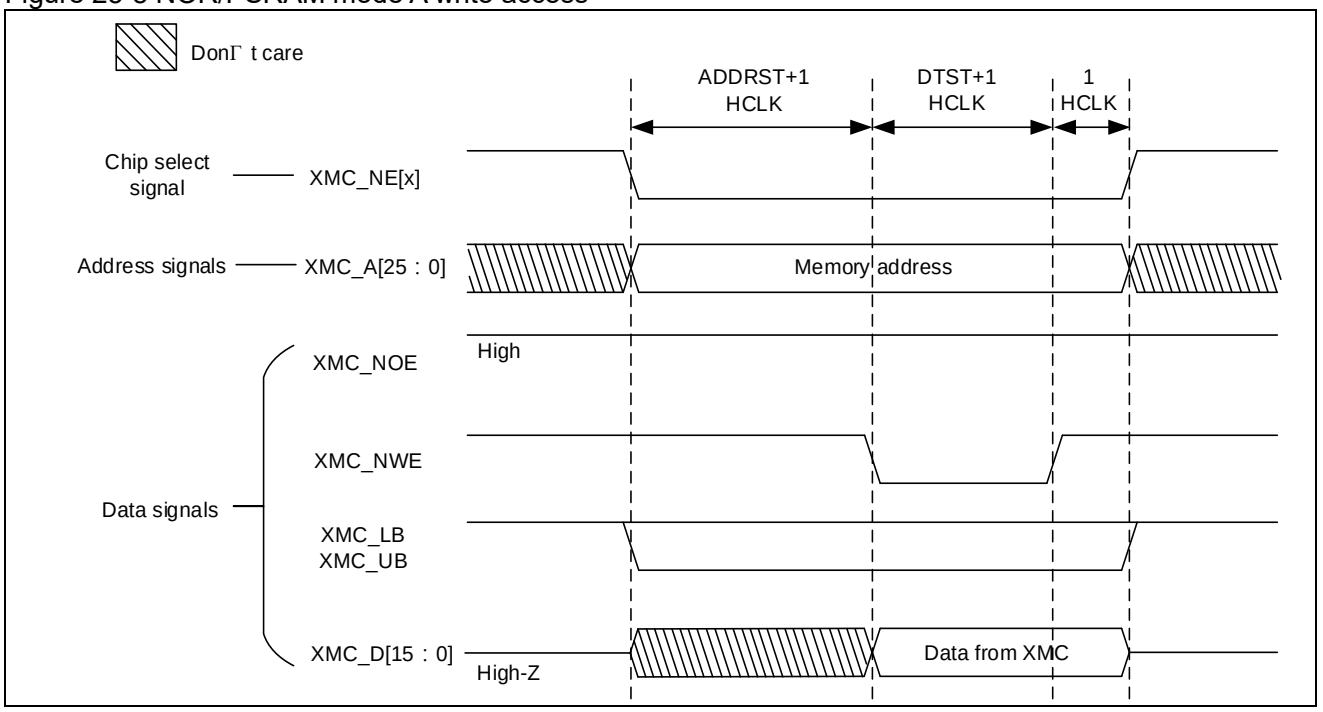


Figure 25-8 NOR/PSRAM mode A write access



Mode B

As configured in Table 25-17, Table 25-18, and Table 25-19, the XMC uses mode B to access the external memory. The timing of read operation is shown in Figure 25-9.The timing of write operation is shown in Figure 25-10.

Table 25-17 Mode B— SRAM/NOR Flash chip select register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x1
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	0x1
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	0x2 (NOR Flash)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-18 Mode B— SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x1 (Mode B)
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-9. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-9. Configure according to needs and memory specifications.

Table 25-19 Mode B— SRAM/NOR Flash write timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x1 (Mode B)
Bit 27: 20	Reserved	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-10. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-10. Configure according to needs and memory specifications.

Figure 25-9 NOR/PSRAM mode B read access

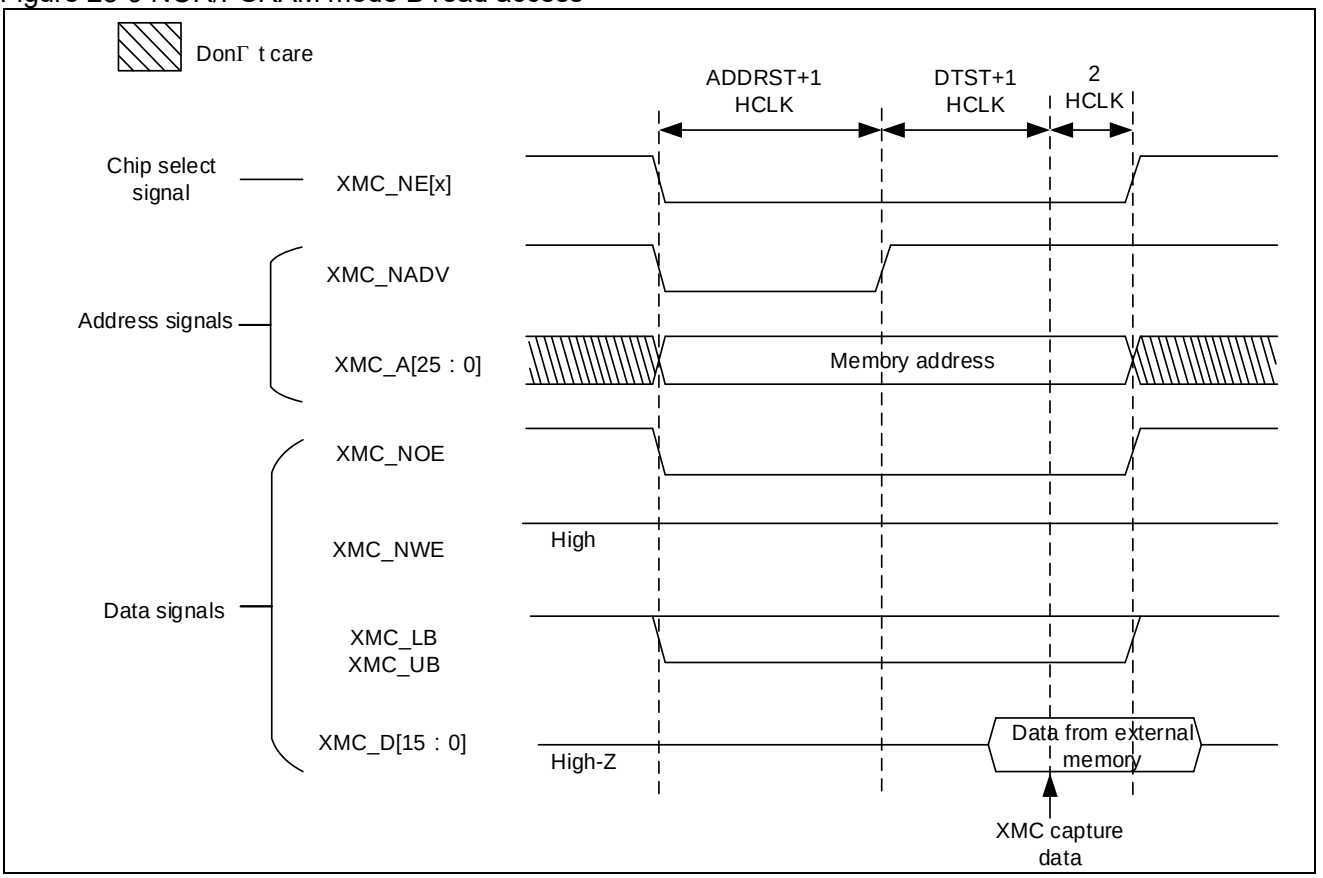
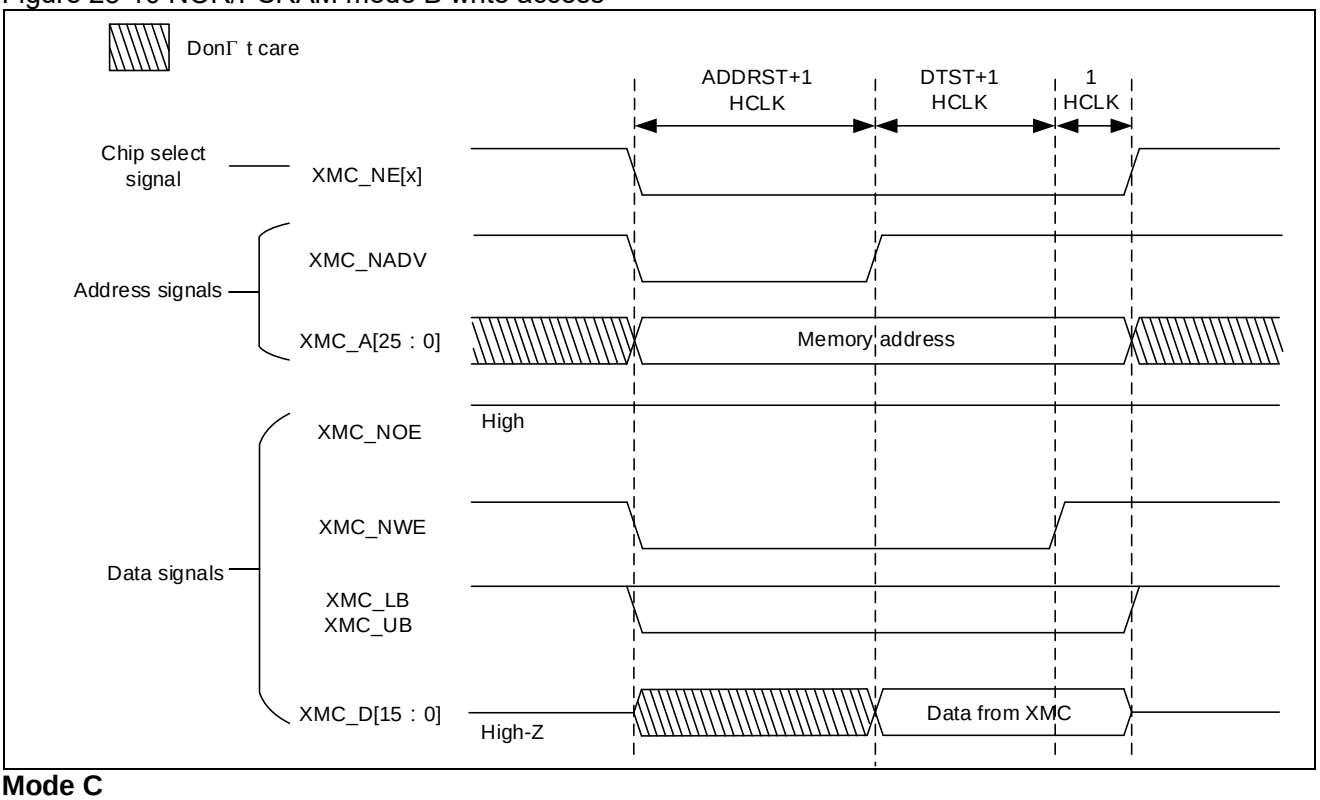


Figure 25-10 NOR/PSRAM mode B write access



Mode C

As configured in Table 25-20, Table 25-21 and Table 25-22, the XMC uses mode C to access the external memory. The timing of read operation is shown in Figure 25-11. The timing of write operation is shown in Figure 25-12.

Table 25-20 Mode C— SRAM/NOR Flash chip select register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x1
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	0x1
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	0x2 (NOR Flash)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-21 Mode C—SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x2 (Mode C)
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-11. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-11. Configure according to needs and memory specifications.

Table 25-22 Mode C— SRAM/NOR Flash write timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x1 (Mode C)
Bit 27: 20	Reserved	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-12. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-12. Configure according to needs and memory specifications.

Figure 25-11 NOR/PSRAM mode C read access

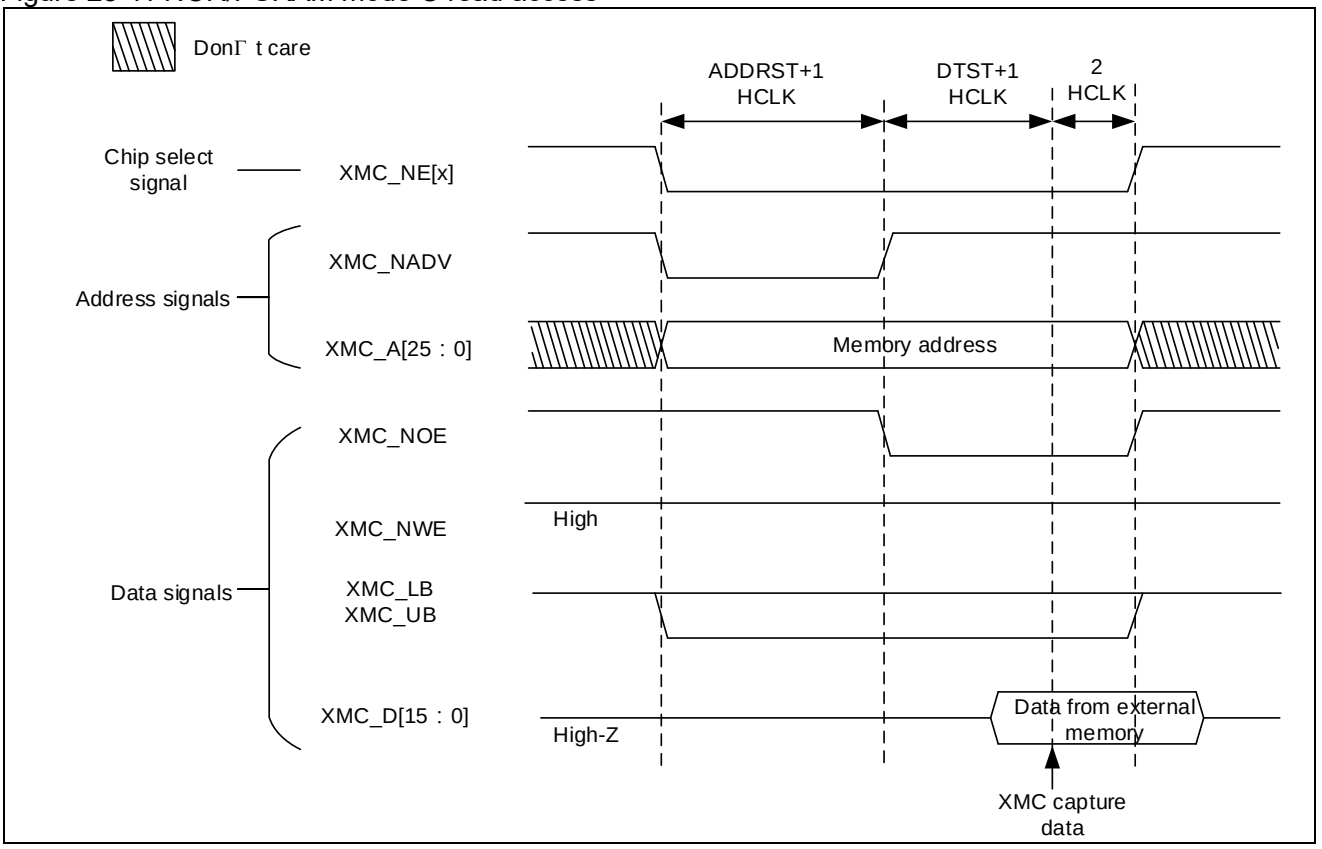
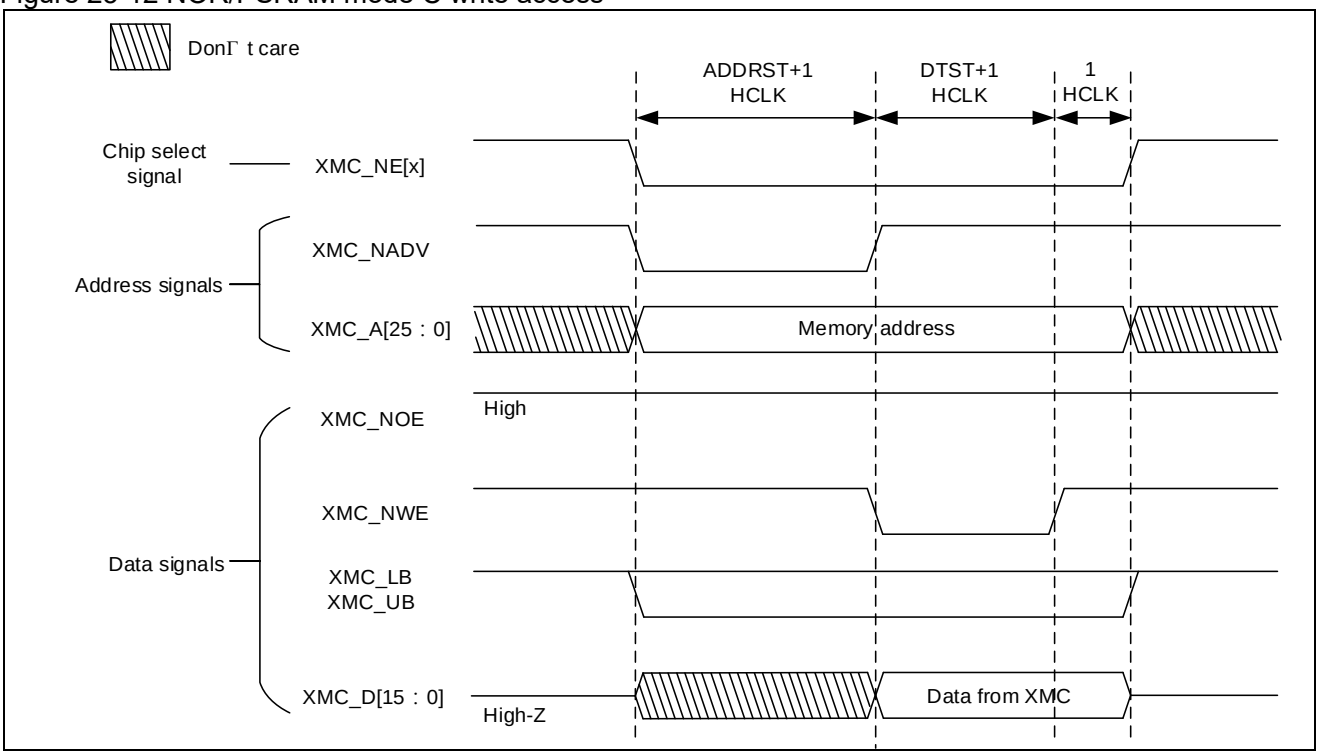


Figure 25-12 NOR/PSRAM mode C write access



Mode D

As configured in Table 25-23, Table 25-24, and Table 25-25, the XMC uses mode D to access the external memory. The timing of read operation is shown in Figure 25-13.The timing of write operation is shown in Figure 25-14.

Table 25-23 Mode D— SRAM/NOR Flash chip select register (XMC_BK1CTRL) configuration

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x1
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	Configure according to memory specifications.
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	Configure according to memory specifications.
Bit 1	ADMUXEN: Address/data multiplexing enable	0x0
Bit 0	EN: Memory bank enable	0x1

Table 25-24 Mode D—SRAM/NOR Flash chip select timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x3 (Mode D)
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-13. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	Refer to Figure 25-13. Configure according to needs and memory specifications.
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-13. Configure according to needs and memory specifications.

Table 25-25 Mode D— SRAM/NOR Flash write timing register

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x3 (Mode D)
Bit 27: 20	Reserved	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-14. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	Refer to Figure 25-14. Configure according to needs and memory specifications.
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-14. Configure according to needs and memory specifications.

Figure 25-13 NOR/PSRAM mode D read access

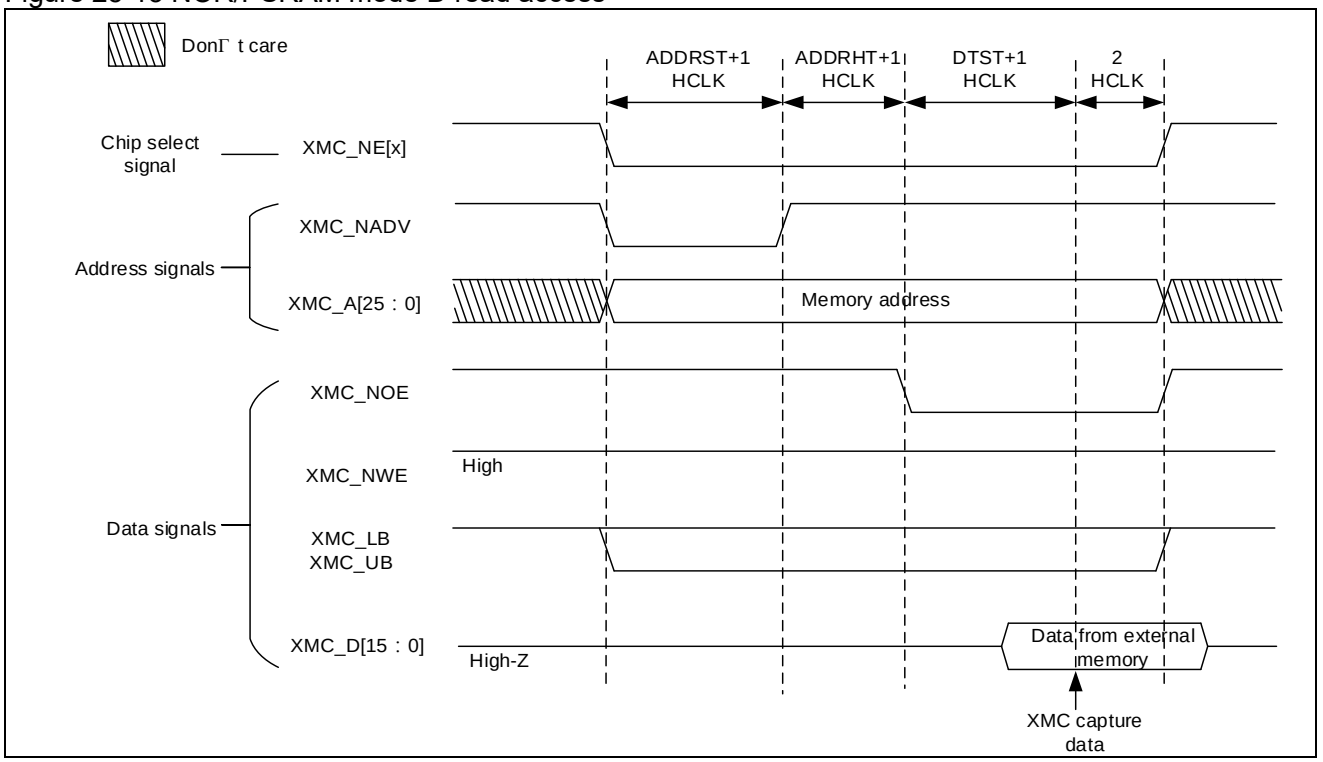
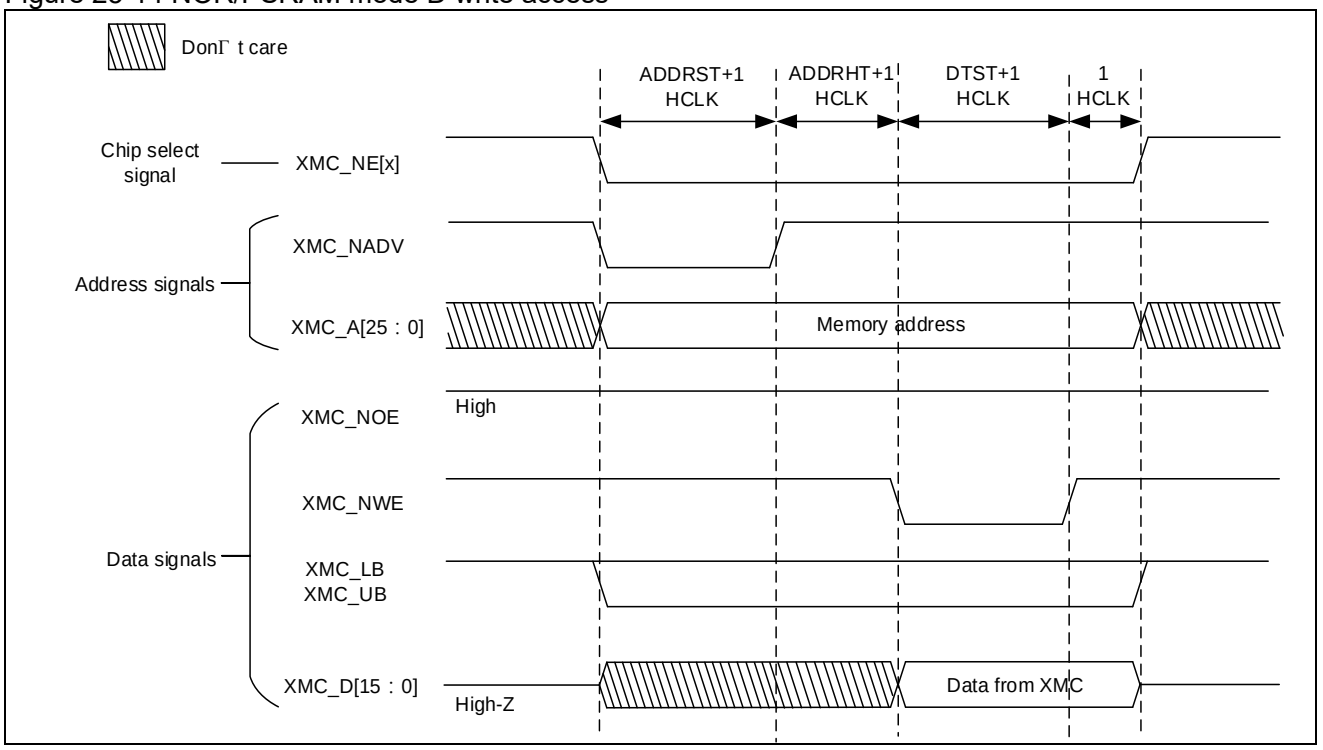


Figure 25-14 NOR/PSRAM mode D write access



25.4.2.3 Multiplexed mode

As configured in Table 25-26 and Table 25-27, the XMC uses mode A to access the external memory. The timing of read operation is shown in Figure 25-15. The timing of write operation is shown in Figure 25-16.

Table 25-26 Multiplexed mode — SRAM/NOR Flash chip select control register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x0
Bit 18: 16	CRPGS:CRAM page size	0x0
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	Configure according to memory specifications.
Bit 14	RWTD: Read-write timing different	0x0
Bit 13	NWSEN: NWAIT in synchronous transfer enable	0x0
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	0x0
Bit 10	WRAPEN: Wrapped enable	0x0
Bit 9	NWPOL: NWAIT polarity	Configure according to memory specifications.
Bit 8	SYNCBEN: Synchronous burst enable	0x0
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	Configure according to memory specifications.
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	Configure according to memory specifications. It is valid except 0x0 (SRAM)
Bit 1	ADMUXEN: Address/data multiplexing enable	0x1
Bit 0	EN: Memory bank enable	0x1

Table 25-27 Multiplexed mode—SRAM/NOR Flash chip select timing register (XMC_BK1TMG) configuration

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0
Bit 27: 24	DTLAT: Data latency	0x0
Bit 23: 20	CLKPSC: Clock prescale	0x0
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	Refer to Figure 25-15 and Figure 25-16. Configure according to needs and memory specifications.
Bit 7: 4	ADDRHT: Address-hold time	Refer to Figure 25-15 and Figure 25-16. Configure according to needs and memory specifications.
Bit 3: 0	ADDRST: Address setup time	Refer to Figure 25-15 and Figure 25-16. Configure according to needs and memory specifications.

Figure 25-15 NOR/PSRAM multiplexed mode read access

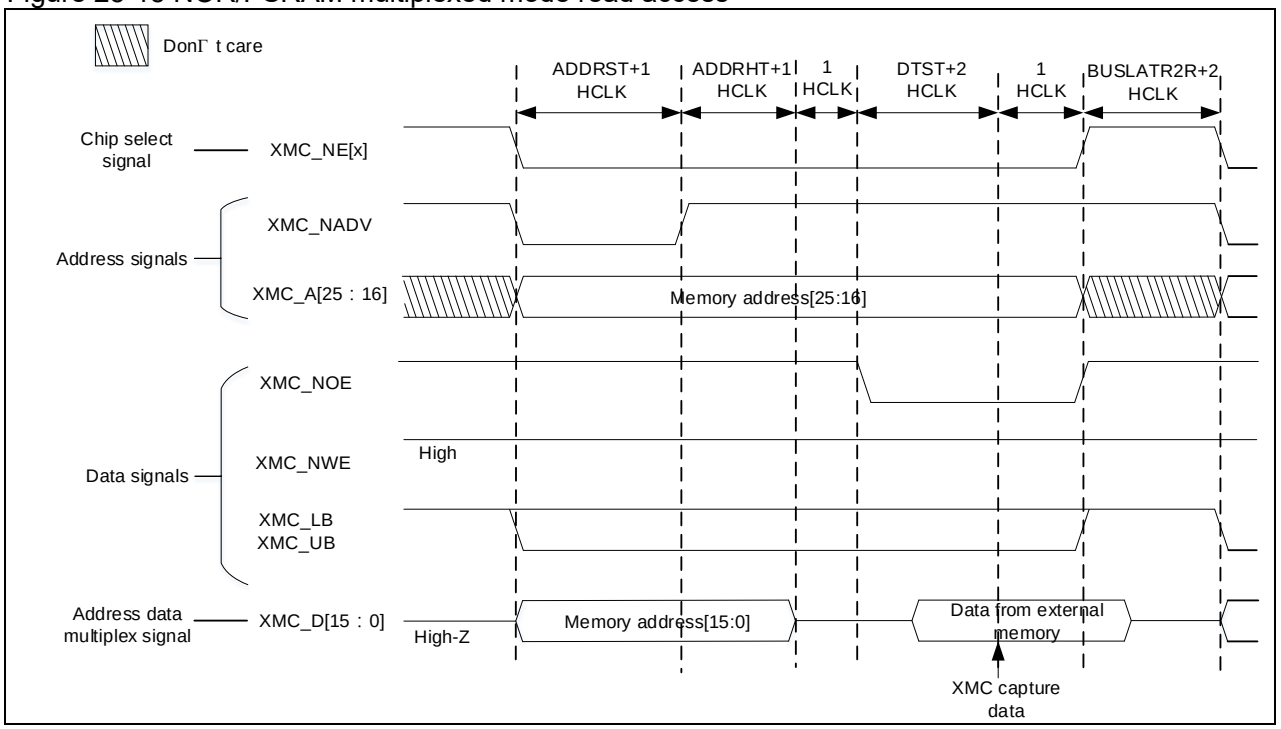
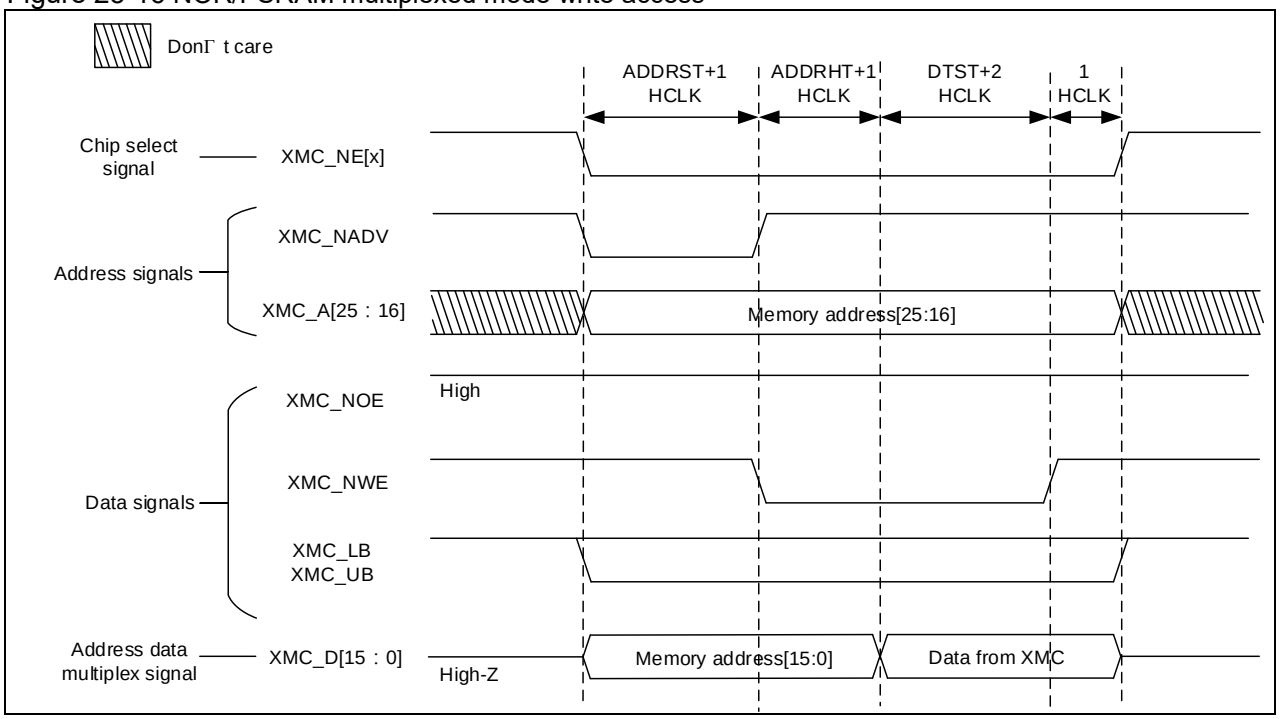


Figure 25-16 NOR/PSRAM multiplexed mode write access



25.4.2.4 Synchronous mode

As configured in Table 25-28 and Table 25-29, the XMC uses synchronous mode to access the external memories.

If the memory inserts XMC_NWAIT signal between the address latch and data transfer, the XMC will not only wait (DTLAT+1) CLK clock cycles but also have to take into account the XMC_NWAIT signal. During data transmission, the XMC will, depending on the NWTCFG configuration, select to wait either one cycle after the XMC_NWAIT signal is active or when the XMC_NWAIT signal is active.

Figure 25-17 shows the timing for read access, while Figure 25-18 shows the timing for write access. Figure 25-17 and Figure 25-18 are examples of XMC waiting in the next cycle of XMC_NWAIT signal (NWTCFG=0)

Table 25-28 Synchronous mode — SRAM/NOR Flash chip select control register

Bit	Description	Configuration
Bit 31: 20	Reserved	0x0
Bit 19	MWMC: Memory write mode control	0x1
Bit 18: 16	CRPGS:CRAM page size	Configure according to memory specifications.
Bit 15	NWASEN: NWAIT in asynchronous transfer enable	0x0
Bit 14	RWTD: Read-write timing different	0x0
Bit 13	NWSEN: NWAIT in synchronous transfer enable	Configure according to memory specifications.
Bit 12	WEN: Write enable	Configure according to needs.
Bit 11	NWTCFG: NWAIT timing configuration	Configure according to memory specifications.
Bit 10	WRAPEN: Wrapped enable	Configure according to needs.
Bit 9	NWPOL: NWAIT polarity	c
Bit 8	SYNCBEN: Synchronous burst enable	0x1
Bit 7	Reserved	0x1
Bit 6	NOREN: NOR Flash access enable	Write synchronization: 0x0 Read synchronization: Configure according to memory specifications.
Bit 5: 4	EXTMDBW: External memory data bus width	Configure according to memory specifications.
Bit 3: 2	DEV: Memory device type	Write synchronization: 0x1 Read synchronization: Configure according to memory specifications. It is valid except 0x0 (SRAM)
Bit 1	ADMUXEN: Address/data multiplexing enable	Configure according to needs.
Bit 0	EN: Memory bank enable	0x1

Table 25-29 Synchronous mode—SRAM/NOR Flash chip select timing register (XMC_BK1TMG)

Bit	Description	Configuration
Bit 31: 30	Reserved	0x0
Bit 29: 28	ASYNCM: Asynchronous mode	0x0
Bit 27: 24	DTLAT: Data latency	Refer to Figure 25-17 and Figure 25-18
Bit 23: 20	CLKPSC: Clock prescale	XMC_CLK cycle is HCLK cycle*(CLKPSC+1). Refer to Figure 25-17 and Figure 25-18
Bit 19: 16	BUSLAT: Bus latency	Indicates the time the XMC_NE[x] from the rising edge to the falling edge. Configure according to needs and memory specifications
Bit 15: 8	DTST: Data setup time	0x0
Bit 7: 4	ADDRHT: Address-hold time	0x0
Bit 3: 0	ADDRST: Address setup time	0x0

Figure 25-17 NOR/PSRAM synchronous multiplexed mode read access

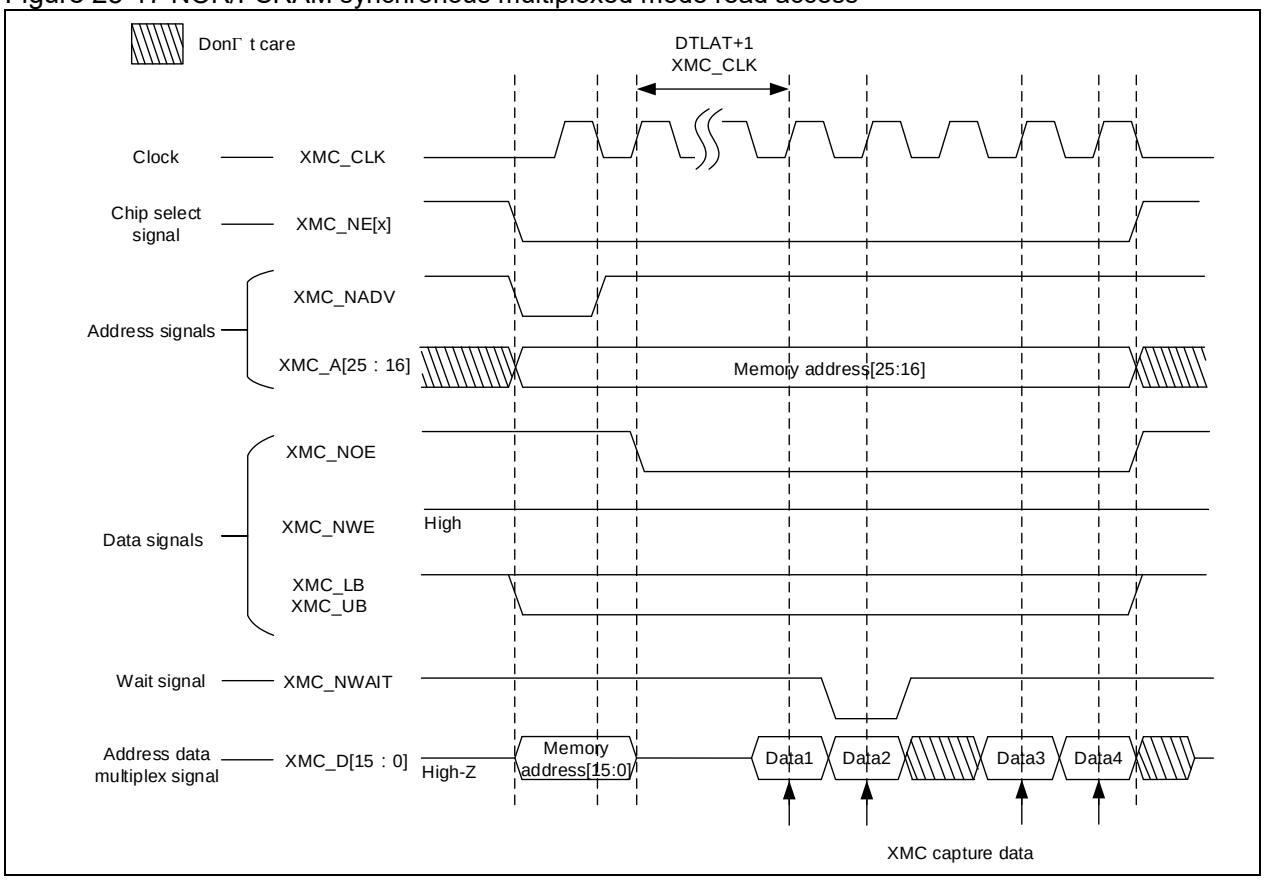
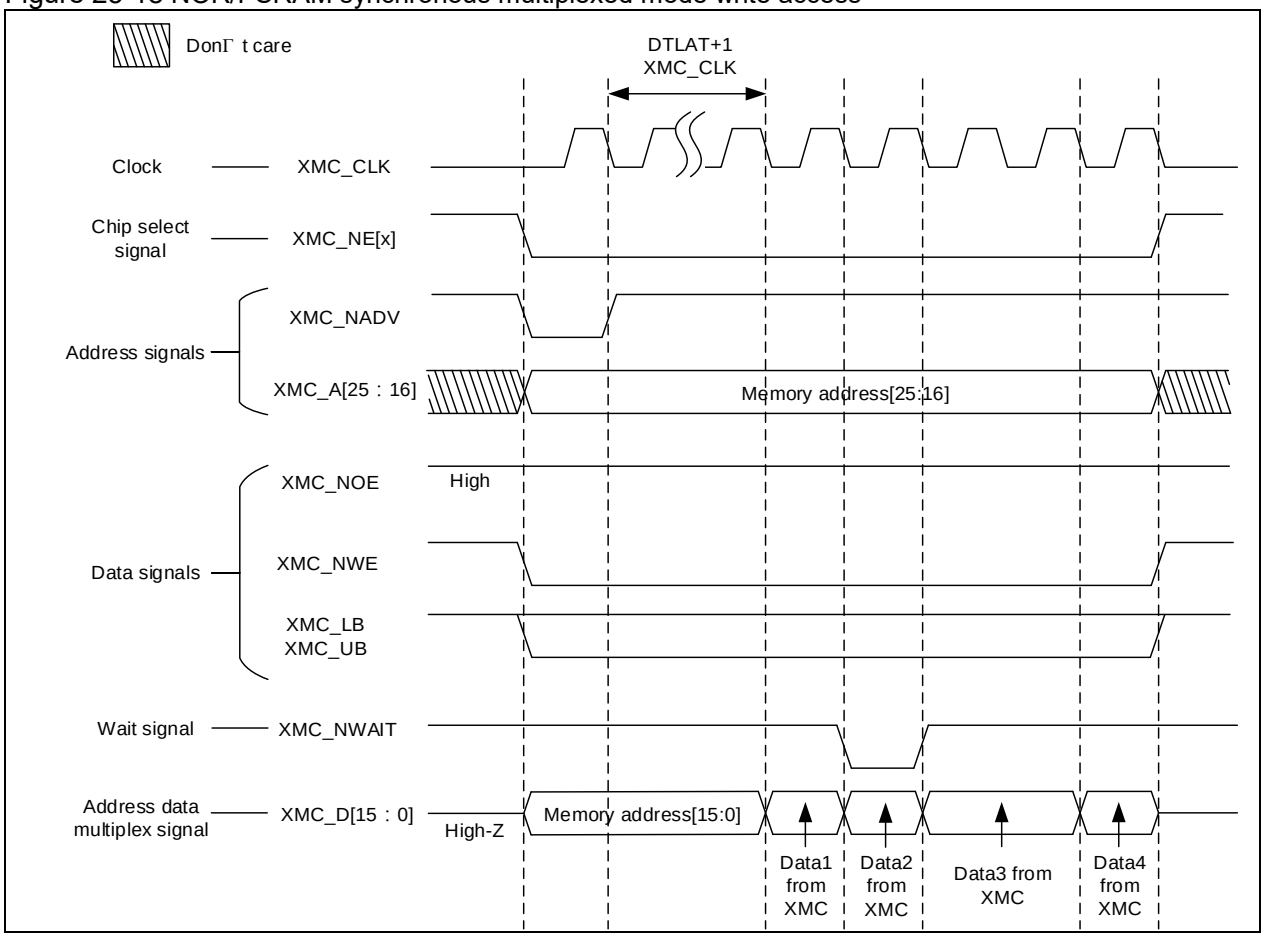


Figure 25-18 NOR/PSRAM synchronous multiplexed mode write access



25.5 SDRAM card

25.5.1 SDRAM access management

SDRAM initialization

The SDRAM initialization sequence defined in the JEDEC specification must be respected before using SDRAM.

The SDRAM initialization is managed by software. To initialize two devices, the BK1 and BK2 bit must be set in the SDRAM_CMD register. The initialization command is applicable to both devices.

1. Program the SDRAM_CTRLx register according to the external SDRAM device row/column and bus width and the number of BANK. Read and write operations from/to the SDRAM must be also programmed into the SDRAM_CTRLx register.
2. Program the memory device timing into the SDRAM_TMx register. The TRP and TRC timings must be programmed into the SDRAM_TM1 register.
3. Set CMD=001 to start providing the clock to the memory.
4. Wait for 100us
5. Set CMD=010 to issue a “Precharge All” command.
6. Set CMD=011 and configure the number of consecutive Auto-refresh commands (ART). Typical number is 8.
7. Configure the MRD field (controller supports BL=1 only). Set CMD=100 to issue a “Load Mode Register” command. If the mode registers of these two SDRAM devices are different from each other, this step has to be repeated twice (by setting the BK1 and BK2)
8. Program the refresh counter in the SDRAM_RCNT register to determine the SDRAM auto-refresh rate.

After the initialization, the SDRAM is ready to accept commands. If a system reset occurs during an on-going SDRAM access, the SDRAM device has to be reset after re-initialization in order to issue a new

access to the SDRAM.

If the “Load Mode Register” and “Self-Refresh” commands are applied to both devices at the same time, access commands are issued using the timing parameters configured for SDRAM device 1.

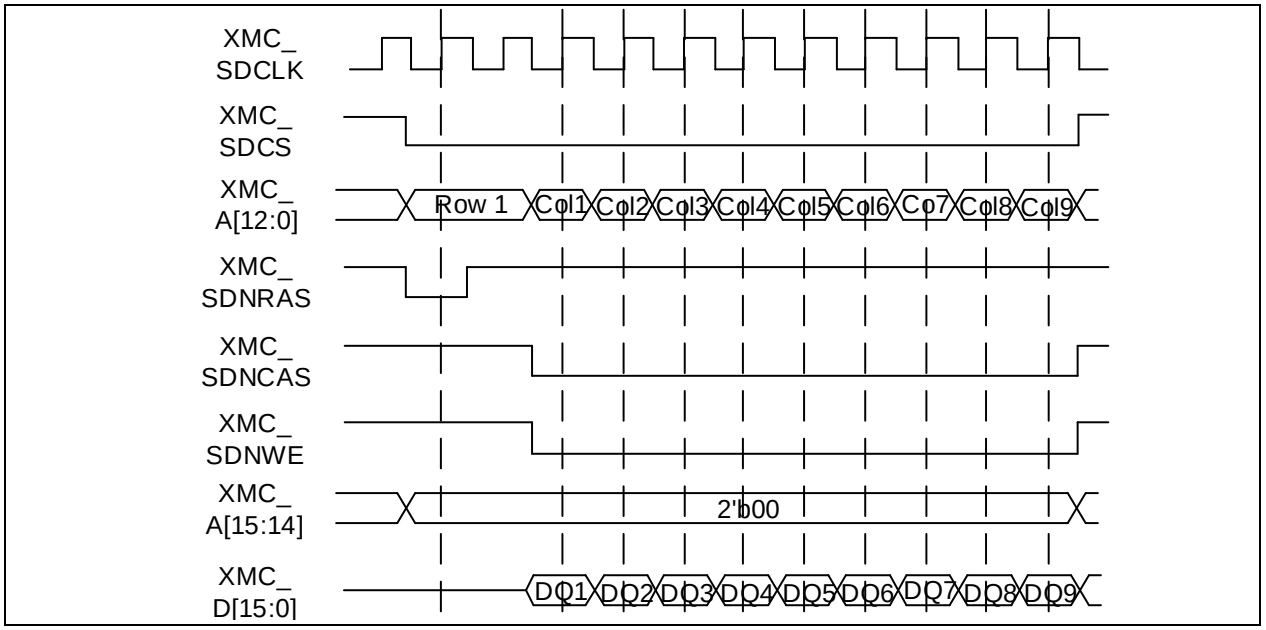
SDRAM controller write operation

Before performing any write access, the SDRAM write protection must be disabled by clearing the WRP bit in the SDRAM_CTRLx register; otherwise, an AHB error may occur.

The SDRAM controller translates single or burst AHB write requests into a single SDRAM write access, which can be done by setting BL=1. In both cases, the SDRAM controller always tries to translate consecutive or discontinuous AHB write requests into a single SDRAM write request in order to increase efficiency.

The SDRAM controller keeps track of the active row for each BANK in order to be able to perform consecutive write accesses to different banks (multibank ping-pong access). If the next write access is in the same row or in another active row, the write operation is performed. If the next write access targets an inactive row, the SDRAM controller generates a precharge command (The BANK where the inactive row is has other open rows, if no other rows, the precharge is unnecessary), and activates the new row and initiates a write operation.

Figure 25-19 SDRAM write access waveforms (Trcd=2, 9 consecutive write access)



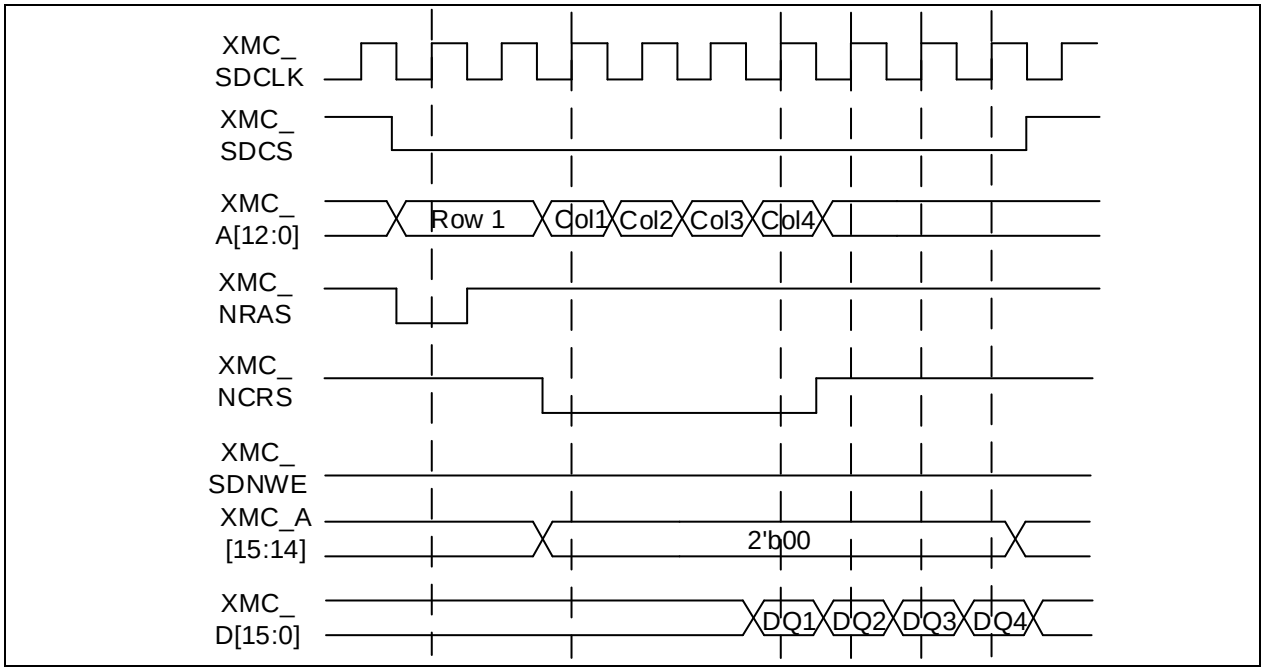
SDRAM controller read operation

The SDRAM controller translates single or burst AHB read requests into a single SDRAM read access, which can be done by setting BL=1.

The SDRAM controller keeps track of the active row for each BANK in order to be able to perform consecutive read accesses to different banks (multibank ping-pong access). If the next read access is in the same row or in another active row, the read operation is performed. If the next read access targets an inactive row, the SDRAM controller generates a precharge command (The BANK where the inactive row is has other open rows, if no other rows, the precharge is unnecessary), and activates the new row and initiates a read operation.

Note: The above-mentioned description applies to the scenario where the SDRAM consecutive read feature is not enabled (that is, read FIFO is unused.)

Figure 25-20 SDRAM read access without using read FIFO (Trcd=2,CL=3, and 4 consecutive read accesses)



When the BSTR is set in the SDRAM_CTRL1 register, the SDRAM controller continues to send another read command after issuing the current read command. The number of read commands to be issues depends on the CAS latency and RD read delay. The data read in advance during the CAS latency is stored into the read FIFO. Each line of the read FIFO has its address tag. The number of the anticipated read commands (M) is calculated as follows:

M = CAS cycles+RD/2+1

For example, CAS=3 and RD= 3xHCLK, then 5 data (not committed) are stored in the FIFO.

Each time a read request occurs, the SDRAM controller will check whether the FIFO address tag matches the address to be read. If matched, take the desired data from the FIFO; if unmatched, issue a read command to the memory, and update FIFO at the same time.

Row and BANK boundary management

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, then the SDRAM controller precharges the currently active row in order to disable the current row, activates a new row, and starts a read/write command to a new row.

If the next access is sequential and the current access crosses BANK boundary, two cases are possible:

- If the current BANK is not the last one, the SDRAM will perform access to the next BANK. If the next BANK is an active row and the active row matches the one to be accessed, a read operation is issued; if such active row is not the one to be accessed, disable the active row and activate a new row; if no active row, activate directly a new row to be accessed, and issue a read/write command.
- If the current BANK is the last memory area, an AHB error occurs.

SDRAM controller self-refresh operation

The SDRAM device needs an auto-refresh operation. The SDRAM controller periodically issues auto-refresh commands. The refresh rate is defined by the RC value in the SDRAM_RCNT register. The auto-refresh commands have priority over read/write accesses. In other words, if the auto-refresh commands and read/write access commands are generated simultaneously, the read/write requests are suspended temporarily, and are processed when the auto-refresh is complete.

If the SDRAM has active rows, the SDRAM controller generates all precharge commands before auto-refresh operations.

If a new auto-refresh request occurs while the previous one was not complete, the ERR (Refresh Error) bit is set in the SDRAM_STS register. An interrupt is generated if the ERIEN has been set.

25.5.2 Self-refresh mode and Power-down mode

The SDRAM controller supports two low-power modes: self-refresh mode and power-down mode.

Self-refresh mode

This mode is selected by setting CMD=101 and by configuring the Target Bank bits (BK1 and/or BK2) in the SDRAM_CMD register.

The SDRAM clock stops running after a TRAS delay. The internal refresh time stops counting if one of the following conditions is present:

When both SDRAM devices have been initialized, a self-refresh command is issued to both devices; or when one of the devices is not initialized, a self-refresh command is issued to one of the device.

After entering the self-refresh mode, the SDRAM device must remain in this mode for a minimum period of time of TRAS, which is determined by the SDRAM device characteristics. To guarantee this minimum period, the BUSY flag will be set after the self-refresh is entered. The BUSY flag is cleared automatically only after a TRAS time. This bit can be used to judge whether to exit self-refresh mode.

It is possible to leave self-refresh mode by setting CMD=000 or by read/write access to the SDRAM.

Power-down mode

This mode is selected by setting CMD=110 and by configuring the Target Bank bits (BK1 and/or BK2) in the SDRAM_CMD register.

During power-down mode, the SDRAM device can also perform auto-refresh operation. When an auto-refresh command occurs, the SDRAM exists from the power-down mode before performing auto-refresh operation. After that, the SDRAM will enter power-down mode again.

It is possible to leave power-down mode by setting CMD=000 or by read/write access to the SDRAM.

25.6 XMC registers

These peripheral registers must be accessed by words (32 bits).

Table 25-30 XMC register address mapping

Register	Offset	Reset value
XMC_BK1CTRL1	0x000	0x0000 30DB
XMC_BK1TMG1	0x004	0x0FFF FFFF
XMC_BK1CTRL2	0x008	0x0000 30D2
XMC_BK1TMG2	0x00C	0x0FFF FFFF
XMC_BK1CTRL3	0x010	0x0000 30D2
XMC_BK1TMG3	0x014	0x0FFF FFFF
XMC_BK1CTRL4	0x018	0x0000 30D2
XMC_BK1TMG4	0x01C	0x0FFF FFFF
XMC_BK1TMGWR1	0x104	0x0FFF FFFF
XMC_BK1TMGWR2	0x10C	0x0FFF FFFF
XMC_BK1TMGWR3	0x114	0x0FFF FFFF
XMC_BK1TMGWR4	0x11C	0x0FFF FFFF
XMC_EXT1	0x220	0x0000 0808
XMC_EXT2	0x224	0x0000 0808
XMC_EXT3	0x228	0x0000 0808
XMC_EXT4	0x22C	0x0000 0808
SDRAM_CTRL1	0x140	0x0000 02D0
SDRAM_CTRL2	0x144	0x0000 02D0
SDRAM_TM1	0x148	0x0FFF FFFF
SDRAM_TM2	0x14C	0x0FFF FFFF

SDRAM_CMD	0x150	0x0000 0007
SDRAM_RCNT	0x154	0x0000 0000
SDRAM_STS	0x158	0x0000 0000
SDRAM_CLKDIV1	0x180	0x0000 0000

25.6.1 NOR Flash and PSRAM control registers

These peripheral registers have to be accessed by words (32 bits).

25.6.1.1SRAM/NOR Flash chip select control register 1 (XMC_BK1CTRL1)

Accessed by words

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x000	resd	Kept at its default value.
Bit 19	MWMC	0x0	rw	Memory write mode control 0: Write operations are performed in asynchronous mode 1: Write operations are performed in synchronous mode
Bit 18: 16	CRPGS	0x0	rw	CRAM page size Cellular RAM 1.5 does not allow synchronous access to cross the address boundaries between pages. When these bits are configured in synchronous mode, the XMC will automatically split the access when the page size is reached. 000: No split access when crossing address boundary (default value) 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved.
Bit 15	NWASEN	0x0	rw	NWAIT in asynchronous transfer enable 0: NWAIT signal is disabled 1: NWAIT signal is enable
Bit 14	RWTD	0x0	rw	Read-write timing different Different timings are used for read and write operations, that is, the XMC_BK1TMGWR1 register is enabled. 0: Same timings for read and write operations 1: Different timings for read and write operations
Bit 13	NWSEN	0x1	rw	NWAIT enable during synchronous transfer 0: NWAIT signal is disabled 1: NWAIT signal is enabled
Bit 12	WEN	0x1	rw	Write enable 0: Disabled 1: Enabled
Bit 11	NWTCFG	0x0	rw	NWAIT timing configuration It is valid only in synchronous mode. 0: NWAIT signal is active one data cycle before the wait state 1: NWAIT signal is active one data cycle during the wait state
Bit 10	WRAPEN	0x0	rw	Wrapped enable This bit defines whether the XMC will split a wrapped AHB access into two accesses. 0: Direct wrapped access is not allowed 1: Direct wrapped access is allowed
Bit 9	NWPOL	0x0	rw	NWAIT polarity This bit defines the polarity of the NWAIT signal in synchronous mode. 0: NWAIT active low 1: NWAIT active high

Bit 8	SYNCBEN	0x0	rw	Synchronous burst enable This bit allows synchronous access to Flash memories. 0: Synchronous burst disabled 1: Synchronous burst enabled
Bit 7	Reserved	0x1	resd	Kept at its default value.
Bit 6	NOREN	0x1	rw	Nor flash access enable 0: Nor flash access is disabled 1: Nor flash access is enabled
Bit 5: 4	EXTMDBW	0x1	rw	External memory data bus width This field defines the external memory data bus width. 00: 8 bits 01: 16 bits 10: Reserved 11: Reserved
Bit 3: 2	DEV	0x2	rw	Memory device type 00: SRAM/ROM 01: PSRAM (Cellular RAM or CRAM) 10: NOR Flash 11: Reserved
Bit 1	ADMUXEN	0x1	rw	Address/data multiplexing enable 0: Address/data not multiplexed 1: Address/data multiplexed
Bit 0	EN	0x1	rw	Memory bank enable 0: Memory bank disabled 1: Memory bank enabled

25.6.1.2 SRAM/NOR Flash chip select control register x (x=2, 3, 4)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x000	resd	Kept at its default value.
Bit 19	MWMC	0x0	rw	Memory write mode control 0: Write operations are performed in asynchronous mode 1: Write operations are performed in synchronous mode
Bit 18: 16	CRPGS	0x0	rw	CRAM page size Cellular RAM 1.5 does not allow synchronous access to cross the address boundaries between pages. When these bits are configured in synchronous mode, the XMC will automatically split the access when the page size is reached. 000: No split access when crossing address boundary (default value) 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved.
Bit 15	NWASEN	0x0	rw	NWAIT enable during asynchronous transfer 0: NWAIT signal is disabled 1: NWAIT signal is enable
Bit 14	RWTD	0x0	rw	Read-write timing different Different timings are used for read and write operations, that is, the XMC_BK1TMGWRx register is enabled. 0: Same timings for read and write operations 1: Different timings for read and write operations
Bit 13	NWSEN	0x1	rw	NWAIT enable during synchronous transfer 0: NWAIT signal is disabled 1: NWAIT signal is enabled
Bit 12	WEN	0x1	rw	Write enable 0: Disabled 1: Enabled
Bit 11	NWTCFG	0x0	rw	NWAIT timing configuration It is valid only in synchronous mode. 0: NWAIT signal is active one data cycle before the wait state

				1: NWAIT signal is active one data cycle during the wait state
Bit 10	WRAPEN	0x0	rw	Wrapped enable This bit defines whether the XMC will split a wrapped AHB access into two accesses. 0: Direct wrapped access is not allowed 1: Direct wrapped access is allowed
Bit 9	NWPOL	0x0	rw	NWAIT polarity This bit defines the polarity of the NWAIT signal in synchronous mode. 0: NWAIT active low 1: NWAIT active high
Bit 8	SYNCBEN	0x0	rw	Synchronous burst enable This bit allows synchronous access to Flash memories. 0: Synchronous burst disabled 1: Synchronous burst enabled
Bit 7	Reserved	0x1	resd	Kept at its default value.
Bit 6	NOREN	0x1	rw	Nor flash access enable 0: Nor flash access is disabled 1: Nor flash access is enabled
Bit 5: 4	EXTMDBW	0x1	rw	External memory data bus width This field defines the external memory data bus width. 00: 8 bits 01: 16 bits 10: Reserved 11: Reserved
Bit 3: 2	DEV	0x0	rw	Memory device type 00: SRAM/ROM 01: PSRAM (Cellular RAM or CRAM) 10: NOR Flash 11: Reserved
Bit 1	ADMUXEN	0x1	rw	Address/data multiplexing enable 0: Address/data not multiplexed 1: Address/data multiplexed
Bit 0	EN	0x0	rw	Memory bank enable 0: Memory bank disabled 1: Memory bank enabled

25.6.1.3 SRAM/NOR Flash chip select timing register x (XMC_BK1TMGx) (x=1,2,3,4)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value.
Bit 29: 28	ASYNCM	0x0	rw	Asynchronous mode This field is valid only when the RWTD bit is enabled. 00: Mode A 01: Mode B 10: Mode C 11: Mode D
Bit 27: 24	DTLAT	0xF	rw	Data latency This field is valid only in synchronous mode. 0000: 1 extra XMC_CLK cycle is inserted 0001: 2 extra XMC_CLK cycles are inserted …… 1111: 16 extra XMC_CLK cycles are inserted
Bit 23: 20	CLKPSC	0xF	rw	Clock prescaler This field is valid only in synchronous mode. It defines the frequency of the XMC_CLK clock. 0000: Reserved 0001: XMC_CLK cycle= 2 x HCLK clock cycles 0010: XMC_CLK cycle =3 x HCLK clock cycles …… 1111: XMC_CLK cycle = 6 x HCLK cycles
Bit 19: 16	BUSLAT	0xF	rw	Bus latency To avoid data bus conflict, a latency is inserted on the data bus after one read operation in multiplexed or synchronous mode. 0000: 1 HCLK cycle is inserted 0001: 2 HCLK cycles are inserted …… 1111: 16 HCLK cycles are inserted
Bit 15: 8	DTST	0xFF	rw	Data setup time 00000000: 1 extra HCLK cycle is inserted 00000001: 2 extra HCLK cycles are inserted …… 11111111: 256 extra HCLK cycles are inserted
Bit 7: 4	ADDRHT	0xF	rw	Address-hold time 0000: 1 HCLK cycle is inserted 0001: 2 extra HCLK cycles are inserted …… 1111: 16 extra HCLK cycles are inserted
Bit 3: 0	ADDRST	0xF	rw	Address setup time 0000: 1 HCLK cycle is inserted 0001: 2 extra HCLK cycles are inserted …… 1111: 16 extra HCLK cycles are inserted

25.6.1.4 SRAM/NOR Flash write timing register x (XMC_BK1TMGWRx), x=1,2,3,4

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value.
Bit 29: 28	ASYNCM	0x0	rw	Asynchronous mode This field is valid only when the RWTD bit is enabled. 00: Mode A 01: Mode B 10: Mode C 11: Mode D
Bit 27: 20	Reserved	0xFF	resd	Kept at its default value.

Bit 19: 16	BUSLAT	0xF	rw	Bus latency To avoid data bus conflict, a latency is inserted on the data bus after one read operation in multiplexed or synchronous mode. 0000: 1 HCLK cycle is inserted 0001: 2 HCLK cycles are inserted …… 1111: 16 HCLK cycles are inserted
Bit 15: 8	DTST	0xFF	rw	Data setup time 00000000: 1 extra HCLK cycle is inserted 00000001: 2 extra HCLK cycles are inserted …… 11111111: 256 extra HCLK cycles are inserted
Bit 7: 4	ADDRHT	0xF	rw	Address-hold time 0000: 1 extra HCLK cycle is inserted 0001: 2 extra HCLK cycles are inserted …… 1111: 16 extra HCLK cycles are inserted
Bit 3: 0	ADDRST	0xF	rw	Address setup time 0000: 1 extra HCLK cycle is inserted 0001: 2 extra HCLK cycles are inserted …… 1111: 16 extra HCLK cycles are inserted

25.6.1.5 SRAM/NOR Flash extra timing register x (XMC_EXTx) (x=1,2,3,4)

Accessed by words.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 8	BUSLATR2R	0x08	rw	Bus turnaround phase for consecutive read duration This field is used to define the bus turnaround phase duration for consecutive read operations. A delay is inserted between two consecutive read operations in order to avoid bus conflicts. 00000000: 1 HCLK cycle is inserted for consecutive read operations 00000001: 2 HCLK cycles are inserted for consecutive read operations …… 00001000: 9 HCLK cycles are inserted for consecutive read operations (default value) …… 11111111: 256 HCLK cycles are inserted for consecutive read operations
Bit 7: 0	BUSLATW2W	0x08	rw	Bus turnaround phase for consecutive write duration This field is used to define the bus turnaround phase duration for consecutive write operations. A delay is inserted between two consecutive write operations in order to avoid bus conflicts. 00000000: 1 HCLK cycle is inserted for consecutive write operations 00000001: 2 HCLK cycles are inserted for consecutive write operations …… 00001000: 9 HCLK cycles are inserted for consecutive write operations (default value) …… 11111111: 256 HCLK cycles are inserted for consecutive write operations

25.6.2 SDRAM controller registers

25.6.2.1 SDRAM control register 1, 2 (SDRAM_CTRL1,SDRAM_CTRL2)

This register contains the control parameters for each SDRAM memory bank.

Bit	Name	Reset value	Type	Description
Bit 31: 15	Reserved	0x00000	resd	Kept at its default value.
Bit 14: 13	RD	0x0	rw	Read delay This field defines the delay (in HCLK clock cycles) for reading data after CAS latency. 00: No HCLK clock cycle delay 01: 1 HCLK clock cycle delay 10: 2 HCLK clock cycle delay 11: Reserved, do not use. Note: The corresponding bits in the CTRL2 register are “don’t care bit”
Bit 12	BSTR	0x0	rw	Burst read When this bit is set, it indicates that single AHB requests (single or burst) are processed as bursts. Several data will be prefetched and stored into the FIFO. 0: Single read requests are not processed as bursts 1: ingle read requests are always processed as bursts Note: The corresponding bits in the CTRL2 register are “don’t care bit”
Bit 11: 10	CLKDIV	0x0	rw	Clock division configuration SDRAM clock configuration. 00: SDCLK clock disabled 01: HCLK/4 10: HCLK/2 11: HCLK/3 Note: The corresponding bits in the CTRL2 register are “don’t care bit”. Note: when CLKDIV1 =1, CLKDIV must be set 0x0. In this case, SDCLK is the inverse of HCLK.
Bit 9	WRP	0x0	rw	Write protection This bit is set to enable the SDRAM write protection. 0: Write access allowed 1: Write access forbidden
Bit 8: 7	CAS	0x0	rw	CAS latency This field is used to select CAS latency. 00: Reserved, do not use. 01: 1 cycle 10: 2 cycles 11: 3 cycles
Bit 6	INBK	0x0	rw	Internal banks This bit is used to define the number of internal banks. 0: 2 internal BANKs 1: 4 internal BANKs
Bit 5: 4	DB	0x0	rw	SDRAM data bus This field enables 8-bit or 16-bit data bus width. 00: 8 bits 01: 16 bits 10: Reserved, do not use. 11: Reserved, do not use.
Bit 3: 2	RA	0x0	rw	Row address This field defines the number of a row address, including 11 bits, 12 bits and 13 bits. 00: 11 bits 01: 12 bits 10: 13 bits 11: Reserved, do not use.
Bit 1: 0	CA	0x0	rw	Column address This field defines the number of a column address, including 8 bits, 9 bits, 10 bits and 13 bits.

				00: 8 bits 01: 9 bits 10: 10 bits 11: 11 bits

25.6.2.2 SDRAM timing register 1, 2 (SDRAM_TM1,SDRAM_TM2)

This register contains the timing parameters of each SDRAM bank.

Bit	Name	Reset value	Type	Description
Bit 31: 28	Reserved	0x0	resd	Kept at its default value.
Bit 27: 24	TRCD	0xF	rw	Row active to Read/Write delay This field defines the delay between the activate command and a read/write command in number of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 23:20	TRP	0xF	rw	Precharge to active delay This field defines the delay between a precharge command and another command in number of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 19: 16	TWR	0xF	rw	Write Recovery delay This field defines the delay between a write command and a precharge command in number of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 15: 12	TRC	0xF	rw	Refresh to active delay This field defines the delay between the refresh command and the activate command, as well as the delay between two consecutive refresh commands. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 11: 8	TRAS	0xF	rw	Self refresh time This field defines the minimum self-refresh period in umber of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 7: 4	TXSR	0xF	rw	Exit Self-refresh to active delay This field defines the delay from exiting the self-refresh command to issuing an activate command in number of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles
Bit 3: 0	TMRD	0xF	rw	Load mode register program to active delay This field defines the delay between a load mode register command and an activate or refresh command in number of clock cycles. 0000: 1 cycle 0001: 2 cycles .... 1111: 16 cycles

Note: If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the

slowest devices.

25.6.2.3 SDRAM command register (SDRAM_CMD)

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value.
Bit 22: 9	MRD	0x0000	rw	Mode Register data Refer to the SDRAM specifications for details.
Bit 8: 5	ART	0x0	rw	Auto-refresh times This field defines the number of consecutive auto-refresh commands issued when MODE =“011”. 0000: 1 auto-refresh cycle 0001: 2 auto-refresh cycles .... 1110: 15 auto-refresh cycles 1111: Reserved
Bit 4	BK1	0x0	wo	SDRAM Bank 1 This bit indicates whether the command will be sent to the SDRAM Bank 1. 0: Command not sent to SDRAM Bank 1 1: Command sent to SDRAM Bank 1
Bit 3	BK2	0x0	wo	SDRAM Bank 2 This bit indicates whether the command will be sent to the SDRAM Bank 2. 0: Command not sent to SDRAM Bank 2 1: Command sent to SDRAM Bank 2
Bit 2: 0	CMD	0x0	wo	SDRAM Command This field defines the command issued to the SDRAM device. 000: Normal mode 001: Clock configuration enable 010: Precharge all banks 011: Auto refresh 100: Load mode register 101: Self refresh 110: Power-down command 111: Reserved

25.6.2.4 SDRAM refresh timer register (SDRAM_RCNT)

This register is used to set the refresh rate of the SDRAM, in number of SDRAM CLK clock cycles.

The RC has to be configured as a non-zero value in order to perform a correct refresh operation. The RC value cannot be changed after initialization.

Refresh operation has priority over a read/write operation. However, if a read/write operation is in progress while a new refresh request occurs, the refresh operation starts only after the read/write operation is complete. It is recommended that the RC value is smaller than the calculated value in order to ensure the anticipated refresh rate.

This register is shared by the SDRAM Bank 1 and Bank 2.

Bit	Name	Reset value	Type	Description
Bit 31: 15	Reserved	0x00000	resd	Kept at its default value.
Bit 14	ERIEN	0x0	rw	Error interrupt enable 0: Error interrupt disabled 1: Error interrupt enabled
Bit 13: 1	RC	0x0000	rw	Refresh counter This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of clock cycles. It must be set at least to 41 clock cycles. Refresh rate = (RC + 1) x SDRAM clock frequency RC = (SDRAM refresh period/number of rows) - 20
Bit 0	ERRC	0x0	wo	Error flag clear This bit is used to clear the error flag (ER) in the status register. 0: No effect

	1: Refresh error flag is cleared.

Note: The programmed COUNT value must not be equal to the sum of the following timings: *TWR+TRP+TRC+TRCD+4 memory clock cycles.*

25.6.2.5 SDRAM status register (SDRAM_STS)

Bit	Name	Reset value	Type	Description
Bit 31: 6	Reserved	0x0000000	resd	Kept at its default value.
Bit 5	BUSY	0x0	ro	Busy This bit indicates the status of the current SDRAM controller. 0: Idle 1: Busy
Bit 4: 3	BK2STS	0x0	ro	Bank2 status This field defines the status mode of the SDRAM Bank 2. 00: Normal mode 01: Auto-refresh mode 10: Power-down mode
Bit 2: 1	BK1STS	0x0	ro	Bank 1 Status This field defines the status mode of the SDRAM Bank 1. 00: Normal mode 01: Auto-refresh mode 10: Power-down mode
Bit 0	ERR	0x0	ro	Error flag 0: No refresh error has been detected 1: A refresh error has been detected

25.6.2.6 SDRAM clock one division register (SDRAM_ CLKDIV1)

Bit	Name	Reset value	Type	Description
Bit 31: 5	Reserved	0x0000000	resd	Kept at its default value.
Bit 5	CLKDIV1	0x0	rw	Clock one division SDRAM clock is divided by 1. 0：SDRAM clock is not divided by 1 1：SDRAM clock is divided by 1 Note: when CLKDIV1 =1, CLKDIV must be set 0x0. In this case, SDCLK is the inverse of the HCLK.
Bit 3:0	ERR	0x0	resd	Kept at its default value.

26 SDIO interface

26.1 SDIO introduction

The SD/SDIO MMC card host interface (SDIO) provides an interface between the AHB peripheral bus and MultiMediaCards (MMC), SD memory cards and SDIO cards.

SD memory card and SDI/O card system specifications are available through the SD card association website www.sdcard.org.

The MultiMediaCard system specifications published by the MMCA technical committee are available through the MultiMediaCard association website www.mmca.org.

26.2 SDIO main features

- Full compatibility with SD memory card specifications version 2.0
- Full compatibility with SDI/O card specification version 2.0 and support 1-bit and 4-bit databus modes.
- Full compatibility with MultiMedia card specification version 4.2 and support 1-bit, 4-bit and 8-bit databus modes.
- Full compatibility with previous versions of MultiMedia card specifications
- DMA transfers
- Data transfer up to 48 MHz in 8-bit bus mode
- Interrupt requests

Note: The SDIO is not compatible with SPI communication mode. It supports only one SD/SDIO/MMC 4.2 card at any one time.

Communication on the bus is based on command and data transfers.

- Command: A command is a token that starts an operation. Commands are sent from the host either to a single card (addressed command) or to all connected cards (broadcast command). Commands are transferred serially on the CMD line.
- Response: A response is a token that is sent from a card to the host as an answer to a previously received command. Responses are transferred serially on the CMD line.
- Data: Data can be transferred from the card to the host or vice versa. Data is transferred via the SDIO_D data line.

The basic operation on the MMC card/SD/SDI I/O bus is the command/response structure. These types of bus operation transfer their information through the command or bus mechanism. In addition, some operations have a data token.

Data transfers to/from SD/SDIO memory cards are done in data blocks. Data blocks are always followed by CRC bit, defining single and multiple block operations. Data transfers to/from MMC are done in data blocks or streams, as shown in Figure below.

Figure 26-1 SDIO “no response” and “response” operations

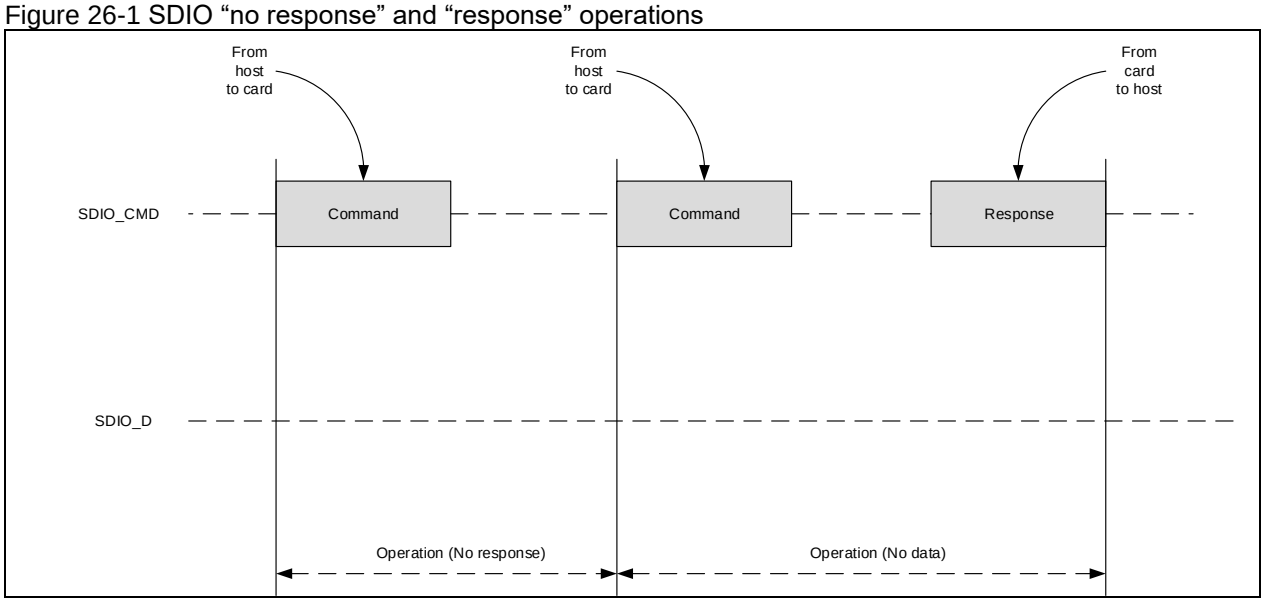


Figure 26-2 SDIO multiple block read operation

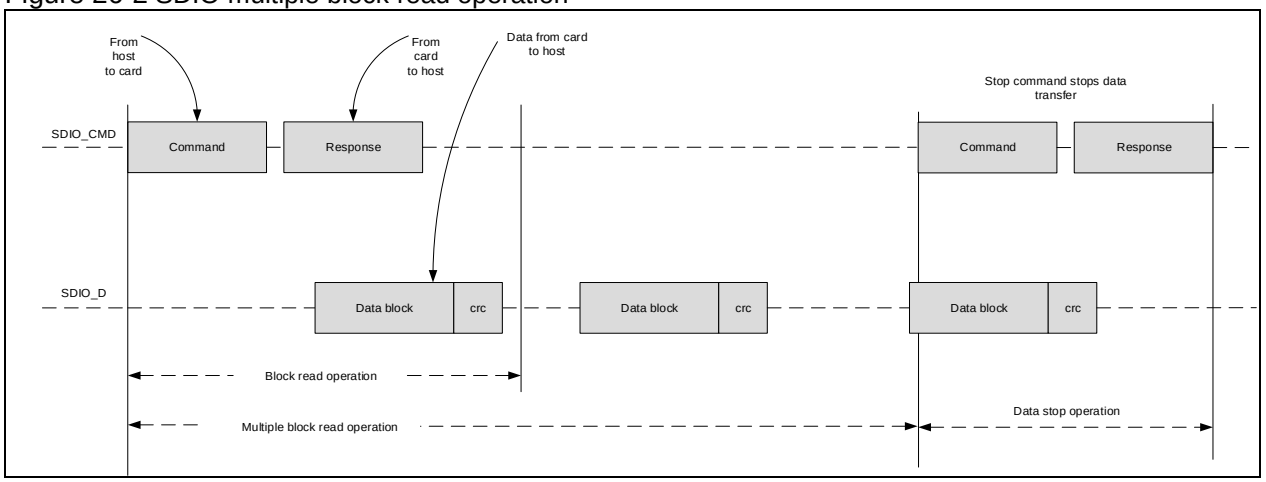
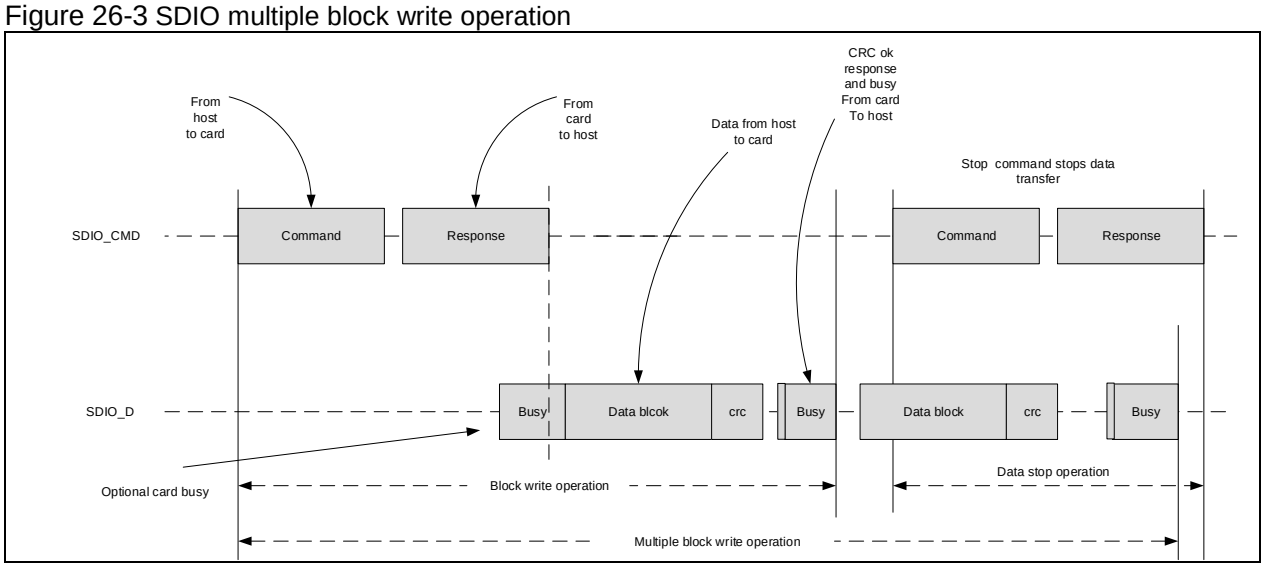


Figure 26-3 SDIO multiple block write operation



Note: The SDIO will not send any data as long as the Busy signal is set (SDIO_D0 pulled low).

Figure 26-4 SDIO sequential read operation

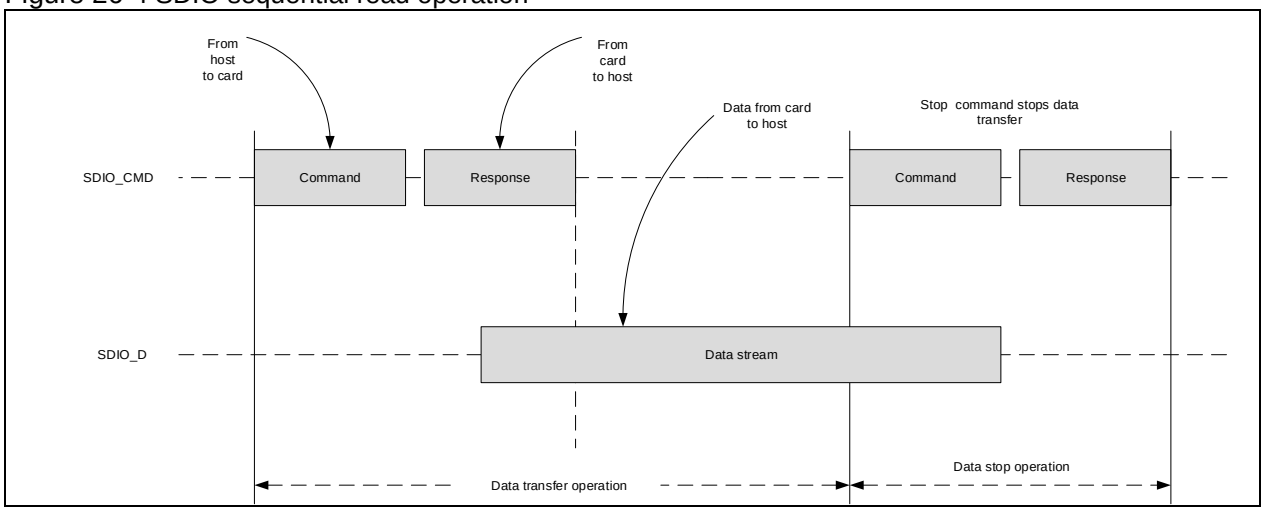
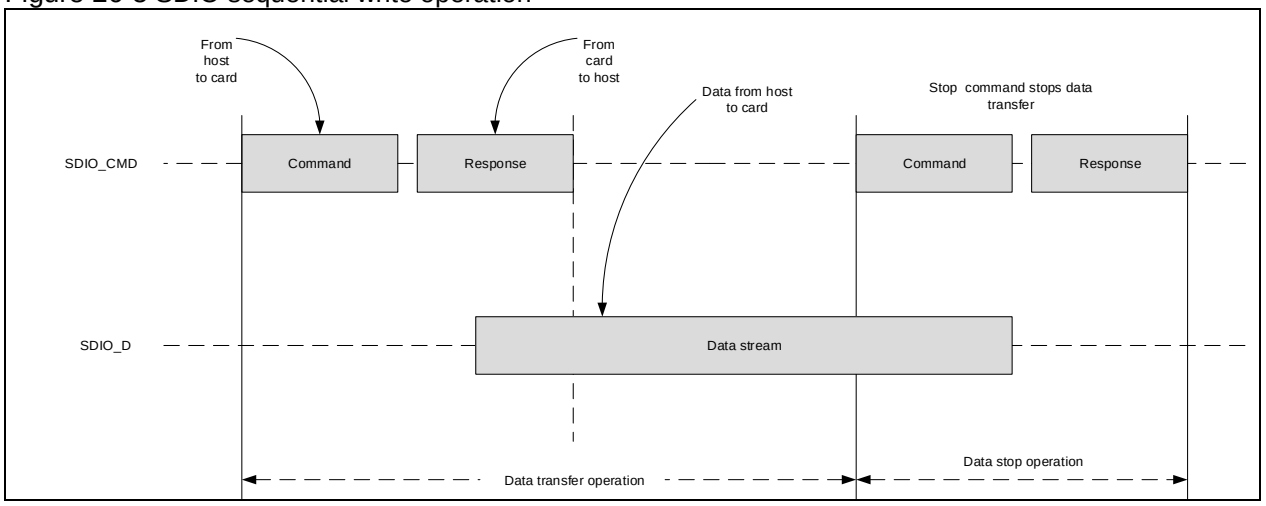


Figure 26-5 SDIO sequential write operation



26.3 Functional description

26.3.1 Card functional description

All the communications between the host and the cards are controlled by the card. The host sends two different types of commands: broadcast command and addressed (point-to-point) command.

- Broadcast command: applicable to all cards, some need responses
- Addressed command: sent to the addressed card, and responses received from the card

Memory card defines two types of operational modes:

- Card identification mode
- Data transfer mode

26.3.1.1 Card identification mode

In card identification mode, the host resets all cards, validates the operation voltage range, identifies cards and sets a relative card address (RCA) for each card on the CMD. All communications in the card identification mode use the command line (CMD).

Card identification process

The card identification process varies from card to card, and the host sends different commands. There are SD, SDI/O and MMC cards. It is possible to send a CMD5 command to identify the type of a card. If the host receives a response, it is a SDI/O card. If no response is received, the host will continue to send ACMD41 command, if a response is received, it is a SD card; otherwise a MMC card.

The card identification process is described as follows:

1. The bus is activated to confirm whether the card is connected or not. The clock frequency is at 0-400kHz during the card identification process.
2. The SDIO host sends a SD card, SDI/O card or MMC card.
3. Card Initialization
   - SD card: The SDIO host sends CMD2 (ALL_SEND_CID) to obtain its unique CID number. After receiving a response (CID number) from the card, the host will send CMD3 (SEND_RELATIVE_ADDR), instructing the card to issue the relative card address (RCA), which is shorter than the CID and is used to address the card in the data transfer mode.
   - SDI/O card: The SDIO host sends CMD3 (SEND_RELATIVE_ADDR) to instruct the card to release the relative card address (RCA), which is shorter than the CID and is used to address the card in the data transfer mode.
   - MMC card: The SDIO host sends CMD1 (SEND_OP_COND), followed by CMD2 and CMD3.
4. If the host wants to assign another RCA number, it can instruct the card to issue a new number by sending another CMD3 command. The last RCA is the actual RCA number of the card. The host repeats the card identification process (CMD2 and CMD3 cycle of each card)

26.3.1.2 Data transfer mode

The host will enter data transfer mode after identifying all cards on the bus. In data transfer mode, the host can operate cards within the range of 0 - 48MHz. It can send CMD9 (SEND_CSD) to get data specific to a card (CSD register) such as block length and car memory size. All communications between the host and the selected card are point-to-point transferred, and the CMD bus will confirm all addressed commands as a response. Data transfer read/write can be done in data block mode or stream mode, configured by the TFRMODE bit in the SDIO_DTCTRL register. In the data stream mode, data is transferred in bytes and without CRC appended to the end of each data block.

Wide bus selection/deselection

Wide bus (4-bit bus width) operation mode is selected or deselected by using ACMD6 (SET_BUS_WIDTH). The default bus width after power-up or CMD0 (GO_IDLE_STATE) is 1 bit. The ACMD6 is only valid in a transfer state, indicating that the bus width can be changed only after a card is selected by CMD7.

Stream read/write (MultiMedia card only)

Read**:**

1. The host sends CMD11 (READ_DAT_UNTIL_STOP) for stream read.
2. Until the host sends CMD12 (STOP_TRANSMISSION). The stop command has an execution delay due to the serial command transmission and the data transfers stops after the end bit of the stop command.

Write:

1. The host sends CMD20 (WRITE_DAT_UNTIL_STOP) for stream write.
2. Until the host sends CMD12 (STOP_TRANSMISSION). As the amount of data to be transferred is not determined in advance, the CRC cannot be used. When the memory range is reached, the command will be discarded by the card and remain in a transfer state, and a response is issued by setting the ADDRESS_OUT_OF_RANGE bit.

Data block read

In block read mode, the basic unit of data transfer is a block whose maximum size (fixed length 512 bytes) is defined in the CSD (READ_BL_LEN). If the READ_BL_PARTIAL is set, smaller blocks whose start and end addresses are entirely contained within 512 bytes may also be transmitted. A CRC is appended to the end of each block to ensuring data transfer integrity. Several commands related to data block read are as follows:

- CMD17 (READ_SINGLE_BLOCK): initiates a data block read and returns to the transfer state after the completion of the transfer.
- CMD18 (READ_MULTIPLE_BLOCK): starts a transfer of several consecutive data blocks.

Data blocks will keep transferring until the host sends CMD12(STOP_TRANSMISSION). The stop

command has an execution delay due to the serial command transmission and the data transfers stops after the end bit of the stop command.

Data block write

During block write (CMD24-27), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block. If the CRC failed, the card indicates a failure on the SDIO_D signal line and the transferred data are discarded and not written, and all transmitted data blocks are ignored.

If the host uses partial blocks with accumulated length is not block aligned, and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card will detect the block misalignment error before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR bit in the SDIO_STS register and waits the stop command in a receive state while ignoring all further data transfer. If the host attempts to perform write operation on a write-protected area, the write operation will be aborted. In this case, however, the card should set the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a block length setting in advance. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in the ROM, then the unchangeable part must match the corresponding part of the receive buffer. If this match failed, then the card will report an error and does not change any register contents. Some cards may require long and unpredictable times to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the SDIO_D signal line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host can check the status of the card with SEND_STATUS command (CMD13) at any time, and the card will respond with its status.

The READY_FOR_DATA status bit indicates whether the car can accept new data or whether the write process in still in progress. The host can deselect the card by issuing CMD7 (select another card), which will place the card in the disconnect state and release the SDIO_D line without interrupting the write operation. When reselecting the card, it will reactivate busy indication by pulling SDIO_D line to low if the programming is still in progress and the write buffer is unavailable.

26.3.1.3 Erase

The erasable unit of the MultiMedia card and SD card is the erase group. The erase group is calculated in write blocks, which are the basic writable units of the card. The size of the erase group is a parameter specific to the card and defined in the CSD.

The host can erase a contiguous range of erase groups. There are three steps to start the erase process, but the commands sent by the MultiMedia card and SD card are different.

1. The host defines the start address of the range using the following command:
   - SD card: issue CMD32 (ERASE_WR_BLK_START)
   - MMC car: issue CMD35 (ERASE_GROUP_START)
2. The host defines the end address of the rang using the following command:
   - SD card: issue CMD33 (ERASE_WR_BLK_END)
   - MMD car: issue CMD36 (ERASE_GROUP_END)
3. The host starts the erase process by sending CMD38 (ERASE)

26.3.1.4 Protection management

Three write protection methods are supported in the SDIO card host module to ensure that the protected data is not erased or changed.

Mechanical write protect switch

There is a mechanical sliding switch on the side of the card to allow the user to set/clear the write protection on the card. When the sliding switch is positioned with the window open, the card is write-protected, and when the window is closed, the card is not write-protected.

Internal card write protection

Card data can be protected against write and erase. The entire card can be permanently write-protected by the manufacturer or the content provider by setting the permanent or temporary write-protect bits in the CSD. For cards that support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in the CSD, part of the data can be protected, and the write protection can be changed by the

application. The SET_WRITE_PROT commands set the write protection of the addressed group. The CLR_WRITE_PROT commands clear the write protection of the addressed group. The SEND_WRITE_PROT command is similar to a single block read command. The card sends a data block containing 32 write protection bits (representing 32 write protect groups starting at the specified address) followed by 16 CRC bits. The address filed in the write protect commands is a group address in byte units.

Password protect

The password protection function enables the SDIO card host to lock and unlock a card with a password. The password is stored in the 128-bit PWD register and its size is set in the 8-bit PWD_LEN register. These registers are nonvolatile so that the content is not erased after power-off.

Locked cards can support certain commands, indicating that the host is allowed to reset, initialize and query for status, but not allowed to access data on the card. When the password is set (PWD_LEN is nonzero value), the card is locked automatically after power-up.

Like the CSD and CID register write commands, the lock/unlock commands are valid only in a transfer state. The command does not include an address parameter and thus the card must be selected before using it.

The card lock/unlock commands have the structure and bus transaction types of a regular single-block write command. The transferred data block include all the information required for the command (the password setting mode, PWD content and card lock/unlock indication). The command data block size is defined by the SDIO card host module before it sends the card lock/unlock command. The lock/unlock command structure is shown below:

Table 26-1 Lock/unlock command structure

Byte	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0	Reserved(set to 0)				ERASE	LOCK_UNLOCK	CLR_PWD	SET_PWD
1	PWDS_LEN							
2	password data							
…								
PWDS_LEN+1								

- ERASE: Setting it will force an erase operation. All other bits must be zero, and only the command byte is sent.
- LOCK_UNLOCK: Setting it will lock the card. Clearing it will unlock the card. LOCK_UNLOCK can be set simultaneously with SET_PWD, but not with the CLR_PWD.
- CLR_PWD: Setting it will clear the password data.
- SET_PWD: Setting this bit will save the password data to memory.
- PWD_LEN: This bit defines the length of the password in bytes. When the password is changed, the length is the combination of the old and new passwords.
- PWD: password (new or currently used, depending on the command)

The data block size should be defined by the host before it sends the card lock/unlock command. The block length should be equal to or greater than the data structure required for the lock/unlock command.

The following sections list the command sequence to set/clear a password, lock/unlock a card, and force an erase operation.

Setting the password

1. Select a card using CMD7 (SELECT/DESELECT_CARD), if none is selected previously
2. Define the block length with CMD16(SET_BLOCKLEN) to send in the 8-bit card lock/unlock mode, 8-bit PWD_LEN, and the number of bytes of the new password. When a password is replaced, the block size must take into account the length of both the old and the new passwords sent with the command.
3. Send CMD42(LOCK/UNLOCK) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the operation mode (SET_PWD=1), the length (PWD_LEN) and the password (PWD). When a password replacement is done, the length value includes the length of the both passwords, the old and the new one, and the PWD field includes the old password (currently used) followed by the new password.
4. When the old password is matched, the new password and its size are saved into the PWD

and PWD_LEN fields, respectively. When the old password sent is not correct (in size and/or content), the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register, and the old password is not changed. When the old password sent is correct (in size and content), the given new password and its size will be saved in the PWD and the PWD_LEN registers, respectively.

The password length field (PWD_LEN) indicates whether a password is currently set or not. When the field is a zero value, it means that a password is not set currently. When the field is nonzero, it means that a password is set and the card locks itself after power-up. It is possible to lock the card immediately in the current power session by setting the LOCK_UNLOCK bit or sending an additional card lock command.

Clearing the password

1. Select a card using CMD7 (SELECT/DESELECT_CARD), if none is selected previously.
2. Define the block length with CMD16(SET_BLOCKLEN) to send in the 8-bit card lock/unlock mode, 8-bit PWD_LEN, and the number of bytes of the new password.
3. Send CMD42(LOCK/UNLOCK) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the operation mode (SET_PWD=1), the length (PWD_LEN) and the password (PWD). When a password is matched, the PWD field is cleared and PWD_LEN is set to 0. If the password sent does not correspond to the expected password (in size and/or content), the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register, and the password is not changed.

Locking a card

1. Select a card using CMD7 (SELECT/DESELECT_CARD), if none is selected previously.
2. Define the block length with CMD16(SET_BLOCKLEN) to send in the 8-bit card lock/unlock mode, 8-bit PWD_LEN, and the number of bytes of the new password.
3. Send CMD42(LOCK/UNLOCK) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the operation mode (SET_PWD=1), the length (PWD_LEN) and the password (PWD).
4. When a password is matched, the card is locked and CARD_IS_LOCKED is set in the SDIO_STS register. If the password sent does not correspond to the expected password (in size and/or content), the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register, and the lock fails.

If the password is previously set (PWD_LEN is not 0), the card is locked automatically after power-on reset. An attempt to lock a locked card or to lock a card that does not have a password fails and the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register.

Unlocking a card

1. Select a card using CMD7 (SELECT/DESELECT_CARD), if none is selected previously
2. Define the block length with CMD16(SET_BLOCKLEN) to send in the 8-bit card lock/unlock mode, 8-bit PWD_LEN, and the number of bytes of the new password.
3. Send CMD42(LOCK/UNLOCK) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the operation mode (SET_PWD=1), the length (PWD_LEN) and the password (PWD).
4. When a password is matched, the card is unlocked and CARD_IS_LOCKED is cleared in the SDIO_STS register. If the password sent does not correspond to the expected password (in size and/or content), the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register, and the lock remains locked.

The unlocking function is valid only for the current power session. The card is locked automatically on the next power-up as long as the PWD field is not cleared.

An attempt to unlock an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register.

Forcing erase

If the user forgot the password (PWD content), it is possible to access the card after clearing all the data on the card. This forced erase operation will erase all card data and all password data.

1. Select a card using CMD7 (SELECT/DESELECT_CARD), if none is selected previously
2. Define the block length with CMD16(SET_BLOCKLEN) to send in the 8-bit card lock/unlock mode, 8-bit PWD_LEN, and the number of bytes of the new password.
3. Send CMD42(LOCK/UNLOCK) with the appropriate data block size on the data line including the 16-bit CRC. The data block indicates the operation mode (ERASE =1). All other bits must be zero.
4. When the ERASE bit is the only bit in the data field, all card content will be erased, including the PWD and the PWD_LEN field, and the card is no longer locked. When any other bits are not zero, the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register, and the card data are retained, and the card remains locked.

An attempt to force erase an unlocked card fails and the LOCK_UNLOCK_FAILED error bit is set in the SDIO_STS register.

26.3.2 Commands and responses

26.3.2.1 Commands

Command types

Four commands are available to control the SD memory card:

1. Broadcast command: sent to all cards, no responses returned
2. Broadcast command with response: sent to all cards, responses received from all cards simultaneously
3. Addressed command: sent to the selected card, and no data transfer on the SDIO_D line
4. Addressed data transfer command: sent to the selected card, and data transfer is present on the SDIO_D line

Command description

The SDIO host module system is designed to provide a standard interface for a variety of application types. In the meantime, specific customer/application features must also be taken into account. Because of this, two types of general commands are defined in the standard: general commands (GEN_CMD) and application-specific commands (ACMD).

To use the application-specific commands, the SDIO host must send CMD55(APP_CMD) first, and waits the response from the card, which indicates that the APP_CMD bit is set and an ACMD is expected, before sending ACMD.

Table 26-2 Commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD0	bc	[31: 0]=stuff bits	-	GO_IDLE_ STATE	Reset all cards to idle state
CMD1	bc	[31: 0]=OCR	R3	SEND_OP_ COND	In idle state, request a card to send OCR register content through the CMD bus
CMD2	bcr	[31: 0]=stuff bits	R2	ALL_SEND_CID	Request all cards to send CID data through the CMD bus
CMD3	bcr	[31: 0]=stuff bits	R6	SEND_RELATIV E_AD DR	Request a card to issue a new relative card address
CMD4	bc	[31: 16]=DSR [15: 0]= stuff bits	-	SET_DSR	Set the DSR register of all cards
CMD5	bcr	[31: 24]Reserved [23: 0] I/O OCR	R4	IO_SEND_OP_C OND	Used only for the SDI/O card to query the voltage range of the required I/O card

CMD6	ac	[31: 26] set to 0 [25: 24] access [23: 16] index [15: 8] value [7: 3] set to 0 [2: 0] command set	R1b	SWITCH	Used only for the MMC card to switch the operation modes or modify the EXT_CSD register
CMD7	ac	[31: 16]=RCA [15: 0]= stuff bits	R1b	SELECT/DESELECT_ CARD	This command is used to switch a card between the standby state and the send state, or between the programmed and the disconnected state. The relative card address is used to select a card. Address 0 is used to deselect the card.
CMD8 (SD)	bcr	[31: 12] Reserved [11: 8]operating voltage (VHS) [7: 0]check mode	R7	SEND_IF_ COND	Send the host power supply voltage to the SD card and check whether the card supports the voltage or not.
CMD8 (MMC)	adtc	[31: 0]= stuff bits	R1	SEND_EXT_ CSD	Used only for the MMC card to send its own EXT_CSD register as a data block
CMD9	ac	[31: 16]=RCA [15: 0]= stuff bits	R2	SEND_CSD	The selected card sends CSD (card-specific data) through the CMD bus
CMD10	ac	[31: 16]=RCA [15: 0]= stuff bits	R2	SEND_CID	The selected card sends CID (card flag) through the CMD bus
CMD12	ac	[31: 0]= stuff bits	R1b	STOP_TRANSMISSION	Force the card to stop transmission
CMD13	ac	[31: 16]=RCA [15: 0]= stuff bits	R1	SEND_ STATUS	Selected card send status register
CMD15	ac	[31: 16]=RCA [15: 0]= stuff bits	-	GO_INACTIVE_ STATE	Selected card switch to inactive state

Table 26-3 Data block read commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD16	ac	[31: 0]=data block length	R1	SET_ BLOCKLEN	This command is used to set the length of data blocks (in bytes) for all block commands. The default value is 512 bytes.
CMD17	adtc	[31: 0]=data address	R1	READ_ SINGLE_ BLOCK	Read a data block of the size set by CMD16
CMD18	adtc	[31: 0]=data address	R1	READ_ MULTIPLE_ BLOCK	Continuously read data from the card to the host until the STOP_TRANSMISSION is received

Table 26-4 Data stream read/write commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD11	adtc	[31: 0]= data address	R1	READ_DAT_ UNTIL_STOP	Read data stream form the card starting from a given address until the STOP_TRANSMISSION is received.
CMD20	adtc	[31: 0]= data address	R1	WRITE_DAT_ UNTIL_STOP	Read data stream form the host starting from a given address until the STOP_TRANSMISSION is received.

Table 26-5 Data block write commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD16	ac	[31: 0]=data block length	R1	SET_ BLOCKLEN	This command is used to set the length of data blocks (in bytes) for all block commands. The default value is 512 bytes
CMD23	ac	[31: 16]=set to 0 [15: 0]=data block size	R1	SET_BLOCK_ COUNT	Define the number of blocks to be transferred in the data block read/write that follows
CMD24	adtc	[31: 0]=data address	R1	WRITE_ BLOCK	Write a data block of the size set by the CMD16
CMD25	adtc	[31: 0]=data address	R1	WRITE_ MULTIPLE BLOCK	Continuously write data blocks until the STOP_TRANSMISSION is received
CMD26	adtc	[31: 0]= stuff bits	R1	PROGRAM_ CID	Program the card identification register
CMD27	adtc	[31: 0]= stuff bits	R1	PROGRAM_ CSD	Program the programmable bits of the CSD

Table 26-6 Block-based write protect commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD28	ac	[31: 0]= data address	R1b	SET_WRITE_ PROT	If the card has write protection features, this command sets the write protection bit of the specified group. The properties of write protection are placed in the card-specific area (WP_GRP_SIZE).
CMD29	ac	[31: 0]= data address	R1b	CLR_WRITE_PR OT	If the card has write protection features, this command clears the write protection bit of the specified group.
CMD30	adtc	[31: 0]=write protect data address	R1	SEND_WRITE_P ROT	If the card has write protection features, this command asks the card to send the status of the write protection bits.

Table 26-7 Erase commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD32 … CMD34	Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMedia card.				
CMD35	ac	[31: 0]=data address	R1	ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase
CMD36	ac	[31: 0]=data address	R1	ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase
CMD37	Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMedia card.				
CMD38	ac	[31: 0]=stuff bits	R1b	ERASE	Erase all previously selected data blocks

Table 26-8 I/O mode commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD39	ac	[31: 16]=RCA [15]=register write flag [14: 8]=register address [7: 0]=register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command specifies a card and a register and provides the data for writing if the write flag is set. The R4 response contains data read from the specified register. This command accesses application-specific registers that are not defined in the MultiMedia card standard.
CMD40	bcr	[31: 0]=stuff bits	R5	GO_IRQ_STATE	Place the system in the interrupt mode

Table 26-9 Card lock commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD42	adtc	[31: 0]=stuff bits	R1	LOCK_ UNLOCK	Sets/clears the password or locks/unlocks the card. The size of the data block is set by CMD16.

Table 26-10 Application-specific commands

CMD index	Type	Parameter	Response format	Abbreviation	Description
CMD55	ac	[31: 16]=RCA [15: 0]=stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application-specific command rather than a standard command.
CMD56	adtc	[31: 1]=stuff bits [0]=RD/WR	R1	GEN_CMD	Used either to transfer a data block to the card or to read a data block from the card for general-purpose/application-specific commands. The size of the data block is defined by the SET_BLOCK_LEN command.
CMD57 … CMD59	Reserved.				
CMD60 … CMD63	Reserved for manufacturer				

26.3.2.2 Response formats

All responses are sent via the CMD bus. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The length depends on the response type.

A response always starts with a start bit (always 0), followed by the transmission bit indicating the direction of transmission (card =0). A value denoted by – in the tables below stands for a variable entry. All responses, except the R3 response type, are protected by a CRC. Every command code word is terminated with the end bit (always 1).

26.3.2.2.1 R1 (normal response command)

Code length = 48 bits. The 45:40 bits indicate the index of the command to be responded to. This value is interpreted as a binary-coded number (between 0 and 63). The status of the card is coded in 32 bits. Note that if it involves a data transfer to a card, a busy signal may appear on the data line after each data block is transmitted. The host should check the busy signal after data block transfer.

Table 26-11 R1 response

Bit	47	46	[45: 40]	[39: 8]	[7: 1]	0
Field width	1	1	6	32	7	1
Value	0	0	-	-	-	1
Description	Start bit	Transmission bit	Command index	Card status	CRC7	End bit

26.3.2.2.2 R1b

It is the same as R1 with an optional busy signal transmitted on the data line. The card may become busy after receiving these commands based on its state prior to the command reception. The host should check the busy signal.

26.3.2.2.3 R2 (CID & CSD registers)

Code length = 136 bits. The contents of the CID register are sent as a response to the CMD2 and CMD10 commands. The contents of the CSD register are sent as a response to the CMD9. Only the bits [127 ... 1] of the CID and CSD are transmitted, and the reserved bit [0] of these registers is replaced with the end bit of the response.

Table 26-12 R2 response

Bit	135	134	[133 : 128]	[127 : 1]	0
Field width	1	1	6	127	1
Value	1	0	111111	-	1
Description	Start bit	Transmission bit	Reserved	CID or CSD register	End bit

26.3.2.2.4 R3 (OCR register)

Code length = 48 bits. The contents of the OCR register are sent as a response to ACMD41.

Table 26-13 R3 response

Bit	**47**	**46**	**[45 : 40]**	**[39: 8]**	**[7: 1]**	**0**
Field width	1	1	6	32	7	1
Value	1	0	111111	-	111111	1
Description	Start bit	Transmission bit	Reserved	OCR register	Reserved	End bit

26.3.2.2.5 R4 (Fast I/O)

Code length = 48 bits. The parameter field contains the RCA of the specified card, the register address to be read out or written to, and its contents.

Table 26-14 R4 response

Bit	**47**	**46**	**[45: 40]**	**[39: 8]**			**[7: 1]**	**0**
Field width	1	1	6	16	8	8	7	1
Value	1	0	100111	-	-	-	-	1
Description	Start bit	Transmission bit	CMD39	RCA	Register address	Read register contents	CRC7	End bit

26.3.2.2.6 R4b

For SD I/O only, an SDIO card will respond with a unique SDIO response R4 after receiving the CMD5.

Table 26-15 R4b response

Bit	**47**	**46**	**[45: 40]**			**[39: 8]**			**[7: 1]**	**0**
Field width	1	1	6	1	3	1	3	24	7	1
Value	1	0	-	-	-	-	-	-	-	1
Description	Start bit	Tx bit	Res.	Card ready	Number of I/O functions	Current memory	Stuff bit	I/O OCR	Res.	End bit

26.3.2.2.7 R5 (interrupt request)

For MultiMedia card only. Code length = 48 bits. If the response is generated by the host, the RCA field in the parameter will be 0x0.

Table 26-16 R5 response

Bit	**47**	**46**	**[45: 40]**	**[39: 8]**		**[7: 1]**	**0**
Field width	1	1	6	16	16	7	1
Value	1	0	101000	-	-	-	1
Description	Start bit	Start bit	Tx bit	RCA[31:16] of a successful card or of the host	Not defined. Maybe used for interrupt data	CRC7	End bit

26.3.2.2.8 R6 (interrupt request)

For SD I/O card only. This is a normal response to CMD3 by a memory device.

Table 26-17 R6 response

Bit	47	46	[45: 40]		[39: 8]	[7: 1]	0
Field width	1	1	6	16	16	7	1
Value	1	0	000011	-	-	-	1
Description	Start bit	Tx bit	CMD3	RCA[31:16] of a successful card or of the host	Card status	CRC7	End bit

The card status bit [23: 8] will be changed when the CMD3 is sent to an I/O-only card. In this case, the 16 bits of response are the SD I/O-only values.

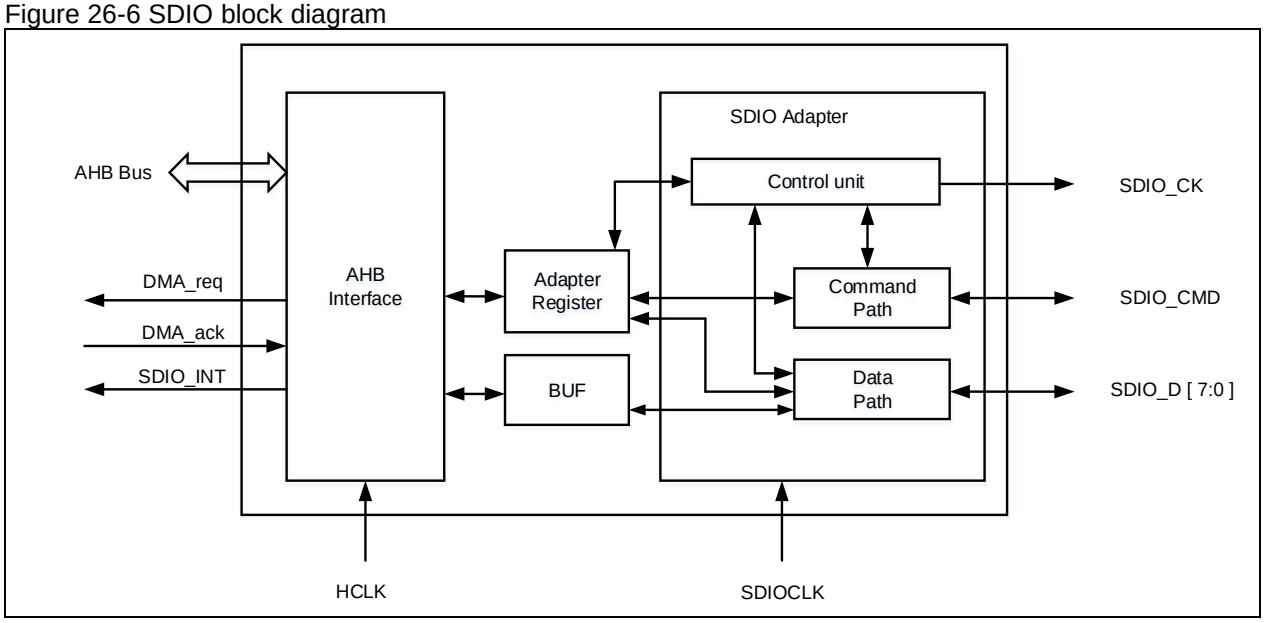
- Bit 15=COM_CRC_ERROR
- Bit 14=ILLEGAL_COMMAND
- Bit 13=ERROR
- Bit [12: 0]=Reserved

26.3.3 SDIO functional description

SDIO consists of four parts:

- SDIO adapter block: contains a control unit, command path and data path that provides all functions specific to the MMC/SD/SD I/O card such as the clock generation, command and data transfer
  - Control unit: manages and generates clock signals
  - Command path: manages command transfer
  - Data path: manages data transfer
- AHB interface: generates interrupt and DMA request signals
- Adapter register: system register
- BUF: used for data transfer

Figure 26-6 SDIO block diagram



26.3.3.1 SDIO adapter

SDIO_CK is a clock to the MultiMedia/SD/SDIO car provided by the host. One bit of command or data is transferred on both command and data lines with each clock cycle. The clock frequency can very between different cards and different protocols.

- MultiMedia card
  - V3.31 protocol 0 – 20MHz
  - V4.0/4.2 protocol 0 – 48MHz
- SD card
  - 0 – 48MHz
- SD I/O card
  - 0 – 48MHz

SDIO_CMD is a bidirectional command channel and used for the initialization of a card and command transfer. When the host sends a command to a card, the card will issue a response to the host. The SDIO_CMD has two operational modes:

- Open-drain mode for initialization (only for MMCV3.31 or previous)
- Push-pull mode for command transfer (SD/SD I/O card and MMC V4.2 also use push-pull drivers for initialization)

SDIO_D [7:0] is a bidirectional data channel. After initialization, the host can change the width of the data bus. After reset, the SDIO_D0 is used for data transfer by default. MMCV3.31 or previous supports only one bit of data line, so only SDIO_DO can be used.

The table below is used for the MultiMedia card/SD/SD I/O card bus:

Table 26-18 SDIO pin definitions

Pin	Direction	Description
SDIO_CK	Output	MultiMedia card/SD/SDIO card clock. This pin is the clock from the host to a card.
SDIO_CMD	Bidirectional	MultiMedia card/SD/SDIO card command. This pin is the bidirectional command/response signal.
SDIO_D[7: 0]	Bidirectional	MultiMedia card/SD/SDIO card data. This pin is the bidirectional databus.

Control unit

The control unit consists of a power management sub-unit and a clock management sub-unit. The power management subunit is controlled by the SDIO_PWRCTRL register. The PS bit is used to define power-up/power-off state. During the power-off and power-up phases, the power management subunit will disable the card bus output signals. The clock management subunit is controlled by the SDIO_CLKCTRL

register where the CLKDIV bit is used to define the divider factor between the SDIOCLK and the SDIO output clock. If BYPSEN = 0, the SDIO_CK output signal is driven by the SDIOCLK divided according to the CLKDIV bit; if BYPSEN = 1, the SDIO_CK output signal is directly driven by the SDIOCLK. The HFCEN is set to enable hardware flow control feature in order to avoid the occurrence of an error at transmission underflow or reception overflow. The PWRSVEN bit can be set by software to enable power save mode, and the SDIO_CK can be output only when the bus is active.

Command path

The command path unit sends commands to and receives responses from the cards. When the CCSMEN bit is set in the SDIO_CMDCTRL register, a command transfer starts. First sends a command to a card by the SDIO_CMD, the command length is 48 bits. The data on the SDIO_CMD is synchronized with the rising edge of the SDIO_CK. A block of data is transferred with each SDIO_CK, including start bit, transfer bit, command index defined by the SDIO_CMDCTRL_CMDIDX bit, parameters defined by the SDIO_ARG, 7-bit CRC and end bit. Then receives responses from the card. There are two response types: 48-bit short response and 136-bit long response. Both use CRC error check. The received responses are saved in the area from SDIO_RSP1 to SDIO_RSP4. The command path can generate command flag, which can be defined by the SDIO_STS register.

Table 26-19 Command formats

Bit	**47**	**46**	**[45: 40]**	**[39: 8]**	**[7: 1]**	**0**
Width	1	1	6	32	7	1
Value	0	1	-	-	-	1
Description	Start bit	Tx bit	Command index	Parameter	CRC7	End bit

Response: A response is sent from a specified card to the host (or synchronously from all cards for MMCV3.31 or previous), as an answer to a previously received command. Responses are transferred serially on the CMD line.

Table 26-20 Short response format

Bit	**47**	**46**	**[45: 40]**	**[39: 8]**	**[7: 1]**	**0**
Width	1	1	6	32	7	1
Value	0	0	-	-	-	1
Description	Start bit	Tx bit	Command index	Parameter	CRC7 (or 1111111)	End bit

Table 26-21 Long response format

Bit	**135**	**134**	**[133: 128]**	**[127: 1]**	**0**
Width	1	1	6	127	1
Value	0	0	111111	-	1
Description	Start bit	Tx	Reserved	CID or CSD (including internal CRC7)	End bit

Table 26-22 Command path status flags

Flag	Description
CMDRSPCMPL	A response is already received (CRC OK)
CMDFAIL	A command response is already received (CRC fails)
CMDCMPL	A command is sent (does not require a response)
CMDTIMEOUT	Command response timeout (64 SDIO_CK cycles)
DOCMD	Command transfer is in progress

Command channel state machine (CCSM)

When the CCSMEN bit is set in the SDIO_CMDCTR register, command transfer starts. When the command has been sent, the command channel state machine (CCSM) will set the status flags and enters the idle state if a response is not required. When a response is received, the received CRC code and the internally generated CRC code are compared, and the appropriate status flags are set.

- The CCSM remains in the idle state for at least 8 SDIO_CK cycles to meet the Ncc (the minimum delay between two host commands) and NRC (the minimum delay between the host command and the card response).
- When the wait state is entered, the command timer is enabled. If the NCR timeout (response time to a command), that is, 64 SDIO_CK periods, is reached before the CCSM moves to the receive state, the timeout flag is set (CMDTIMEOUT) and the idle state is entered.

If the interrupt bit is set in the command register, the timer is disabled and the CCSM waits for an interrupt request from one card. If the pending bit is set in the command register, the CCSM will enter pend state and wait for a CmdPend signal from the data path subunit. When detecting the CmdPend, the CCSM goes to the send state, which triggers the data counter to send the stop command.

Figure 26-7 Command channel state machine (CCSM)

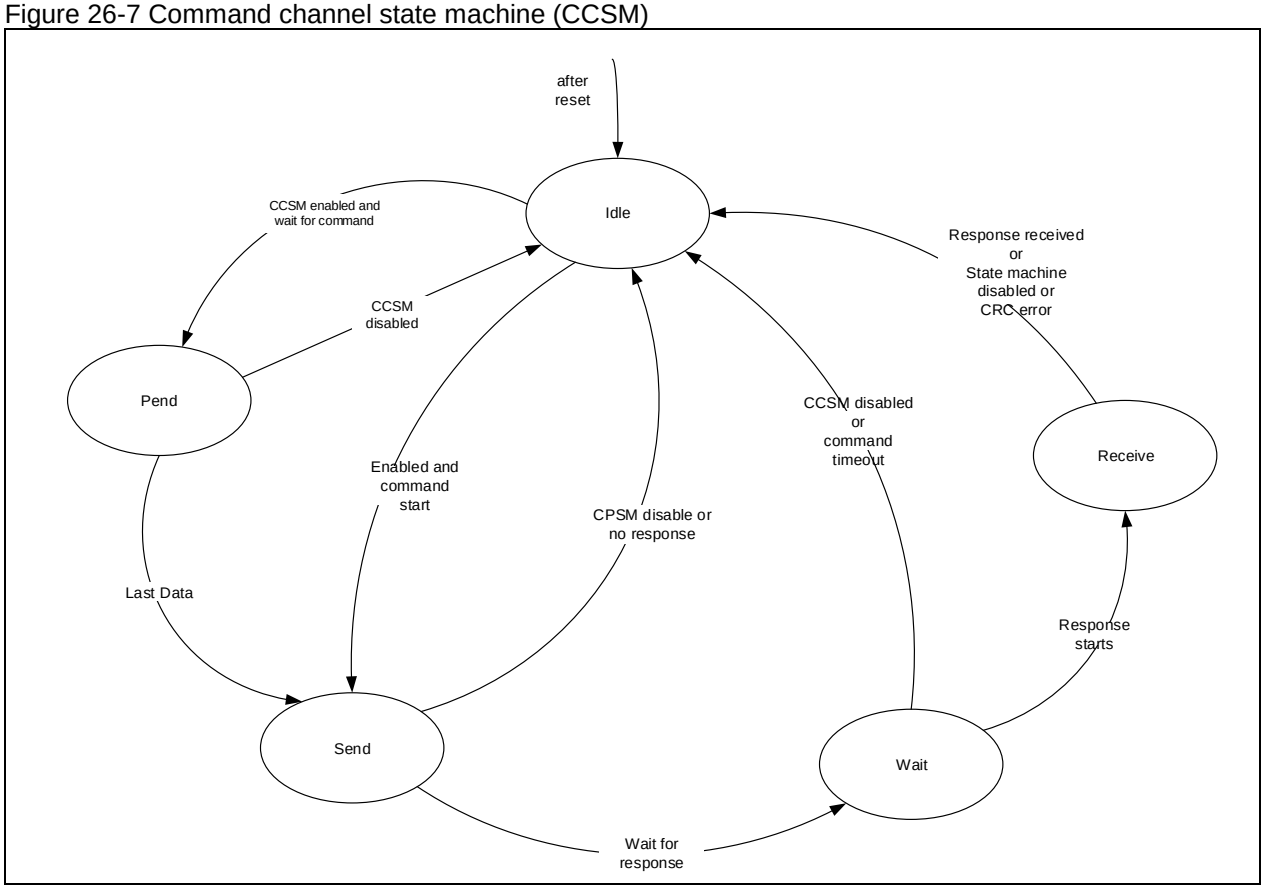
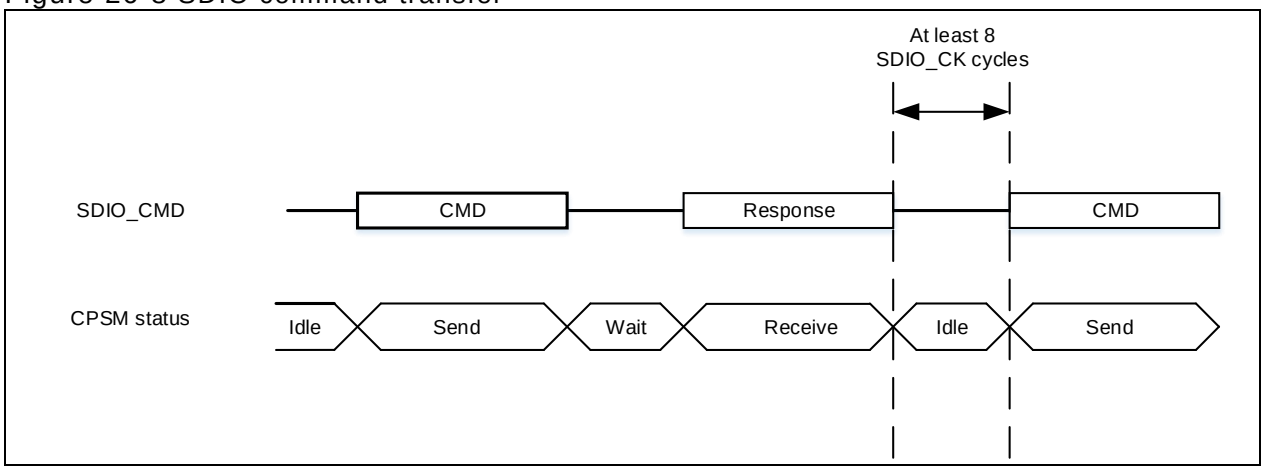


Figure 26-8 SDIO command transfer



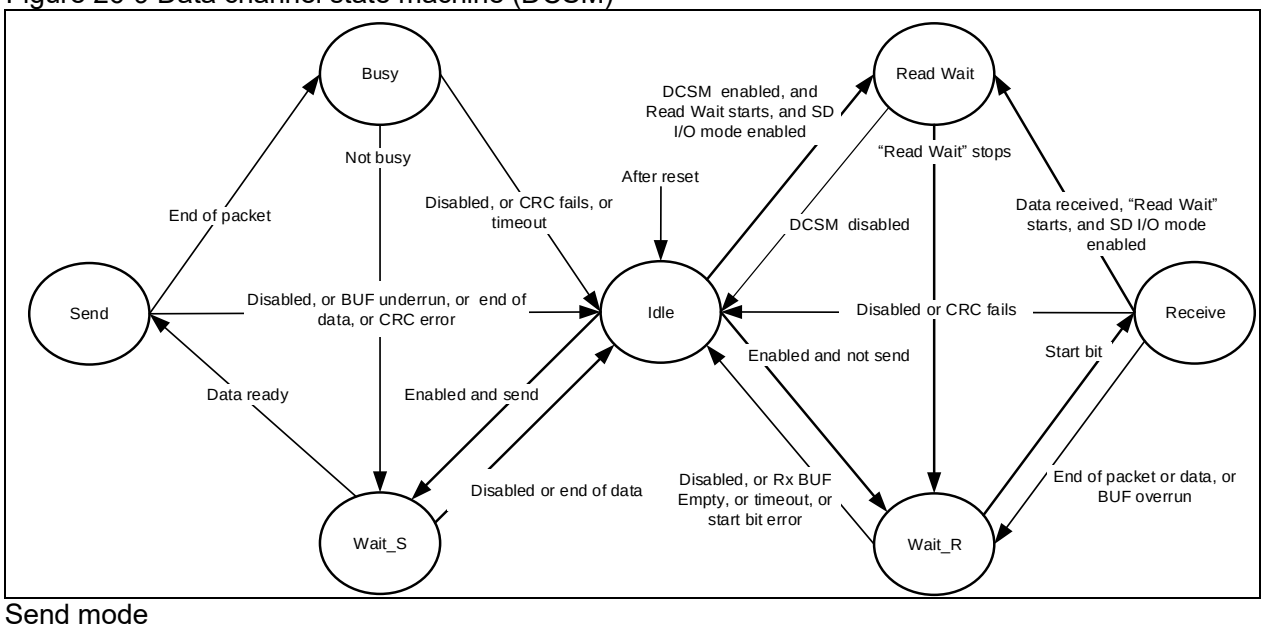
Data channel

The data path subunit transfers data between the host and the cards. The databus width can be configured using the BUSWS bit in the SDIO_CLKCTRL register. By default, only the SDIO_DO signal line is used for transfer. Only one bit of data is transferred with each clock cycle. If the 4-bit wide bus mode is selected, four bits are transferred per clock cycle over the SDIO_D [3:0] signal line. If the 8-bit wide bus mode is selected, eight bits are transferred per clock cycle over the SDIO_D [7: 0] signal line. The TFRDIR bit is set in the SDIO_DTCTR register to define the transfer direction. If TFRDIR=0, it indicates that the data is transferred from the controller to the card; if TFRDIR=1, it indicates that the data is transferred from the card to the controller. The TFRMODE bit can be used to select block data transfer or stream transfer for the MultiMedia card. If the TFREN bit is set, data transfer starts. Depending on the TFRDIR bit, the DCSM enters Wait_S or Wait_R state.

Data channel state machine (DCSM)

The DCSM has seven states, in send and receive mode, as shown in the Figure below:

Figure 26-9 Data channel state machine (DCSM)



Send mode

- Idle: The data channel is inactive, either in the Wait_S or the Wait_R state.
- Wait_S: Waits until the BUF flag becomes empty or the data transmission is completed. The DCSM must remain in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements where the N_{WR} is the interval between the reception of the card response and the start of the data transfer from the host.
- Send: The DCSM sends data to a card, and the data transfer mode can be either block or stream, depending on the SDIO_DTCTRL_TFRMODE bit. If an overflow error occurred, the DCSM then moves to the idle state

- Busy: The DCSM waits for the CRC flag. If the DCSM receives a correct CRC status and is not busy, it will enter the Wait_S state. If it does not receive a correct CRC status or a timeout occurs while the DCSM is in the busy state, a CRC fail flag or timeout flag is generated.
- Wait_R: The start bit of the Wait_R. If a timeout occurs before it detects a start bit, the DCSM moves to the idle state and generates a timeout flag.
- Receive: Data is received from a card and written to the BUF. The data transfer mode can be either block or stream, depending on the SDIO_DTCTRL _TFRMODE bit. If an overflow error occurs, it then returns to Wait_R state.

Table 26-23 Data token formats

Description	Start bit	Data	CRC16	End bit
Block data	0	-	Y	1
Stream data	0	-	N	1

26.3.3.2 Data BUF

The data BUF contains a transmit and receive unit. It is a 32-bit wide and 32-word deep data buffer. Because the data BUF operates in the AHB clock domain (HCLK), all signals connected to the SDIO clock domain (SDIOCLK) are resynchronized.

- Transmit BUF: Data can be written to the transmit BUF via the AHB interface when the SDIO transmission feature is enabled.

The transmit BUF has 32 sequential addresses. It contains a data output register that holds the data word pointed by the read pointer. When the data path has loaded its shift register, it moves its read pointer to the next data and outputs data.
If the transmit BUF is disabled, all status flags are inactive. The data path sets the DOTX when it transmits data.

- Receive BUF: When the data path receives a data word, it will write the data to the BUF. The write pointer is incremented automatically after the end of the write operation. On the other side, a read pointer always points to the current data in the BUF. If the receive BUF is disabled, all status flags are cleared, and the read and write pointers are reset as well. The data path sets the DORX when it receives data.

26.3.3.3 SDIO AHB interface

The AHB interface generates the interrupt and DMA requests, and access the SDIO interface registers and the data BUF.

SDIO interrupts

The interrupt logic generates an interrupt request when one of the selected status flags is high. The SDIO_INTEN register is used to select the conditions that will generate an interrupt.

SDIO/DMA interface: data transfer process between the SDIO and memory

In the following examples, data is transferred from the host to the card. The SDIO BUF is filled with data stored in a memory through the DMA controller.

1. Card identification process
2. Increase the SDIO_CK frequency
3. Select a card by sending CMD7
4. Enable the DMA2 controller and clear all interrupt flag bits, configure the DMA2 channel4 source address register as the memory buffer's base address, and the DMA2 channel4 destination address register as the SDIO_BUF register address. Then configure the DMA2 channel4 control register (memory increment, non-peripheral increment, and peripheral and source data width is word width). Finally enable DMA2 channel4.
5. Send CMD24 (WRITE_BLOCK) as follows:

   Program the SDIO data length register (SDIO_DTLEN), the BLKSIZE bit in the SDIO data control register (SDIO_DTCTRL), and the SDIO parameter register (SDIO_ARG) with the address of the card where data is to be transferred, and program the SDIO command register (SDIO_CMD), enable the CCSMEN bit, wait for SDIO_STS [6]=CMDRSPCMPL interrupt,

and then program the SDIO data control register (SDIO_DTCTRL): TFREN=1 (enable the SDIO card host to send data), TFRDIR=0 (from the controller to the card), TFRMODE=0 (block data transfer), DMAEN=1 (enable DMA), BLKSIZE=9 (512 bytes), and wait from SDIO_STS [10]=DTBLKCMPL.

6. Check that no channels are still enabled by confirming the DMA channel enable SDIO status register (SDIO_STS)

26.3.3.4 Hardware flow control

The HFCEN bit is set in the SDIO_CLKCTRL register to enable hardware flow control, which is sued to avoid BUF underflow and overflow errors. Read/write access to the BUF is still active even if flow control is enabled.

26.3.4 SDIO I/O card-specific operations

The SDIO can support the following operations (except read suspend, for it does not require specific hardware operation) when the SDIO_DTCTRL [11] is set.

SDIO read wait operation by SDIO_D2 signal lines

The optional read wait operation is used only for SD card 1-bit or 4-bit mode. The read wait operation can instruct the host to stop data transfer temporarily while the host is reading from multiple registers (IO_RW_EXTENDED, CMD53), and also allows the host to send commands to other functions in the SD I/O device in order to start a read wait process after the reception of the first data block. The detailed process as follows:

- Enable data path (SDIO_DTCTRL [0] = 1)
- Enable SDIO-specific operation (SDIO_DTCTRL [11] = 1)
- Start read wait (SDIO_DTCTRL [10]=0 and SDIO_DTCTRL [8]=1)
- Data direction is from a card to the SDIO host (SDIO_DTCTRL [1]=1)
- The data unit in the SDIO adapter will enter read wait state, and drive the SDIO_D2 to 0 after 2 SDIO_CK cycles
- The data unit starts waiting to receive data from a card. The DCSM will not enter read wait even if read wait start is set. The read wait process will start after the CRC is received. The RDWTSTOP has to be cleared to start a new read wait operation.

During the read wait period, the SDIO host can detect the SDIO interrupts over the SDIO_D1.

SDIO read wait operation by stopping clock

If the SDIO card does not support the mentioned above read wait operation, the SDIO can enter a read wait by stopping SDIO_CK, described as follows:

- Enable data path (SDIO_DTCTRL [0] = 1)
- Enable SDIO-specific operation (SDIO_DTCTRL [11] = 1)
- Start read wait (SDIO_DTCTRL [10]=0 and SDIO_DTCTRL [8]=1)

The DCSM stops the clock two SDIO_CK cycles after the end bit of the current received data block and starts the clock again after the read wait end bit is set.

Note that as the SDIO_CK is stopped, the SDIO host cannot send any command to the card. The SDIO host can detect the SDIO interrupt over the SDIO_D1.

SDIO suspend/resume operation (write and read operation suspend)

To free the bus to provide higher-priority transfers for other functions or memories, the host can suspend data transfer to certain functions or memories. As soon as the higher-priority transfer is completed, the previous transfer operation will restart at the suspended location.

While sending data to a card, the SDIO can suspend the write operation. The SDIO_CMD [11] bit is set and indicates to the CCSM that the current command is a suspend command. The CCSM analyzes the response and when an ACK signal is received from the card (suspend accepted), it acknowledges the DCSM that enters the idle state after receiving the CRC of the current data block.

SDIO interrupts

There is a pin with interrupt feature on the SD interface in order to enable the SD I/O card to interrupt the MultiMedia card/SD module. In 4-bit SD mode, this pin is SDIO_D1. The SD I/O interrupts are detected when the level is active. In other words, the interrupt signal line must be active (low) before it is recognized and responded by the MultiMedia card/SD module, and will remain inactive (high) at the end of the interrupt routine.

When the SDIO_DTCTRL [11] bit is set, the SDIO interrupts are detected on the SDIO_D1 signal line.

26.4 SDIO registers

The device communicates with the system through 32-bit control registers accessible via AHB.

The peripheral registers must be accessed by words (32-bit).

Table 26-24 A summary of the SDIO registers

Register	Offset	Reset value
SDIO_PWRCTRL	0x00	0x0000 0000
SDIO_CLKCTRL	0x04	0x0000 0000
SDIO_ARG	0x08	0x0000 0000
SDIO_CMD	0x0C	0x0000 0000
SDIO_RSPCMD	0x10	0x0000 0000
SDIO_RSP1	0x14	0x0000 0000
SDIO_RSP2	0x18	0x0000 0000
SDIO_RSP3	0x1C	0x0000 0000
SDIO_RSP4	0x20	0x0000 0000
SDIO_DTTMR	0x24	0x0000 0000
SDIO_DTLEN	0x28	0x0000 0000
SDIO_DTCTRL	0x2C	0x0000 0000
SDIO_DTCNTR	0x30	0x0000 0000
SDIO_STS	0x34	0x0000 0000
SDIO_INTCLR	0x38	0x0000 0000
SDIO_INTEN	0x3C	0x0000 0000
SDIO_BUFCNTR	0x48	0x0000 0000
SDIO_BUF	0x80	0x0000 0000

26.4.1 SDIO power control register (SDIO_ PWRCTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 2	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 1: 0	PS	0x0	rw	Power switch These bits are set or cleared by software. They are used to define the current status of the card clock. 00: Power-off, the card clock is stopped. 01: Reserved 10: Reserved 11: Power-on, the card clock is started.

Note: Write access to this register is not allowed within seven HCLK clock periods after data is written.

26.4.2 SDIO clock control register (SDIO_ CLKCTRL)

The SDIO_CLKCTRL register controls the SDIO_CK output clock.

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16: 15	CLKDIV	0x0	rw	Clock division This field is set or cleared by software. It defines the clock division relations between the SDIOCLK and the SDIO_CK: SDIO_CK frequency=SDIOCLK / [CLKDIV[9: 0] + 2].
Bit 14	HFCEN	0x0	rw	Hardware flow control enable This bit is set or cleared by software. 0: Hardware flow control disabled 1: Hardware flow control enabled Note: When hardware flow control is enabled, refer to the SDIO_STS register for the meaning of the TXBUF_E and RXBUF_F interrupt signals.
Bit 13	CLKEGS	0x0	rw	SDIO_CK edge selection This bit is set or cleared by software. 0: SDIO_CK generated on the rising edge of the master clock SDIOCLK 1: SDIO_CK generated on the falling edge of the master clock SDIOCLK
Bit 12: 11	BUSWS	0x0	rw	Bus width selection This bit is set or cleared by software. 00: Default bus mode, SDIO_D0 used 01: 4-bit bus mode,SDIO_D[3: 0] used 10: 8-bit bus mode, SDIO_D[7: 0] used
Bit 10	BYPSEN	0x0	rw	Clock divider bypass enable bit This bit is set or cleared by software. When disabled, the SDIO_CK output signal is driven by the SDIOCLK that is divided according to the CLKDIV value. When enabled, the SDIO_CK output signal is directly driven by the SDIOCLK. 0: Clock divider bypass disabled 1: Clock divider bypass enabled
Bit 9	PWRSVEN	0x0	rw	Power saving mode enable This bit is set or cleared by software. When disabled, the SDIO_CK is always output; when enabled, the SDIO_CK is only output when the bus is active. 0: Power saving mode disabled 1: Power saving mode enabled
Bit 8	CLKOEN	0x0	rw	Clock output enable This bit is set or cleared by software. 0: Clock output disabled 1: Clock output enabled
Bit 7: 0	CLKDIV	0x00	rw	Clock division This field is set or cleared by software. It defines the clock division relations between the SDIOCLK and the SDIO_CK: SDIO_CK frequency=SDIOCLK / [CLKDIV[9: 0] + 2].

Note: 1. While the SD/SDIO card or MultiMedia car is in identification mode, the SDIO_CK frequency must be less than 400kHz.

2. When all card are assigned with relative card addresses, the clock frequency can be changed to the maximum card frequency.

3. This register cannot be written within seven HCLK clock periods after data is written. The SDIO_CK can be stopped during the read wait period for SD I/O cards. In this case, the SDIO_ CLKCTRL register does not control the SDIO_CK.

26.4.3 SDIO argument register (SDIO_ARG)

The SDIO_ARG register contains 32-bit command argument, which is sent to a card as part of a command.

Bit	Name	Reset value	Type	Description
Bit 31: 0	ARGU	0x0000 0000	rw	Command argument Command argument is sent to a card as part of a command. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

26.4.4 SDIO command register (SDIO_CMD)

The SDIO_CMD register contains the command index and command type bits. The command index is sent to a card as part of a command. The command type bits control the command channel state machine (CCSM).

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value.
Bit 11	IOSUSP	0x0	rw	SD I/O suspend command This bit is set or cleared by software. If this bit is set, the command to be sent is a suspend command (used for SDIO only). 0: SD I/O suspend command disabled 1: SD I/O suspend command enabled
Bit 10	CCSMEN	0x0	rw	Command channel state machine (CCSM) enable bit This bit is set or cleared by software. 0: Command channel state machine disabled 1: Command channel state machine enabled
Bit 9	PNDWT	0x0	rw	CCSM Waits for ends of data transfer (CmdPend internal signal) This bit is set or cleared by software. If this bit is set, the CCSM waits for the end of data transfer before it starts sending a command. 0: Disabled 1: Enabled
Bit 8	INTWT	0x0	rw	CCSM waits for interrupt request This bit is set or cleared by software. If this bit is set, the CCSM disables command timeout and waits for an interrupt request. 0: Disabled 1: Enabled
Bit 7: 6	RSPWT	0x0	rw	Wait for response bits This bit is set or cleared by software. This bit indicates whether the CCSM is to wait for a response, and if yes, it will indicates the response type. 00: No response 01: Short response 10: No response 11: Long response
Bit 5: 0	CMDIDX	0x00	rw	Command index The command index is sent to a card as part of a command.

Note: 1.This register cannot be written within seven HCLK clock periods after data is written.
2.MultiMedia card can sent two types of responses: 48-bit short response or 136-bit short response. The SD card and SD I/O card can send only short responses, and the argument can vary according to the type of response. The software will distinguish the type of response according to the command sent.

26.4.5 SDIO command response register (SDIO_RSPCMD)

The SDIO_RSPCMD register contains the command index of the last command response received. If the command response transmission does not contain the command index (long or OCR response), the SDIO_RSPCMD field is unknown, although it should have contained 111111b (the value of the reserved field from a response)

Bit	Name	Reset value	Type	Description
Bit 31: 6	Reserved	0x0000000	resd	Kept at its default value.
Bit 5: 0	RSPCMD	0x00	ro	Response command index This field contains the command index of the command response received.

26.4.6 SDIO response 1..4 register (SDIO_RSPx)

The SDIO_RSPx (x=1..4) register contains the status of a card, which is part of the response received.

Bit	Name	Reset value	Type	Description
Bit 31: 0	CARDSTSx	0x0000 0000	ro	See Table 23-25

The card status size is 32 or 127 bits, depending on the response type.

Table 26-25 Response type and SDIO_RSPx register

Register	Short response	Long response
SDIO_RSP1	Card status [31: 0]	Card status [127: 96]
SDIO_RSP2	Unused	Card status [95: 64]
SDIO_RSP3	Unused	Card status [63: 32]
SDIO_RSP4	Unused	Card status [31: 1]

The most significant bit of the card status is always received first. The least significant bit of the SDIO_RSP4 register is always 0.

26.4.7 SDIO data timer register (SDIO_DTTMR)

The SDIO_DTTMR register contains the data timeout period in the unit of card bus clock periods. A counter loads the value from the SDIO_DTTMR register and starts decrementing when the DCSM enters the Wait_R or busy state. If the counter reaches 0 while the DCSM is in either of these states, a timeout status flag will be set.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TIMEOUT	0x0000 0000	rw	Data timeout period Data timeout period in card bus clock cycles.

Note: A data transfer must be written to the SDIO_DTCNTR and the SDIO_DTLEN register before being written to the SDIO data control register (SDIO_DTCTRL).

26.4.8 SDIO data length register (SDIO_DTLEN)

The SDIO_DTLEN register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

Bit	Name	Reset value	Type	Description
Bit 31: 25	Reserved	0x00	resd	Kept at its default value.
Bit 24: 0	DTLEN	0x0000000	rw	Data length value Number of data bytes to be transferred.

Note: For a block data transfer, the value in the SDIO_DTLEN must be a multiple of the block data size.

A data transfer must be written to the SDIO_DTCNTR and the SDIO_DTLEN register before being written to the SDIO data control register (SDIO_DTCTRL).

26.4.9 SDIO data control register (SDIO_DTCTRL)

The SDIO_DTCTRL register controls the data channel statue machine (DCSM).

Bit	Name	Reset value	Type	Description
Bit 31: 12	Reserved	0x00000	resd	Kept at its default value.
Bit 11	IOEN	0x0	rw	SD I/O enable functions This bit is set or cleared by software. If the bit is set, the DCSM performs an SD IO card-specific operation. 0: Disabled 1: Enabled
Bit 10	RDWTMODE	0x0	rw	Read wait mode This bit is set or cleared by software. If disabled, the SDIO_D2 controls the read wait; if enabled, the SDIO_CK controls the read wait. 0: Disabled 1: Enabled
Bit 9	RDWTSTOP	0x0	rw	Read wait stop This bit is set or cleared by software. While the RDWTSTART is set, If this bit is set, it indicates that read wait is stopped; if this bit cleared, it indicates that the read wait is in progress. 0: Read wait is in progress if the RDWTSTART is set. 1: Read wait is stopped if the RDWTSTART is set.
Bit 8	RDWTSTART	0x0	rw	Read wait start This bit is set or cleared by software. When this bit is set, read wait starts; when this bit is cleared, no actions occurs. 0: Read wait disabled 1: Read wait enabled
Bit 7: 4	BLKSIZE	0x0	rw	Data block size This bit is set or cleared by software. This field defines the length of data block when the block data transfer is selected. 0000: block length = 2^0 = 1 byte 0001: block length = 2^1 = 2 bytes 0010: block length = 2^2= 4 bytes 0011: block length = 2^3= 8 bytes 0100: block length = 2^4= 16 bytes 0101: block length = 2^5= 32 bytes 0110: block length = 2^6= 64 bytes 0111: block length = 2^7= 128 bytes 1000: block length = 2^8= 256 bytes 1001: block length = 2^9= 512 bytes 1010: block length = 2^{10}= 1024 bytes 1011: block length = 2^{11}= 2048 bytes 1100: block length = 2^{12}= 4096 bytes 1101: block length = 2^{13}= 8192 bytes 1110: block length = 2^{14} = 16384 bytes 1111: Reserved
Bit 3	DMAEN	0x0	rw	DMA enable bit This bit is set or cleared by software. 0: Disabled 1: Enabled
Bit 2	TFRMODE	0x0	rw	Data transfer mode selection This bit is set or cleared by software. If this bit is set, it indicates stream data transfer; if this bit cleared, it indicates block data transfer. 0: Disabled 1: Enabled
Bit 1	TFRDIR	0x0	rw	Data transfer direction selection This bit is set or cleared by software. If this bit is set, data transfer is from a card to a controller; if this bit is cleared, data transfer is from a controller to a card. 0: Disabled

				1: Enabled
Bit 0	TFREN	0x0	rw	Data transfer enabled bit This bit is set or cleared by software. If this bit is set, data transfer starts. The DCSM enters the Wait_S or Wait_R state, depending on the direction bit TFRDIR. The DCSM goes to the read wait state if the RDWTSTART bit is set from the beginning of the transfer. It is not necessary to clear the enable bit after the end of data transfer but the SDIO_DTCTRL must be updated to enable a new data transfer. 0: Disabled 1: Enabled

Note: This register cannot be written within seven HCLK clock periods after data is written.

26.4.10 SDIO data counter register (SDIO_DTCNTR)

The SDIO_DTCNTR register loads the value from the SDIO_DTLEN register when the DCSM moves from the idle state to the Wait_R or Wait_S state. During the data transfer, the counter value decrements to 0, and then the DCSM enters the idle state and sets the data status end flag bit DTCMPL.

Bit	Name	Reset value	Type	Description
Bit 31: 25	Reserved	0x00	resd	Kept at its default value.
Bit 24: 0	CNT	0x0000000	ro	Data count value When this register is read, the number of data bytes to be transferred is returned. Write access has no effect.

Note: This register can be read only when the data transfer is complete.

26.4.11 SDIO status register (SDIO_STS)

The SDIO_STS is a read-only register, containing two types of flags:

- Static flags (bits [23: 22, 10: 0]): These bits can be cleared by writing to the SDIO_INTCLR register.
- Dynamic flags (bit [21: 11]): These bit status changes with the state of the corresponding logic (for example, BUT full or empty flag is set or cleared as data written to the BUF).

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value.
Bit 22	IOIF	0x0	ro	SD I/O interrupt received
Bit 21	RXBUF	0x0	ro	Data available in receive BUF
Bit 20	TXBUF	0x0	ro	Data available in transmit BUF
Bit 19	RXBUFE	0x0	ro	Receive BUF empty
Bit 18	TXBUFE	0x0	ro	Transmit BUF empty If hardware flow control is enabled, the TXBUF_E signal becomes valid when the BUF contains two words.
Bit 17	RXBUFF	0x0	ro	Receive BUF full If hardware flow control is enabled, the RXBUF_F becomes valid two words before the BUF is full.
Bit 16	TXBUFF	0x0	ro	Transmit BUF full
Bit 15	RXBUFH	0x0	ro	Receive BUF half full There are at least 8 words in the BUF. This flag bit can be used as DMA request.
Bit 14	TXBUFH	0x0	ro	Transmit BUF half empty: At least 8 words can be written to the BUF. This flag bit can be used as DMA request.
Bit 13	DORX	0x0	ro	Data receive in progress
Bit 12	DOTX	0x0	ro	Data transmit in progress
Bit 11	DOCMD	0x0	ro	Command transfer in progress
Bit 10	DTBLKCMPL	0x0	ro	Data block sent/received CRC check passed)
Bit 9	SBITERR	0x0	ro	Start bit not detected on all data signals in wide bus mode
Bit 8	DTCMPL	0x0	ro	Data end (data counter, SDIO CNT, is zero)
Bit 7	CMDCMPL	0x0	ro	Command sent (no response required)
Bit 6	CMDRSPCMPL	0x0	ro	Command response (CRC check passed)
Bit 5	RXERRO	0x0	ro	Received BUF overrun error

Bit 4	TXERRU	0x0	ro	Transmit BUF underrun error
Bit 3	DTTIMEOUT	0x0	ro	Data timeout
Bit 2	CMDTIMEOUT	0x0	ro	Command response timeout The command timeout is a fixed value of 64 SDIO_CK clock periods.
Bit 1	DTFAIL	0x0	ro	Data block sent/received (CRC check failed)
Bit 0	CMDFAIL	0x0	ro	Command response received (CRC check failed)

26.4.12 SDIO clear interrupt register (SDIO_INTCLR)

The SDIO_INTCLR is a read-only register. Writing 1 to the corresponding register bit will clear the correspond bit in the SDIO_STS register.

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value.
Bit 22	IOIF	0x0	rw	SD I/O interface flag clear bit This bit is set by software to clear the IOIF flag.
Bit 21: 11	Reserved	0x000	resd	Kept at its default value.
Bit 10	DTBLKCMPL	0x0	rw	DTBLKCMPL flag clear bit This bit is set by software to clear the DTBLKCMPL flag.
Bit 9	SBITERR	0x0	rw	SBITERR flag clear bit This bit is set to clear the SBITERR flag.
Bit 8	DTCMPL	0x0	rw	DTCMPL flag clear bit This bit is set by software to clear the DTCMPL flag.
Bit 7	CMDCMPL	0x0	rw	CMDCMPL flag clear bit This bit is set by software to clear the CMDCMPL flag.
Bit 6	CMDRSPCMPL	0x0	rw	MDRSPCMPL flag clear bit This bit is set by software to clear the CMDRSPCMPL flag.
Bit 5	RXERRO	0x0	rw	RXERRO flag clear bit This bit is set by software to clear the RXERRO flag.
Bit 4	TXERRU	0x0	rw	TXERRU flag clear bit This bit is set by software to clear the TXERRU flag.
Bit 3	DTTIMEOUT	0x0	rw	DTTIMEOUT flag clear bit This bit is set by software to clear the DTTIMEOUT flag.
Bit 2	CMDTIMEOUT	0x0	rw	CMDTIMEOUT flag clear bit This bit is set by software to clear the CMDTIMEOUT flag.
Bit 1	DTFAIL	0x0	rw	DTFAIL flag clear bit This bit is set by software to clear the DTFAIL flag.
Bit 0	CMDFAIL	0x0	rw	CMDFAIL flag clear bit This bit is set by software to clear the CMDFAIL flag.

26.4.13 SDIO interrupt mask register (SDIO_INTEN)

The SDIO_INTEN register determines which status bit generates an interrupt by setting the corresponding bit.

Bit	Name	Reset value	Type	Description
Bit 31: 23	Reserved	0x000	resd	Kept at its default value.
Bit 22	IOIFIEN	0x0	rw	SD I/O mode received interrupt enable This bit is set or cleared by software to enable/disable the SD I/O mode received interrupt function. 0: Disabled 1: Enabled
Bit 21	RXBUFIEN	0x0	rw	Data available in RxBUF interrupt enable This bit is set or cleared by software to enable/disable the Data Available in RxBUF Interrupt. 0: Disabled 1: Enabled
Bit 20	TXBUFIEN	0x0	rw	Data available in TxBUF interrupt enable This bit is set or cleared by software to enable/disable the Data Available in TxBUF Interrupt. 0: Disabled 1: Enabled
Bit 19	RXBUFEIEN	0x0	rw	RxBUF empty interrupt enable This bit is set or cleared by software to enable/disable the

				RxBUF empty interrupt. 0: Disabled 1: Enabled
Bit 18	TXBUFEIEN	0x0	rw	TxBUF empty interrupt enable This bit is set or cleared by software to enable/disable the TxBUF empty interrupt. 0: Disabled 1: Enabled
Bit 17	RXBUFFIEN	0x0	rw	RxBUF full interrupt enable This bit is set or cleared by software to enable/disable the RxBUF full interrupt. 0: Disabled 1: Enabled
Bit 16	TXBUFFIEN	0x0	rw	TxBUF full interrupt enable This bit is set or cleared by software to enable/disable the TxBUF full interrupt. 0: Disabled 1: Enabled
Bit 15	RXBUFHIEN	0x0	rw	RxBUF half full interrupt enable This bit is set or cleared by software to enable/disable the RxBUF half full interrupt. 0: Disabled 1: Enabled
Bit 14	TXBUFHIEN	0x0	rw	TxBUF half empty interrupt enable This bit is set or cleared by software to enable/disable the TxBUF half empty interrupt. 0: Disabled 1: Enabled
Bit 13	DORXIEN	0x0	rw	Data receive acting interrupt enable This bit is set or cleared by software to enable/disable the Data receive acting interrupt. 0: Disabled 1: Enabled
Bit 12	DOTXIEN	0x0	rw	Data transmit acting interrupt enable This bit is set or cleared by software to enable/disable the Data transmit acting interrupt. 0: Disabled 1: Enabled
Bit 11	DOCMDIEN	0x0	rw	Command acting interrupt enable This bit is set or cleared by software to enable/disable the Command acting interrupt. 0: Disabled 1: Enabled
Bit 10	DTBLKCMPLIEN	0x0	rw	Data block end interrupt enable This bit is set or cleared by software to enable/disable the Data block end interrupt. 0: Disabled 1: Enabled
Bit 9	SBITERRIEN	0x0	rw	Start bit error interrupt enable This bit is set or cleared by software to enable/disable the Start bit error interrupt. 0: Disabled 1: Enabled
Bit 8	DTCMPLIEN	0x0	rw	Data end interrupt enable This bit is set or cleared by software to enable/disable the Data end interrupt. 0: Disabled 1: Enabled
Bit 7	CMDCMPLIEN	0x0	rw	Command sent interrupt enable This bit is set or cleared by software to enable/disable the Command sent interrupt. 0: Disabled 1: Enabled
Bit 6	CMDRSPCMPLIEN	0x0	rw	Command response received interrupt enable

				This bit is set or cleared by software to enable/disable the Command response received interrupt. 0: Disabled 1: Enabled
Bit 5	RXERROIEN	0x0	rw	RxBUF overrun error interrupt enable This bit is set or cleared by software to enable/disable the RxBUF overrun error interrupt. 0: Disabled 1: Enabled
Bit 4	TXERRUIEN	0x0	rw	TxBUF underrun error interrupt enable This bit is set or cleared by software to enable/disable the TxBUF underrun error interrupt. 0: Disabled 1: Enabled
Bit 3	DTTIMEOUTIEN	0x0	rw	Data timeout interrupt enable This bit is set or cleared by software to enable/disable the Data timeout interrupt. 0: Disabled 1: Enabled
Bit 2	CMDTIMEOUTIEN	0x0	rw	Command timeout interrupt enable This bit is set or cleared by software to enable/disable the Command timeout interrupt. 0: Disabled 1: Enabled
Bit 1	DTFAILIEN	0x0	rw	Data CRC fail interrupt enable This bit is set or cleared by software to enable/disable the Data CRC fail interrupt. 0: Disabled 1: Enabled
Bit 0	CMDFAILIEN	0x0	rw	Command CRC fail interrupt enable This bit is set or cleared by software to enable/disable the Command CRC fail interrupt. 0: Disabled 1: Enabled

26.4.14 SDIOBUF counter register (SDIO_BUFCNTR)

The SDIO_BUFCNTR register contains the number of words to be written to or read from the BUF. The BUF counter loads the value from the SDIO_DTLEN register when the data transfer bit TFREN is set in the SDIO_DTCTRL register. If the data length is not word-aligned, the remaining 1 to 3 bytes are regarded as a word.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x00	resd	Kept at its default value.
Bit 23: 0	CNT	0x000000	ro	Number of words to be written to or read from the BUF.

26.4.15 SDIO data BUF register (SDIO_BUF)

The receive and data BUF is group of a 32-bit wide registers that can be written or read. The BUF contains 32 registers on 32 sequential addresses. The CPU can use BUF for read/write multiple operations.

Bit	Name	Reset value	Type	Description
Bit 31: 0	DT	0x0000 0000	rw	Receive and transmit BUF data The BUF data occupies 32x 32-bit words, the address: SDIO base + 0x80 to SDIO base + 0xFC

27 Ethernet media access control (EMAC)

This module applies to AT32F457 series only, but not AT32F455/456 series.

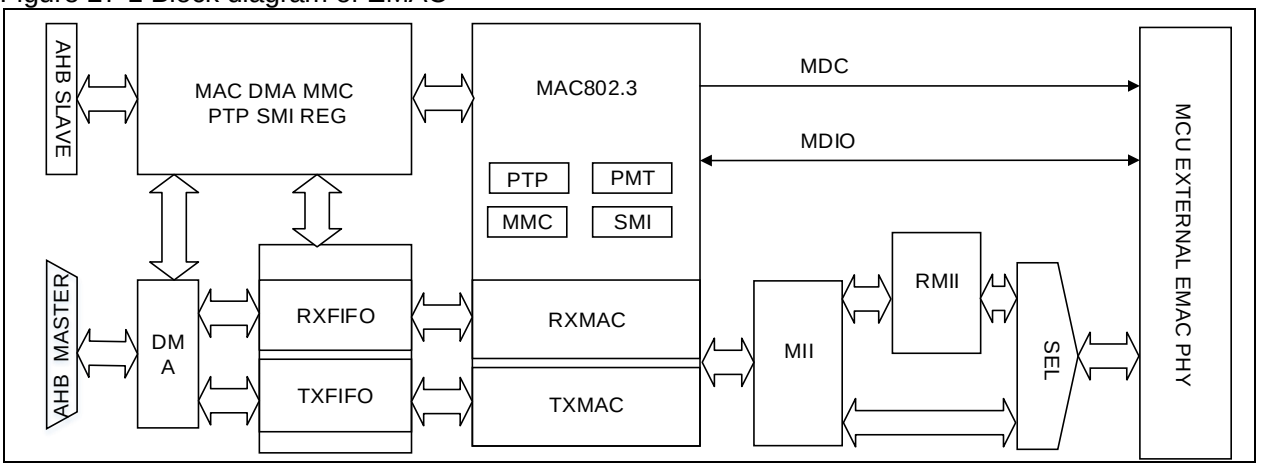
27.1 EMAC introduction

The Ethernet peripheral enables the AT32F457 to transmit and receive data (10/100Mbps) through Ethernet in compliance with IEEE 802.3-2002 standard.

The AT32F457 Ethernet peripheral supports two standard interfaces to the external PHY: media independent interface (MII) defined in the IEEE 802.3 standard and the reduced media independent interface (RMII).

27.1.1 EMAC structure

Figure 27-1 Block diagram of EMAC



27.1.2 EMAC main features

The Ethernet peripheral contains an EMAC core and a DMA controller. The transmission and reception of the frame is scheduled by DMA.

DMA features

- Programmable AHB burst transfer types
- Supports ring or chained descriptor
- Each descriptor can transfer up to 8 KB of data
- Poll or fixed-priority arbitration between transmission and reception
- Programmable interrupts for different operational conditions
- Status information report for each transfer

EMAC core features

- Supports 10/100Mbps data transfer rates
- IEEE 802.3-compliant MII interface to communicate with an external high-speed Ethernet PHY
- Supports flow control for full-duplex operation, and CSMA/CD protocol for half-duplex operations
- Automatic CRC and pad generation controllable on transmission frames
- Automatically remove pad bits/CRC on reception frames
- Programmable frame length up to 16 KB
- Programmable frame gap (40-96 bits)
- Supports a variety of address filtering modes and promiscuous modes
- Supports IEEE 802.1Q VLAN tag detection for reception frames
- Supports mandatory network statistics with RMON/MIB counters (RFC2819/RFC2665)
- Detection of LAN remote wakeup frames and AMD Magic Packet™ frames

- Supports checking IPv4 header checksum and IPv4, TCP, UDP or ICMP (packaged in IPv4 or IPv6 data formats) checksum
- Supports Ethernet frame time stamp as defined in IEEE 1588-2008. 64-bit time stamps are recorded in the transmit or receive status
- Two 2 KB FIFOs: one for transmit, and one for receive with a configurable threshold
- Filter received error frames and not forward them to the application in store-forward mode
- Supports store-forward mechanism for data transfer to the MAC controller
- Discard frames on late collision, excessive collisions, excessive deferral and underflow conditions
- Clear FIFO by software
- Calculates and inserts IPv4 header checksum and TCP, UDP or ICMP checksum in frames transmitted in store-forward mode
- Supports loopback mode on the MII interface for debugging
- Programmable time stamps of receive and transmit frames as defined in IEEE 1588-2008 standard
- Supports two correction methods: coarse and fine correction
- Second pulse output (programmable)
- Trigger interrupts when the system is greater the specified time

27.2 EMAC functional description

The Ethernet peripheral consists of a MAC 802.3 (media access controller), MII interface and a dedicated DMA controller.

It implements the following functions:

- Data transmit and receive
  - Framing (frame boundary and frame synchronization)
  - Handling of source and destination addresses
  - Error detection
- Media access management in half-duplex mode
  - Medium allocation (avoid collision)
  - Collision resolve (handle collision)

Usually there are two operating modes for the MAC sublayer:

- Half-duplex mode: The stations compete for the use of the physical medium using the CSMA/CD algorithm. Two stations can only communicate in a single transfer direction at the same time.
- Full-duplex mode: CSMA/CD algorithm is unnecessary, but the following conditions must be met:
  - Physical medium supports simultaneous transmission and reception
  - Only two stations connected to the LAN
  - Both two stations configured as full-duplex mode

27.2.1 EMAC communication interfaces

The EMAC allows to configure the station management interface (SMI) of PHY, media-independent interface (MII) for Ethernet frame communication, and reduced media-independent interface RMII.

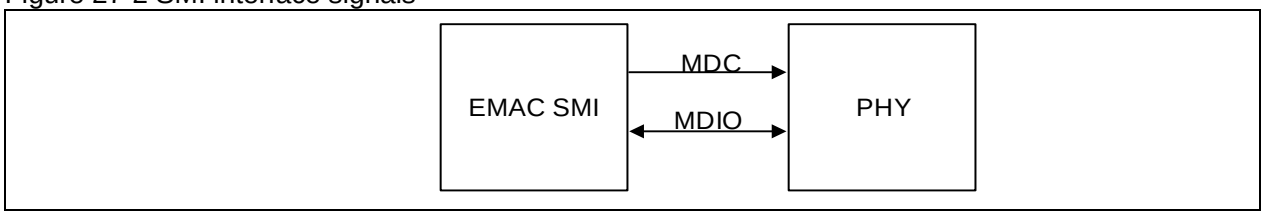
Station management interface (SMI)

The PHY management interface (SMI) accesses PHY registers through a clock and data line. It supports up to 32 PHYs.

MDC: PHY configures clock signals, at the maximum frequency of 2.5 MHz. The minimum high and low times for MDC must be 160ns, and the minimum period is 400ns. In idle state, the MDC clock signal remains low.

MDIO: Bidirectional port, data input/output lines.

Figure 27-2 SMI interface signals



Before write operation, PHY address, MII register and EMAC_MACMIIDT register must be configured first, followed by the MII MW and MB bits, and then the SMI interface will transfer the PHY address, PHY register address and data to the PHY. During the transaction, the contents of the EMAC_MACMIIADDR and the EMAC_MACMIIDT registers are not allowed to change.

Before read operation, the PHY address and MII register must be configured first, and then the MB bit is set and MW bit is set to 0 in the EMAC_MACMIIADDR register.

The SMI interface sends the PHY address and PHY register address, and then starts reading the PHY register contents. During the transaction, the MB bit is always set. It is cleared by the SMI interface at the end of read operation. Attention should be paid to the fact that the contents of the EMAC_MACMIIADDR and EMAC_MACMIIDT registers are not allowed to change (The application should not change these register contents. After the transaction, the EMAC_MACMIIDT register is automatically updated with the data read from the SMI).

The SMI clock source is a divided AHB clock. The divide factor depends on the ABH clock frequency. Note that the divide factor must be configured correctly since the MDC frequency must not be greater than 2.5 MHz.

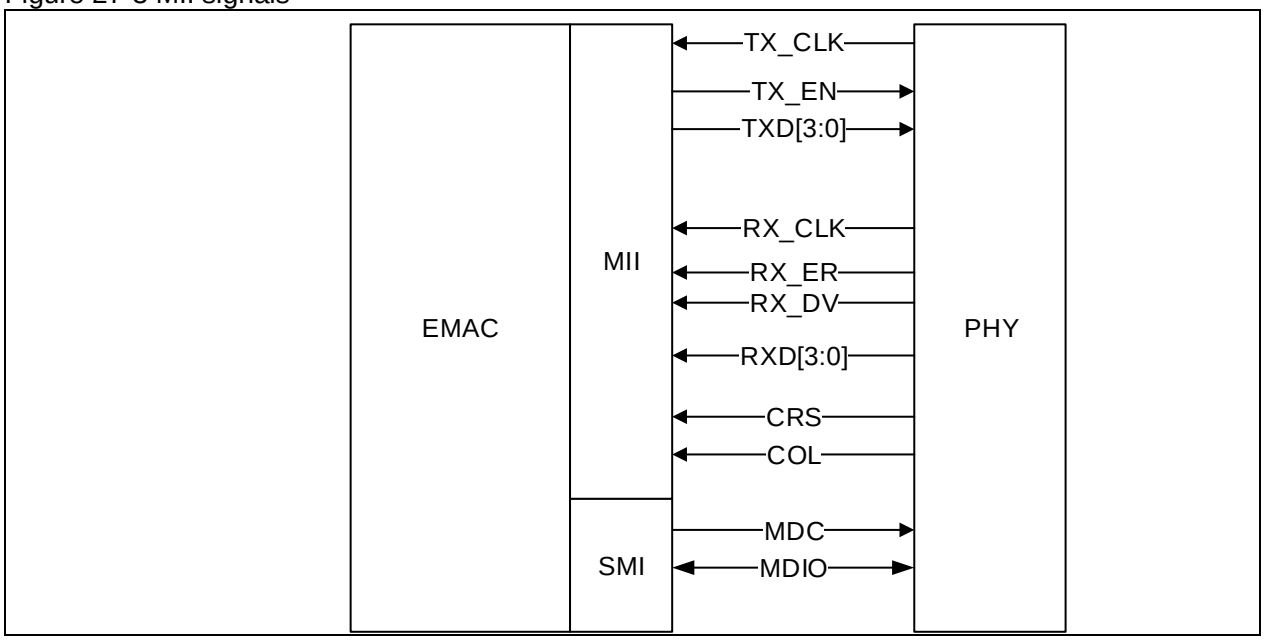
Table 27-1 shows the clock range.

Selection bit	AHB clock	MDC clock
0000	60~100MHz	AHB clock/42
0001	100~150MHz	AHB clock/62
0010	20~35MHz	AHB clock/16
0011	35~60MHz	AHB clock/26
0100	150~192MHz	AHB clock/102
0101,0110,0111	Reserved	--

Media-independent interface: MII

The media-independent interface (MII) acts an interconnection between the MAC sublayer and the PHY for data transfer at 10Mbit/s and 100Mbit/s.

Figure 27-3 MII signals



MII_TX_CLK: Transmit data clock signal. This clock is 2.5MHz at 10Mbps speed; 25MHz at 100Mbps speed.

MII_RX_CLK: Receive data clock signal. This clock is 2.5MHz at 10Mbps speed; 25MHz at 100Mbps speed.

MII_TX_EN: Transmit enable signal. It must be set synchronously with the start bit of the preamble, and must remain asserted until all bits to be transmitted are transmitted.

MII_TXD[3: 0]: Transmit data. 4-bit data are transmitted each time synchronously. Data is valid when the MII_TX_EN signal is active. The MII_TXD[0] is the least significant bit while the MII_TXD[3] is the most significant bit.

MII_CRS: carrier sense signal defined in the CSMA/CD. It is controlled by the PHY and can be valid only in half-duplex mode. This signal is active when either the transmit or receive medium is not idle. The PHY must ensure that the MII_CS signal remains active throughout the duration of a collision condition. This signal is not required to be synchronized with the TX and RX clocks.

MII_COL: Collision detection signal defined in CSMA/CD. It is controlled by the PHY and can be valid only in half-duplex mode. This signal is enabled upon detection of a collision on the medium and will remain enabled during the duration of the collision. This signal is not required to be synchronized with the TX and RX clocks.

MII_RXD[3: 0]: It is controlled by the PHY. Four-bit data to be received are transmitted synchronously. Data is valid when the MII_RX_DV signal is active. The MII_RXD[0] is the least significant bit, while the MII_RXD[3] is the most significant bit. While the MII_RX_DV is inactive but the MII_RX_ER is active, the PHY will transmit a specific MII_RXD[3: 0] value to indicate specific information.

MII_RX_DV: Receive data valid signal. It is controlled by the PHY. This signal is valid when the PHY has kept data on the MII for reception. It must be enabled synchronously with the first bit (MII_RX_CLK) and must remain enabled until data transfer is fully completed. It must be cleared prior to the first clock following the transmission of the final four-bit data. In order to receive a frame correctly, the MII_RX_DV signal must remain valid throughout the frame transmission. The active level must be no later than the SFO bit.

MII_RX_ER: Receive error signal. It must be active for one or more clock periods (MII_RX_CLK) to indicate to the MAC that an error was detected in a frame. Detailed error information depends on the state of the MII_RX_DV and MII_RXD[3: 0] data.

Table 27-2 Transmit interface signal encode

MII_TX_EN	MII_TXD[3: 0]	Description
0	0000 to 1111	Normal frame interval
1	0000 to 1111	Normal data transfer

Table 27-3 Receive interface signal encode

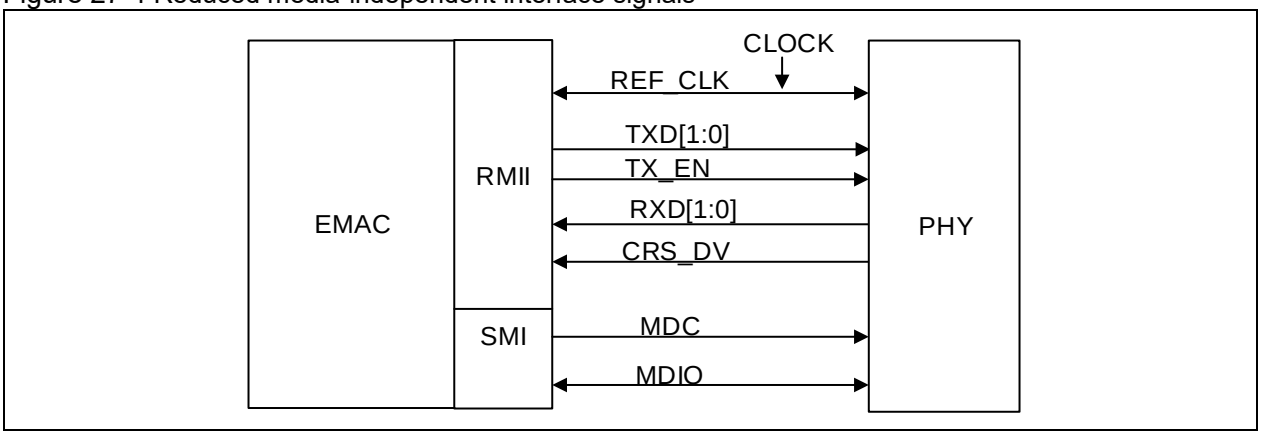
MII_RX_DV	MII_RX_ER	MII_RXD[3: 0]	Description
0	0	0000 to 1111	Normal frame interval
0	1	0000	Normal frame interval
0	1	0001 to 1101	Reserved
0	1	1110	False carrier indication
0	1	1111	Reserved
1	0	0000 to 1111	Normal data reception
1	1	0000 to 1111	Data reception error

Reduced media-independent interface: RMII

The reduced media-independent interface reduces the pin count between the AT32F457xx Ethernet peripheral and the external Ethernet. According to the IEEE802.3u standard, the MII interface requires 16 pins for data and control signals, while the RMII reduces the pin count to 7 pins (a 62.5% decrease in pin count)

The RMII interface is used to connect the EMAC and the PHY. This helps translate the MAC MII signal to the RMII.

Figure 27-4 Reduced media-independent interface signals



MII/RMII selection and clock sources

Either the MII or RMII mode can be selected using the 23rd bit MII_RMII_SEL in the SCFG_CFG2 register. The MII/RMII mode must be selected when the Ethernet controller is in reset state or before the clock is enabled.

MII clock sources

The EMAC TX_CLK and RX_CLK clock signals are provided by the PHY. External PHY module is driven by an external 25MHz clock, which can be provided by either an oscillator or the MCU CLKOUT pin. Refer to CRM section for more information.

Figure 27-5 MII clock sources (provided by CLKOUT pin)

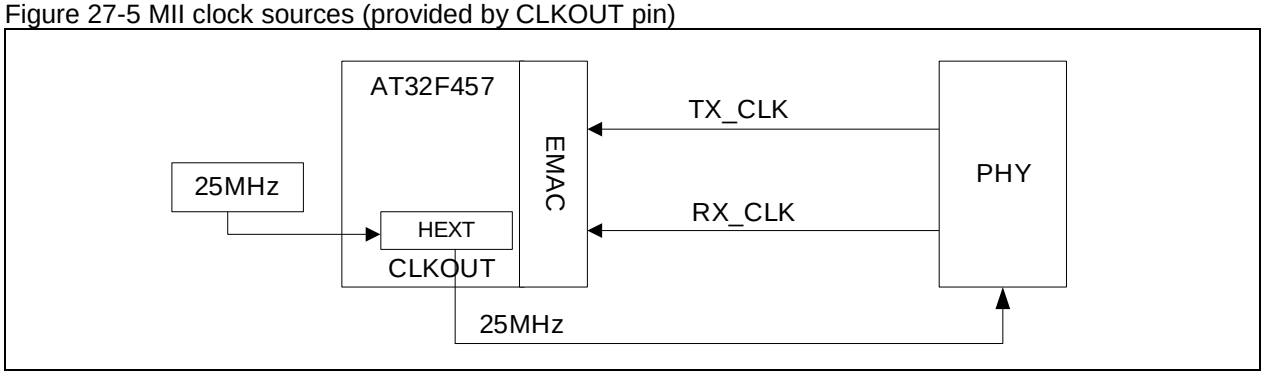
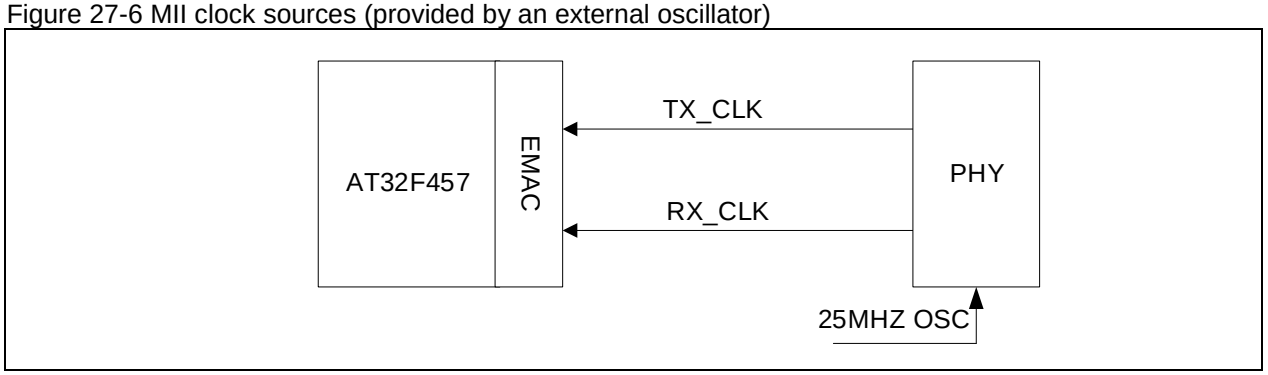


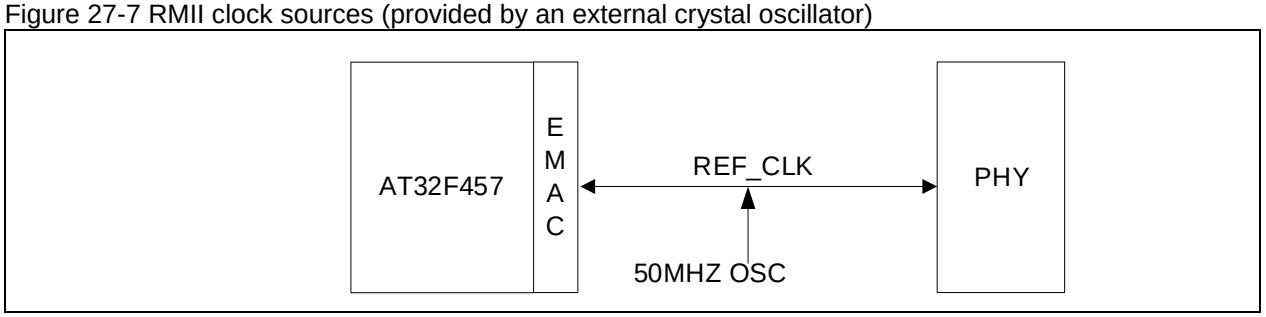
Figure 27-6 MII clock sources (provided by an external oscillator)



RMII clock sources

As shown in Figure 27-7, both the EMAC and PHY require 50MHz clock sources, which can be done with an external crystal oscillator.

Figure 27-7 RMII clock sources (provided by an external crystal oscillator)



EMAC pin allocation and multiplexing

Table 27-4 Ethernet peripheral pin configuration

EMAC signal	Name	Pin	Pin description
SMI	MDC	PC1	Multiplexed function
	MDIO	PA2	Multiplexed function
MII	TX_CLK	PC3	Multiplexed function
	TX_EN	PB11/ PG11	Multiplexed function
	TXD3	PB8/PE2	Multiplexed function
	TXD2	PC2	Multiplexed function
	TXD1	PB13/PG14	Multiplexed function
	TXD0	PB12/PG13	Multiplexed function
	RX_CLK	PA1	Multiplexed function
	RX_ER	PB10	Multiplexed function
	RX_DV	PA7/PD8	Multiplexed function
	RXD3	PB1/PD12	Multiplexed function
	RXD2	PB0/PD11	Multiplexed function
	RXD1	PC5/PD10	Multiplexed function
	RXD0	PC4/PD9	Multiplexed function
	CRS	PA0	Multiplexed function
	COL	PA3	Multiplexed function
RMII	REF_CLK	PA1	Multiplexed function
	TXD1	PB13/PG14	Multiplexed function
	TXD0	PB12/PG13	Multiplexed function
	TX_EN	PB11/PG11	Multiplexed function
	RXD1	PC5/PD10	Multiplexed function
	RXD0	PC4/PD9	Multiplexed function
	CRS_DV	PA7/PD8	Multiplexed function
PPS	PPS_OUT	PB5/PG8	Multiplexed function

27.2.2 EMAC frame communication

Frame format

Figure 27-8 shows the MAC frame format and tagged MAC frame format (Refer to IEEE 802.C-2002 for more information on MAC frame formats)

Figure 27-8 MAC frame format

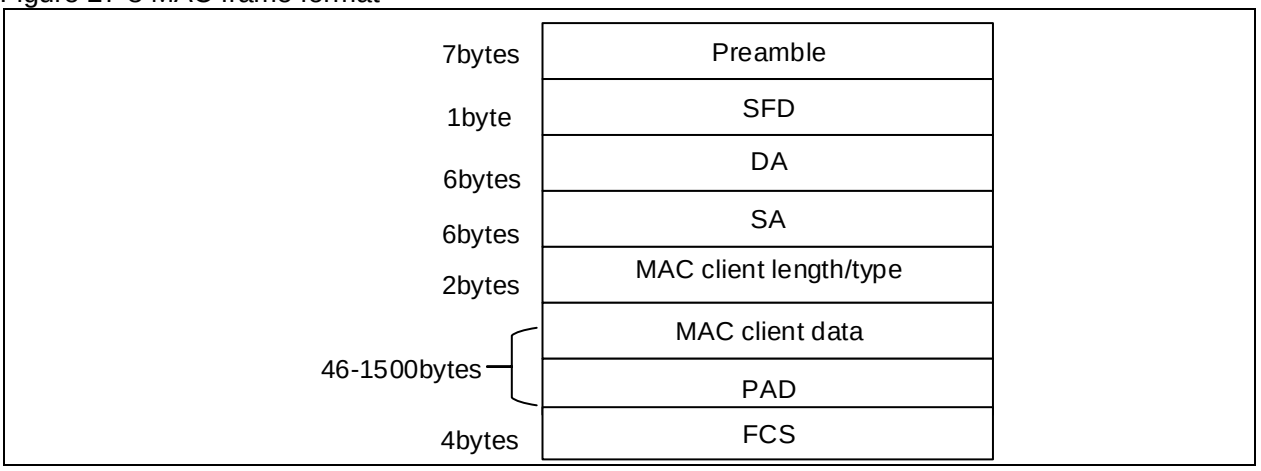
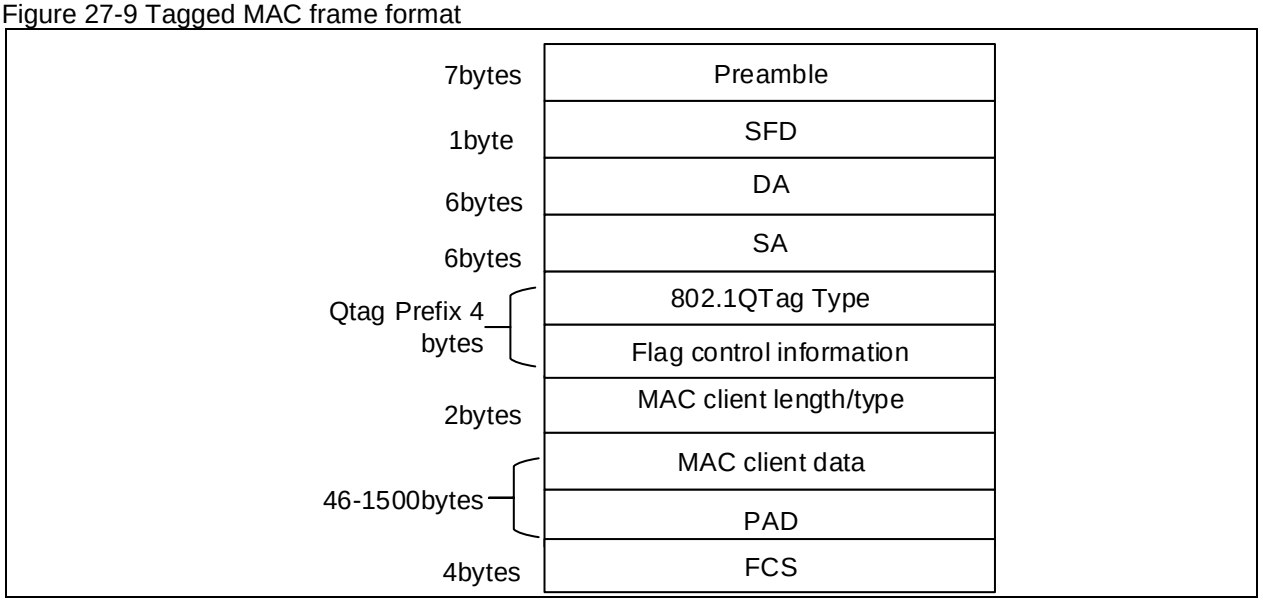


Figure 27-9 Tagged MAC frame format



EMAC frame filtering

EMAC supports source and destination address filtering.

Address filtering

Based on frame filtering register chosen by the application, address filtering checks the source and destination addresses over all received frames using the MAC address and multicast HASH table. The address filtering status is reported accordingly.

Unicast destination address filter

There are two filtering modes: perfect address filtering and HASH table filtering.

1. Perfect address filtering: It is enabled by setting HUC=0 in the frame filtering register. Four MAC addresses are used for perfect filtering. The MACADDR0 is always enabled. The MACADDR1, MACADDR2 and MACADDR3 bits are enabled using their respective AE bit. The MBC bit in the address register is set to mask the address comparison of the corresponding bytes. If the MBC bit is all 0, the EMAC will compare all 48 bits of the received unicast address with the programmed MAC address, if matched, the unicast address is said to have passed through the filtering.
2. HASH table filtering: The HUC must be set in the frame filtering register. The EMAC performs imperfect filtering for unicast addresses through 64-bit HASH table. For HASH filtering, the MAC calculates the CRC value of the received destination address (see the note below) and uses the 6 upper CRC bits to index the HASH table. A value of 000000 corresponds to bit 0 in the HASH table register, and a value of 111111 corresponds to bit 63 in the HASH table register. If the corresponding bit in the HASH table relative to the CRC value is set to 1, it indicates that the frame has passed through the HASH filter; otherwise, it has failed the HASH filter.

Multicast destination address filter

1. The MAC can be programmed to receive all multicast frames by setting PMC=1 in the frame filter register
2. If the PMC is set to 0 and the HMC is set to 0, perfect address filtering can be done using the MACADDR0/1/2/3 addresses.
3. If the PMC is set to 0, and the HMC is set to 1, the 64-bit HASH table is used to perform imperfect filtering.

Broadcast address filter

If the DBF is set in the frame filter register, the EMAC will reject all broadcast frames. If DBF=0, the EMAC will accept all broadcast frames.

Unicast source address filter

If the bit 30 is set in the MAC address register 1/2/3, the filter will compare source address instead of destination address of the received frames.

If the SAF bit is set in the frame filter address, the frames that failed the SA filter will be dropped by the

EMAC. In this case, the frames that pass through both SA and DA filtering can be forwarded to the application; otherwise, they will be dropped.

Inverse filtering operation

The DAIF and SAIF bits in the frame filter register are used to invert the filtering output result for both destination and source address filtering. The DAIF bit is applicable for both unicast and multicast destination frames, while the SAIF bit for the unicast source address.

Table 27-5 Destination address filtering

Frame type	PMC	HPF	HUC	DAIF	HMC	PMC	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all frames
	0	X	0	0	X	X	X	Pass on perfect/group filter match
	0	X	0	1	X	X	X	Fail on perfect/group filter match
	0	0	1	0	X	X	X	Pass on HASH filter match
	0	0	1	1	X	X	X	Fail on HASH filter match
	0	1	1	0	X	X	X	Pass on HASH or perfect/group filter match
	0	1	1	1	X	X	X	Fail on HASH or perfect/group filter match
Multicast	1	X	X	X	X	X	X	Pass all frames
	X	X	X	X	X	1	X	Pass all frames
	0	X	X	0	0	0	X	Pass on perfect/group filter match and drop PAUSE frames if PCF= 0x
	0	0	X	0	1	0	X	Pass on HASH filter match and drop PAUSE frames if PCF= 0x
	0	1	X	0	1	0	X	Pass on HASH or perfect/group filter match and drop PAUSE frames if PCF= 0x
	0	X	X	1	0	0	X	Fail on perfect/group filter match and drop PAUSE frames if PCF= 0x
	0	0	X	1	1	0	X	Fail on HASH filter match and drop PAUSE frames if PCF= 0x
	0	1	X	1	1	0	X	Fail on HASH or perfect/group filter match and drop PAUSE frames if PCF= 0x

Table 27-6 Source address filtering

Frame type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all frames
	0	0	0	Pass status on perfect/group filter match but do not drop frames that failed
	0	1	0	Fail status on perfect/group filter match but do not drop frames that failed
	0	0	1	Pass on perfect/group filter match and drop frames that failed
	0	1	1	Fail on perfect/group filter match and drop frames that failed

EMAC frame transmission

The EMAC frame transmission is controlled by the DMA controller and MAC. Once a transmission command is sent by the application, Ethernet frames read from the user data buffer (such as SRAM) are

stored into TXFIFO by the DMA. The frames are then popped out and transferred to the MAC core. The MAC core sends the frames to external Ethernet PHY via MII/RMII interface so as to communicate with external stations. When the end-of-frame is transferred, the status of the transmission is generated from the MAC core and transferred back to the DMA controller.

When the SOF is detected, the MAC accepts data and begins transmitting to the MII/RMII. The time required to transmit the data frame to the MII/RMII after the application initiates is variable, due to various delay factors like frame interval, time to transmit preamble/SFD and any back-off delays caused by CSMA/CD algorithm in half-duplex mode. After the EOF is transferred to the MAC core, the core completes normal transmission and then gives the status of the transmission back to the DMA.

There are two modes of operation for popping data from TXFIFO to the MAC core:

- In threshold mode: If the number of bytes in the FIFO crosses the configured threshold (or when the EOF is written before the threshold is crossed), the data is ready to be popped out and forwarded to the MAC core. The threshold level is configured using the TTC bits in the EMAC_DMAOPM register.
- In store-and-forward mode: Only after a complete frame is written to the FIFO, the data is transferred to the MAC core. If the TXFIFO size is smaller than the Ethernet frame to be transmitted, then the data is also transferred to the MAC core when the TXFIFO becomes almost full.

  The FTF bit (the bit 20 in the EMAC_DMAOPM register) can be set to flush the TXFIFO. If the FTF bit is mistakenly set in the middle of transferring a frame to the MAC core, the FIFO is flushed and the frame transmission is aborted. An underflow event is marked in the EMAC transmitter and the corresponding status is given to the DMA core.

Automatic CRC and pad generation

If the length of a transmitting fame is less than 46 bytes, the minimum data size for an Ethernet frame defined in IEEE802.3 standard, the EMAC can be programmed to append padding (zeroes are appended) automatically so that zeroes are appended to the transmitting frame to make the data length exactly 46 bytes.

During a frame transmission, the EMAC can be programmed either to append CRC to the frame check sequence or not to append CRC. When the EMAC is programmed to append pads for frames (DA+SA+LT+Data) less than 60 bytes, the CRC value will be appended at the end of the padded frames.

Collision detection in CSMA/CD

The collision detection is applicable only to half-duplex mode, not to full-duplex mode (refer to Ethernet protocol for more information).

In MII mode, if a collision occurs at any time from the beginning of a frame to the end of the CRC, the collision signal is sent to the EMAC core. The EMAC core will send a 32-bit jam signal of 0x5555 5555 to inform all other stations on the LAN that a collision has occurred. If the collision happened during the preamble transmission phrase, the EMAC completes the transmission of the preamble and SFD and then sends the jam signal.

Jabber timer

The EMAC jabber timer is enabled (set EMAC_MACCTRL[20]=0) to prevent certain station on the LAN from occupying the LAN for a long time. Once enabled, the EMAC will stop transmitting if the application tries to send a frame more than 2048 bytes.

Interframe gap management

The EMAC enables transmission after satisfying the interframe gap and backoff delays. The MAC maintains an idle period of the programmed interframe gap (IFG bits in the EMAC_MACCTRL register) between any two transmitted frames. The EMAC starts its IFG counter as soon as the carrier signal of the MII becomes inactive. If a frame arrives sooner that the configured IFG time, it will be delayed for transmission until the interframe time is reached. The MAC starts its IFG counter as soon as the carrier signal of the MII becomes inactive. In full-duplex mode, the MAC enables transmission at the end of the configured IFG value. In half-duplex mode, if the IFG is configured as 96 bit times, the MAC will follow the rule defined in Section 4.2.3.2.1 of the IEEE 802.3 specification. The MAC will reset its IFG counter if a carrier is detected during the first two-thirds of the IFG interval (64-bit times). If the carrier is detected during the final one-third of the IFG interval, the MAC will continue the IFG count and enables

transmission after the IFG interval is reached. In half-duplex mode the MAC follows the truncated binary exponential backoff algorithm.

Transmit flow control

In full-duplex mode the EMAC implements Transmit flow control using pause frames. The EMA can request the suspension of the frame transmission by sending Pause frames in two ways.

When the FCB bit in the EMAC_MACFCTRL register is set, the EMAC generates and transmits a single Pause frame. The value of the pause time in the generated frame holds the programmed pause time value in the EMAC_MACFCTRL register. To extend the pause time or cancel the remaining pause time, the EMAC must send another pause frame (PT=0 will cancel the remaining pause time).

If flow control is enabled (EFT=1 in the EMAC_MACFCTRL register), when the RXFIFO is full, the EMAC generates and transmits a Pause time. If the RXFIFO remains full while the programmed pause time threshold is satisfied, the MAC will transmit another Pause frame. The process is repeated as long as the receive FIFO remains full. If the RXFIFO is not full prior to the pause time, the MAC transmit a Pause time with zero pause time to indicate to the remote station that the receive buffer is ready to receive new data frames.

Retransmission during collision

When a frame is being transferred to the MAC, a collision event may occur on the MAC line in half-duplex mode. The MAC would then try retransmission even before the end of the frame is received. At this point, if retransmission is enabled, the frame is popped out again from the FIFO. After 96 bytes have been popped towards the MAC core, the FIFO controller will release that space to allow the DMA to push in more data. This means that the retransmission is not possible if this threshold (96 bytes) is crossed or when the MAC core indicates a late collision event.

Transmit FIFO flush operation

The FTF bit (bit 20) in the EMAC_DMAOPM register is set to flush TXFIFO. The TXFIFO flush operation is immediate and the corresponding points are reset to their initial states even if the TXFIFO is in the process of transferring a frame to the MAC core. This operation will abort the current frame transmission and result in an underflow event in the EMAC. The status of such a frame is generated (TDES0 bits 1 and 13). The flush operation is completed when the application (DMA) has received all of the status words of the frames that were flushed. Then the FTF bit is cleared in the EMAC_DMAOPM register. In this case, TXFIFO is allowed to receive new frames from the application (DMA). All data that do not start with an SOF marker will be discarded after the flush operation.

Transmit status word and time stamp

At the EMAC controller has completed the transmission of the Ethernet frame, the transmit status is given to the application. If IEEE 1588 time stamp is enabled, a 64-bit time stamp, along with the transmit status, will be written to the transmit descriptor.

Transmit checksum offload

The most widespread use of Ethernet is to encapsulate TCP or UDP over IP datagrams, so the Ethernet controller has a transmit checksum feature that supports checksum calculation and insertion in the transmit path, and error detection in the receive path.

Note: This function is enabled only when the TXFIFO is configured as store-and-forward mode (that is, when the TSF is set in the EMAC_DMAOPM register). If the TXFIFO depth is less than the input Ethernet frame size, the checksum function is invalid and only the IPv4 header checksum is calculated and modified by the EMAC, even in store-and-forward mode.

See IETF specifications RFC791, RFC 793, RFC 768, RFC 792, RFC 2460 and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6 and ICMPv6 specifications.

IP header checksum

The checksum module will detect an IPv4 datagrams when the Ethernet frame's type field has the value of 0x0800 and the IP datagram's version field has the value of 0x4. The input frame's checksum field is ignored and replaced by the calculated value. IPv6 headers do not have a checksum field, thus the checksum module does not modify IPv6 header fields. The result of this IP header checksum calculation is indicated by the IP header error status bit (TDES0 bit 16). This status bit is set whenever the values of the Ethernet type field and the IP header's version filed are not consistent, or when the Ethernet frame does not have enough data. In other words, this bit is set when the following errors occurred.

- For IPv4 datagrams
  - The received Ethernet type is 0x0800, but the IP header's version field is not equal to 0x4
  - The IPv4 header length field has a value less than 0x5 (20 bytes)
  - The total frame length is less than the value given in the IPv4 header length field
- For IPv6 datagrams
  - The received Ethernet type is 0x86DD, but the IP header's version field is not equal to 0x6
  - The frame ends before the IPv6 header (40 bytes) or extension header (including header length field) has been completely received. Even if the checksum module detects such an IP header error, it inserts an IPv4 header checksum if the Ethernet type field indicates an IPv4 payload.

TCP/UDP/ICMP checksum

The TCP/UDP/ICMP checksum determines whether the encapsulated data is TCP, UDP or ICMP by analyzing the IPv4 or IPv6 header (including extension headers).

Note: 1. For non-TCP, - UDP or - ICMP/ICMPv6 data, this checksum function is invalid and the frame data is not modified.

2. Fragmented IP frames (IPv4 or IPv6), IP frames with security features (such as an authentication header or encapsulated security data) and IPv6 frames with routing headers are not processed by the checksum.

The checksum is calculated for the TCP, UDP or ICMP data and inserted into its corresponding field in the header. It can work in the following two modes:

1. The TCP, UDP or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the input frame's checksum field. The checksum field is included in the checksum calculation, and then replaced by the final calculated checksum.
2. The checksum field is ignored, and the TCP, UDP or ICMPv6 pseudo-header data are included into the checksum calculation, and the checksum field is overwritten with the final calculated value.

The result of this operation is indicated by the checksum error status bit in the transmit status vector (TDES0 bit 12). The data checksum error status bit is set when either of the following is detected:

1. The frame has been forwarded to the MAC transmitter in store-and-forward mode without the end of the frame being written to the TXFIFO.

2. The packet ends before the number of bytes indicated by the data length field in the IP header is received.

When the packet is longer than the indicated data length, the bytes are ignored as stuff bytes, and no error is reported. When the first type of error is detected, the TCP, UDP or ICMP header is not modified. For the second type of error, the calculated checksum is still inserted into the corresponding header field.

EMAC frame reception

The MAC received frames are stored into the RXFIFO and sent out by the DMA using the AHB interface. There are two modes: Cut-through mode and store-and-forward mode.

In the default Cut-through mode, when the frame data length greater than the programmed threshold (configured with the RTC bit in the EMAC_DMAOPM register) or a full packet of data are received in the RXFIFO, the data are popped out and the DMA is notified of its availability. The DMA will keep popping out the data from the RXFIFO until the data transfer is completed. Upon completion of the EOF frame transfer, the status word is popped out and sent to the DMA controller.

In store-and-forward mode (configured by the RSF bit in the EMAC_DMAOPM register), a frame is read by the DMA only after being written completely into the RXFIFO.

If the EMAC core is configured to drop all error frames, behavior varies in different reception modes:

1. In store-and-forward mode, only the valid frames are read and forwarded to the application by the DMA.
2. In Cut-through mode, some error frames are not dropped because the error status is received at the end of the frame.

A reception operation is initiated when the EMAC detects an SFD on the MII. The MAC core strips the

preamble and SFD before processing the frame. The header fields are checked for the filtering and the FCS field used to verify the CRC of the frame. The frame is dropped in the core if it failed the address filter.

If IEEE1588 time stamping is enabled, a snapshot of the system time is taken when any frame's SFD is detected on the MII. This time stamp is passed onto the application when the EMAC has completed the current frame.

If the received frame length/type field is less than 0x600 and if the EMAC is configured for the auto CRC/pad stripping option, the EMAC sends the data of the frame to RXFIFO up to the count specified in the length/type field, then starts dropping bytes (including the FCS filed). If the length/type field is greater than 0x600, the MAC sends all received Ethernet frame data to RXFIFO, regardless of the value on the programmed auto-CRC strip option.

The EMAC watchdog timer is enabled by default, frames above 2048 bytes (DA + SA + LT + Data + pad + FCS) are cut off. This feature can be disabled by the WD bit in the EMAC_MACCTRL register. However, even if the watchdog timer is disabled, frame length greater than 16 KB are cut off and a watchdog timeout event is reported.

Receive checksum offload

Both IPv4 and IPv6 frames are detected for data integrity by setting the IPC bit in the EMAC_MACCTRL register. The EMAC identifies IPv4 or IPv6 frames by checking for the Ethernet type field value. The receive checksum offload calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of the header checksum is indicated by the bit 7 of the receive descriptor (RDES0). The IP header error bit is set either one of the following conditions:

1. Ethernet type field does not match the IP header version field
2. Received frames are less than the length indicated by the IPv4 header length field
3. IPv4 or IPv6 headers less than 20 bytes

The receive checksum offload also identifies a TCP, UDP or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP or ICMP specifications. It includes the TCP/UDP/ICMPv6 pseudo-header bytes for checksum calculation and the result of CRC is indicated by the bit 0 in the receive descriptor (RDES0). This bit is set either one of the following conditions:

1. Received TCP, UDP or ICMP data length does not match that of the IP headers
2. The calculated checksum does not equal the value of the TCP, UDP or ICMP checksum field

Receive flow control

In Full-duplex mode, the MAC detects the receiving Pause frame and pauses the frame transmission according to the delay specified within the received Pause frame.

The ERF bit in the EMAC_MACFCTRL register to enable or disable Pause frame detection function. Once receive flow control is enabled, the EMAC will decode the Pause frames.

During the pause period, if another Pause frame is detected with a zero Pause time value, the MAC clears the PAUSE time and retransmits the data. If it is not a zero Pause time value, the pause time in the frame will be immediately loaded into the pause time counter.

Receive operation multiframe handling

Since the status is available immediately following the data, the RXFIFO is capable of storing any number of frames into it, as long as it is not full.

Receive status word

At the end of the Ethernet frame reception, the EMAC will send the receive status to the application (DMA). The detailed information of the receive status is given in the RDES0.

EMAC loopback mode

The EMAC supports loopback of transmitted frames onto its receiver. The loopback mode is very useful to debug the Ethernet communication. This feature is enabled by setting the LM bit in the EMAC_MACCTRL register.

Note that the loopback mode is applicable only to the MII interface.

EMAC frame counter

The MAC management counters (MMC) contain a set of registers for gathering statistics on the received

and transmitted frames. These include the EMAC_MMCCTR, EMAC_MMCRI, EMAC_MMCRIM, EMAC_MMCTI and EMAC_MMCTIM registers.

If a frame transmission is aborted due to any of the following errors, this frame will not be counted:

1. No carrier/Loss of carrier
2. Jabber timeout
3. Late collision
4. Excessive collision
5. Excessive deferral
6. Frame overflow

If a frame reception is aborted due to any of the following errors, this frame will not be counted:

1. CRC error
2. Runt frame (shorter than 64 bytes)
3. Alignment error
4. Length error (length field value does not match the received frame length)
5. Out of range (frame length beyond the maximum size, untagged frame maximum size=1518 bytes, tagged frame maximum size=1522 bytes)
6. MII_RXER input error

27.2.3 Ethernet frame transmission and reception using DMA

The transmission and reception of the Ethernet frames are scheduled through DMA.

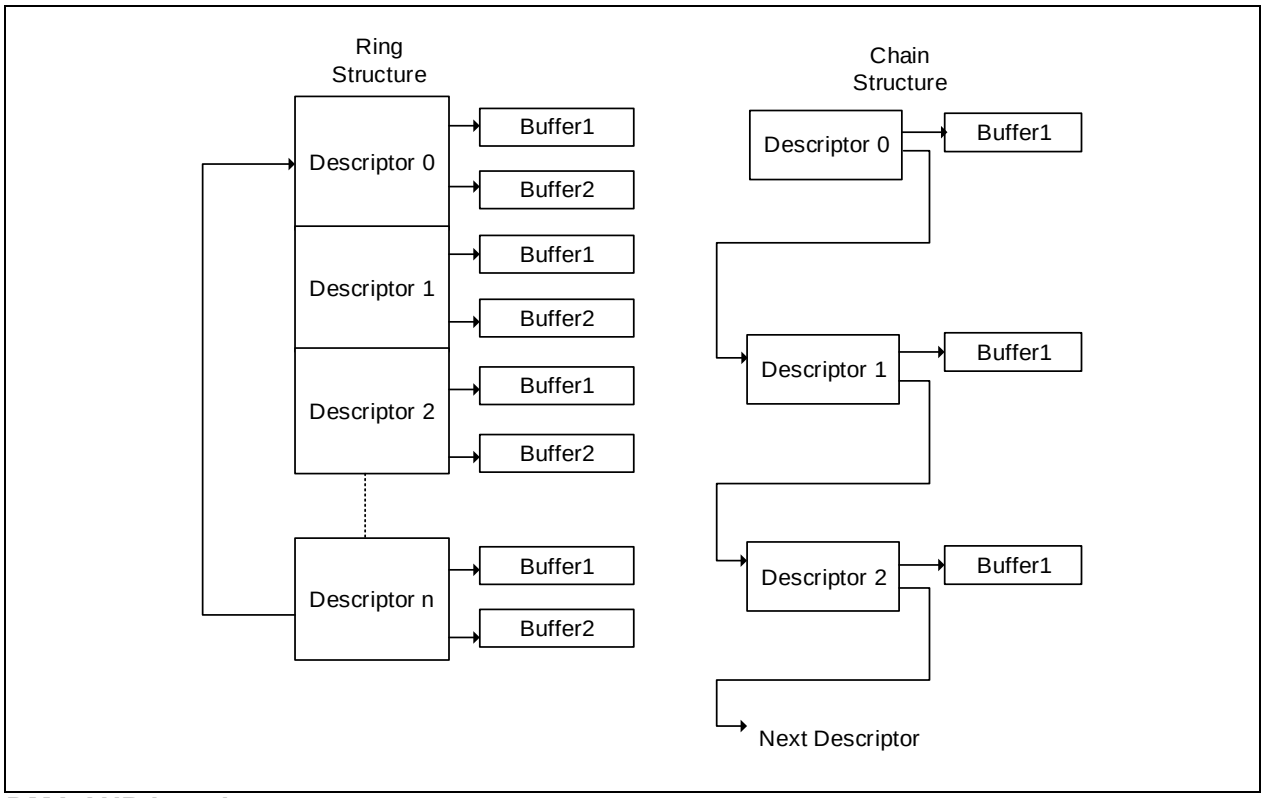
For transmission, the DMA reads out the Ethernet frames from the user system buffer (such as SRAM) via the AHB master interface and forwards them to the TXFIFO. The EMAC core then transfers the data frames in the TXFIFO to the MII/RMII interface.

For reception, the EMAC core sends the received Ethernet frames from the MII/RMII interface to the RXFIFO so that the DMA controller reads out the Ethernet frames from the RXFIFO and transfers them to the user data buffer (SRAM) via the AHB master interface.

DMA control and status register and descriptor table are used to manage the whole transmission and reception process, which has its respective descriptor list. The descriptor list is usually stored in the system buffer area (SRAM). When the transmission and reception is enabled, the DMA polls the descriptor table through the transmit and receive poll register to start the transmission and reception process. The base address the descriptor list for transmission and reception is stored into the transmit descriptor list register and receive descriptor list register.

There are two descriptor structures: ring structure and chain structure. In a ring structure, each descriptor may point to two buffers. In a chain structure (TDES0[20]=1 is configured for transmission, but RDES1[14]=1 for reception), each descriptor points to the only one buffer. The contents of the TDES3 and RDES3 for the current frames refer to the next descriptor address for transmission and reception.

Figure 27-10 Descriptor for ring and chain structure



DMA AHB host burst access

The DMA executes a fixed-length burst access on the AHB master interface if the FB bit is set in the EMAC_DMABM register. The maximum burst length is defined by the PBL filed (bit [13: 8] in the EMAC_DMABM register). The receive and transmit descriptors are always accessed in the maximum possible burst size (limited by PBL) for the 16 bytes to read.

The DMA provides the start address and the number of transfers to the AHB master interface before starting one transfer.

Note that one of the following conditions must be respected for transmission:

1. TXFIFO space is greater than the programmed burst size
2. The number of bytes before the end of a frame is less than the burst size and the TXFIFO can accommodate these bytes

One of the following conditions must be respected for reception:

1. The data available in the RXFIO is greater than the programmed burst size
2. The number of bytes before the end of a frame is less than the burst size and the end of the frame is detected in the RXFIFO

AHB host data alignment

The DMA always initiates transfers with address aligned to the bus width. But the start address of the buffers can aligned to any of the four bytes.

- Example of buffer read: If the transmit buffer address is 0x2000 0AA3, and 15 bytes are to be transferred, then the DMA will read five words (32 bits) from the address 0x2000 0AA0, but when transferring data to the TXFIFO, the first three bytes and the last two bytes will be ignored. The DMA always ensures that is transfer a 32-bit data to the TXFIFO, unless it is the end of the frame.
- Example of buffer write: If the receive buffer address is 0x2000 0BB2 and 16 bytes are to be transferred, the DMA will read five 32-bit data from the address 0x2000 0BB0. But the first two bytes and the last two bytes are dummy data.

Buffer size calculation

For transmission, software needs to calculate the buffer size. The TXDMA transfers the exact number of bytes programmed by buffer size field in the TDES1 to the EMAC core. If the FS bit is set in the TDES0, the DMA marks the first transfer from the buffer as the start of frame. If the LS bit is set in the TDES0,

the DMA marks the last transfer from the buffer as the end of frame.

During a frame reception, if the receive buffer address is word-aligned, the valid length of the buffer refers to the value programmed in the RDES1 if the receive buffer address is not word-aligned, the valid length of the buffer is less than the value configured in the RDES1. The valid length value of the buffer is the value indicated by the RDES1 minus the lower two bit value of the buffer address. For example, if the total buffer size is 1024 bytes and the buffer address is 0x2000 0001, the lower 2-bit value of the address is 0x01, then the valid buffer size is 1023 bytes.

The FS bit is set by the DMA controller when an SOF is received. The LS is set when an EOF is received. If the receive buffer length field is big enough to accommodate a full frame, then both the FS and LS bits will be set in the same descriptor. The actual length of the received frame is indicated by the FL bit in the RDES0.

DMA arbiter

Two types of arbitrations are used for the arbitration between transmit and receive controller: round-bin, and fixed-priority. When round-robin is selected (DA bit is set to 0 in the EMAC_DMABM register), the arbiter allocates the databus according to the ratio set by the PR bit in the EMAC_DMABM register, when both transmit and RXDMA request access to the AHB bus simultaneously. When the DA bit is set to 1, the RXDAM always has priority over the TXDMA for data access.

Error response to DMA

If an error response is received during DMA transfer, then the DMA stops all operations and updates the error bit and the fatal bus error bit in the EMAC_DMASTS register. The DMA can resume operation after software or hardware resets the Ethernet peripherals and re-initiates the DMA.

DMA initialization

1. Configure AT32F457xx bus access parameters in the EMAC_DMABM register.
2. Mask unnecessary interrupt sources in the EMAC_DMAIE register.
3. The application generates the transmit and receive descriptor lists. Then it writes the start addresses of the descriptor lists to both the EMAC_DMARDLADDR and EMAC_DMATDLADDR registers.
4. Configure address filtering registers.
5. Configure the EMAC_MACCTRL register to enable the transmit and receive operating modes Program the PS and DM bits according to the auto-negotiation result
6. Set the bit 13 and bit 1 in the EMAC_DMAOPM register to enable transmission and reception.
7. The transmit and receive controllers begin reading descriptors from the corresponding descriptor lists to process receive and transmit operations.

TXDMA operation: non-OSF mode

The TXDMA proceeds as follows, in default mode:

1. The application sets up the Ethernet frame data buffer and the transmit descriptor (TDES0-TDES3), and sets the OWN bit (TDES0[31])
2. Once the SSTC is set (EMAC_DMAOPM bit [13]), the DMA enables transmission.
3. The DMA polls the transmit descriptor to get a frame to be transmitted. If the DMA detects a descriptor that is being owned by the CPU or if an error occurs, transmission is suspended and suspend state is entered, and both the transmit buffer unavailable (EMAC_DMASTS bit 2) and normal interrupt summary bit (EMAC_DMASTS bit 16) are set. The transmit controller jumps to Step 8.
4. DMA fetches the data from the AT32F457xx memory based on the transmit descriptor indication and transfers the data to the TXFIFO.
5. If the Ethernet frame is stored in multiple descriptors with different descriptor, the DMA will close the intermediate descriptor and fetch the next descriptor. Steps 3, 4 and 5 are repeated until the end of frame data is transferred.
6. When the frame transmission is complete, if IEEE1588 time stamping was enabled for the frame (as indicated in the transmit status), the time stamp value is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer while the transmit status information

is sent to the TDES0. Then the OWN bit is cleared, and the current descriptor is disabled. If time stamping was not enabled for the frame, the DMA only updates the status information to the TDES0 without recording the time stamping

7. When the frame transmission is complete, if the Interrupt on Completion (TDES0[30]) is set, then the transmit interrupt bit (EMAC_DMASTS bit [0]) will be set. The DMA then returns to Step 3 and is ready for the next transmission.
8. In the suspend state, the DAM tries to re-fetch the descriptor (back to Step 3) when it receives a transmit poll request and the overflow interrupt flag bit is cleared.

TXDMA operation: OSF mode

In this mode, the OSF bit is set (EMAC_DMAOPM bit 2=1). When the current data frame transmission is complete, the DMA immediately polls the transmit descriptor for the second frame without the need of waiting for the status information is written.

1. The DMA operates according to steps 1-6 of the TXDMA.
2. The DMA fetches the next descriptor without waiting for the status information update of the last description for the previous frame.
3. If the DMA owns the descriptor, then it will decode the transmit buffer address. If the DMA does not own the descriptor, then it will enter into suspend state and jump to Step 7.
4. The DMA fetches the transmit frame data from the AT32F457xx memory and transfer the frame until the end of frame is transferred. If the frame is split among multiple buffers, the DMA will close the intermediate descriptors.
5. The DMA waits for the transmit status and time stamp of the previous frame. After the status information is available, the DMA writes the time stamp to TDES2 and TDES3 if such time stamp is captured. The DMA then clears the OWN bit and closes the descriptor. If the time stamping was not enabled for the frame, the DMA will not alter the contents of TDES2 and TDES3.
6. If enabled, the transmit interrupt bit is set. The DMA fetches the next descriptor when the status information is normal, and jumps to Step 3. If the previous transmit status shows an underflow error, the DMA enters into suspend state and jumps to Step 7.
7. In suspend state, when the DMA receives a pending status information and time stamp, if the time stamping is enabled it will write the time stamp to TDES2 and TDES3, and writes the status to TDES0. It then sets relevant interrupt flag bits and returns to suspend state.
8. The DMA can exit suspend state and enter run state only after receiving a transmit poll request (EMAC_DMACTD register).

Transmit frame processing

Ethernet frames stored in the transmit buffer must contain destination address, source address, correct type/length field and valid data. As for whether to include CRC value, it depends on the transmit descriptor. If the transmit descriptor requires the EMAC core to disable CRC or pad insertion, the buffer must contain the CRC.

A frame can be stored in multiple buffers that are linked in chain structure. When the transmission starts, the TDES0 bit 28 must be set in the first descriptor, and then the data are transferred from the memory to the TXFIFO. If the TDES0 bit 29 is set, it indicates the last buffer of the frame. After the last buffer of the data has been completed, the DMA writes back the final status information to the TDES0. If the transmit complete interrupt bit (TDES0[30]) is set, the transmit interrupt bit (EMAC_DMASTS bit 0) is set, the next descriptor is fetched, and the above steps are repeated. Actual frame transmission depends on whether the store-and-forward mode or threshold mode is selected. The descriptor is disabled (TDES0[31] is cleared) when the DMA finishes frame transmission.

Transmit polling suspend

Transmit polling can be suspended by one of the following conditions:

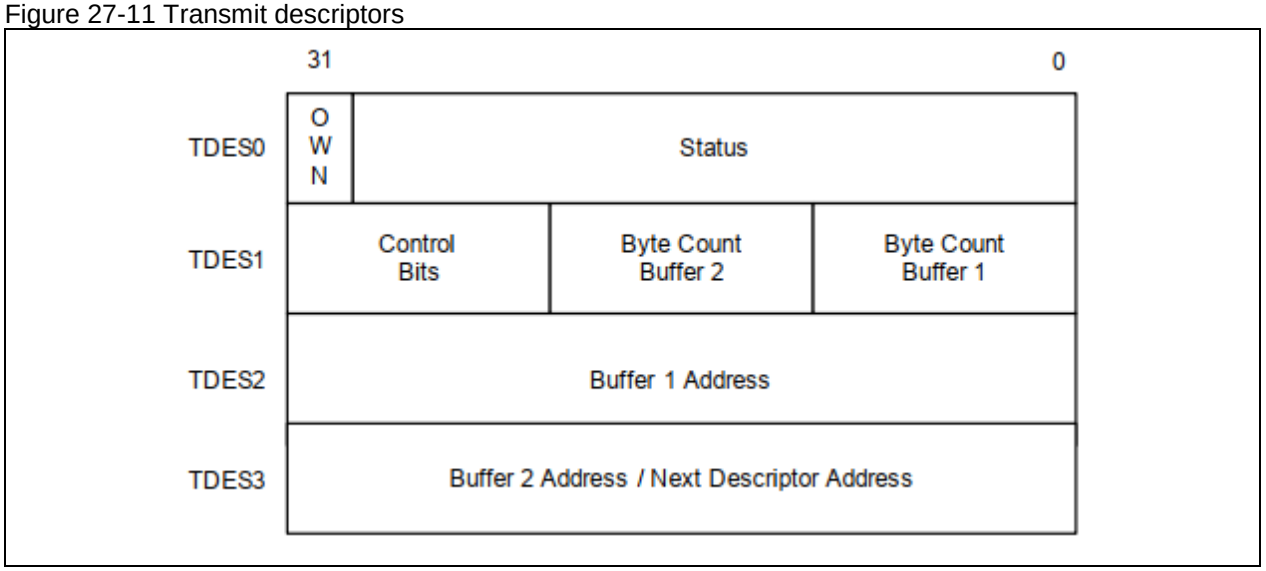
The DMA detects a descriptor owned by the CPU (TDES0[31]=0), and the DMA enters suspend state.

A frame transmission is aborted when an underflow is detected. The abnormal interrupt summary bit (EMAC_DMASTS bit 15) and transmit data underflow bit (EMAC_DMASTS bit 5) are set, and the appropriate error bit is set in the TDES0.

TXDMA descriptors

The descriptor structure consists of four 32-bit words. The bit definitions of TDES0, TDES1, TDES2 and TDES3 are as shown below:

Figure 27-11 Transmit descriptors



TDES0: Transmit descriptor word0

The software must configure the control bits [30: 26]+ [23: 20] and the OWN bit during descriptor initialization. When the DMA updates or writes the descriptor, it clears all the control bits and OWN bit, and report only the status bits.

Bit	Name	Type	Description
Bit 31	OWN	rw	Own bit 0: The descriptor is owned by the CPU 1: The descriptor is owned by the DMA This bit is cleared by the DMA when the DMA completes the frame transmission or when all the data in the buffer are read completely. The own bit of the frame's first descriptor can be set only after subsequent descriptors for the same frame have been set.
Bit 30	IC	rw	Interrupt on completion When set, this bit sets the transmit interrupt bit (EMAC_DMASTS bit [0]) after the present frame has been transmitted. This bit is valid only when the LS bit is set.
Bit 29	LS	rw	Last segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, neither TBS1 nor TBS2 can be cleared in the TDES1.
Bit 28	FS	rw	First segment When set, this bit indicates that the buffer contains the first segment of the frame.
Bit 27	DC	rw	Disable CRC When set, the MAC does not append a CRC field to the end of the transmitted frame. This bit is valid only when the FS bit is set (TDES0[28]=1).
Bit 26	DP	rw	Disable pad 0: The MAC automatically adds padding to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]). This bit is valid only when the FS bit is set (TDES0[28]). 1: The MAC does not automatically add padding to a frame shorter than 64 bytes.
Bit 25	TTSE	rw	Transmit time stamp enable When this bit is set, the IEEE1588 hardware time stamp is activated for the transmit frame described the descriptor. This bit is valid only when both the TSE (EMAC_PTPTSCTRL[0]) and FS bits (TDES0[28]) are set.
Bit 24	Reserved	resd	Kept at its default value.
Bit 23: 22	CIC	rw	Checksum insertion control These two bits control the checksum calculation and insertion, as shown below: 00: Checksum insertion disabled 01: Only IP header checksum calculation and insertion are enabled 10: IP header checksum and data checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated. 11: IP header checksum and data checksum calculation and insertion are enabled, and pseudo-header checksum is calculated.
Bit 21	TER	rw	Transmit end of ring

			When set, it indicates that the descriptor list reached its final descriptor. The DMA returns to the start address of the list, creating a descriptor ring.
Bit 20	TCH	rw	Second address chained When set, it indicates that the TBS2 bit in the TDES1 refers to the second descriptor address rather than the second buffer address. TDES0[21] takes precedence over TDES0[20]. This bit is valid only when the TDES0[28] is set.
Bit 19: 18	Reserved	resd	Kept at its default value.
Bit 17	TTSS	rw	Transmit time stamp status This bit is used as a status bit to indicate that a time stamp is captured for the described transmit frame. When this bit is set, it indicates the time stamp of the transmitted frame described by the descriptor has been captured, which are stored in the TDES2 and TDES3. This bit is valid only when the LS bit is set (TDES0[29]).
Bit 16	IHE	rw	IP header error When set, it indicates that the MAC transmitter detected an error in the IP data packet header. For IPv4 frames, the MAC checks whether the length field in the IPv4 datagram does match the number of he received IPv4 data. If there is a mismatch, an error is given. For IPv6 frames, an error is reported when the header length is not 40 bytes. Furthermore, the length/type field value for an IPv4 or IPv6 frame must match the IP header version. For IPv4 frame, an error is reported and this bit is set if the header length field value is shorter than 0x5.
Bit 15	ES	rw	Error summary This bit indicates the logical OR of the following bits: TDES0[14]: Jabber timeout TDES0[13]: Frame flush TDES0[11]: Loss of carrier TDES0[10]: No carrier TDES0[9]: Late collision TDES0[8]: Excessive collision TDES0[2]: Excessive deferral TDES0[1]: Underflow error TDES0[16]: IP header error TDES0[12]: IP data error
Bit 14	JT	rw	Jabber timeout When set, this bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JAD bit is not set in the EMAC_MACCTRL register.
Bit 13	FF	rw	Frame flushed When set, this bit indicates that the DMA or MTL flushed the frame in the FIFO due to a flush command given by the CPU.
Bit 12	IPE	rw	IP payload error When set, this bit indicates that the MAC transmitter detected an error in the TCP, UDP or ICMP. The transmitter compares the length field received in the IPv4 or IPv6 with the actual number of TCP, UDP or ICMP bytes. This bit is set as an error warning if there is a match.
Bit 11	LOC	rw	Loss of carrier When set, this bit indicates that a loss of carrier occurred during a frame transmission (The MII_CRS signal is active for one or more transmit clock periods). This bit is valid only for the frame transmitted without collision while the MC operates in half-duplex mode.
Bit 10	NC	rw	No carrier When set, this bit indicates that the carrier sense signal from the PHY was not set during a frame transmission.
Bit 9	LC	rw	Late collision When set, this bit indicates that frame transmission was aborted due to a collision detected after the collision window (64 byte times, including preamble, in MII mode). This bit is invalid if the underflow error bit is set.
Bit 8	EC	rw	Excessive collision When set, this bit indicates that the frame transmission was aborted after 16 consecutive collisions while attempting to transmit the current frame. If the RD bit (Retry disabled) bit in the EMAC_MACCTRL register is set, then this bit is set after the first collision, and the transmission of the frame is aborted.
Bit 7	VF	rw	VLAN frame When set, this bit indicates that the transmitted frame is a VLAN-type frame.
Bit 6: 3	CC	rw	Collision count

			This field indicates the number of collisions experienced before the frame was transmitted. This bit is invalid when the EC bit is set (TDES0[8]). It is valid only in half-duplex mode.
Bit 2	ED	rw	Excessive deferral When set, this bit indicates that the transmission ended because of excessive deferral of over 24288 bit times when the DC bit is set in the EMAC_MACCTRL register.
Bit 1	UF	rw	Underflow error When set, this bit indicates that the MAC stopped the frame transmission because data arrived late from the system memory to the MAC. Underflow error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the suspend state and sets bot the bit 5 (TU) in the EMAC_DMASTS register and the transmit interrupt bit (bit 0 in the EMAC_DMASTS register).
Bit 0	DB	rw	Deferred bit When set, this bit indicates that the MAC defers frame transmission because of the presence of the carrier. This bit is valid only in half-duplex mode.

TDES1: Transmit descriptor word1

Bit	Name	Type	Description
Bit 31: 29	Reserved	resd	Kept at its default value.
Bit 28: 16	TBS2	rw	Transmit buffer 2 size This field indicates the second data buffer size in bytes. When the TDES0[20] bit is set, this field indicates the second descriptor address.
Bit 15: 13	Reserved	resd	Kept at its default value.
Bit 12: 0	TBS1	rw	Transmit buffer 1 size This field indicates the first data buffer size in bytes. If the field is 0, the DMA ignores this buffer and uses buffer 2 or the next buffer, depending on the TDES0[20] bit.

TDES2: Transmit descriptor word2

TDES2 contains the address pointer to the first buffer of the descriptor.

Bit	Name	Type	Description
Bit 31: 0	TBAP1/TTSL	rw	Transmit buffer 1 address pointer / Transmit frame time stamp low This field has two functions: 1: The application indicates to the DMA the location of the Ethernet data in system memory. 2: After all data are transferred, the DMA can use these bits to store the time stamp of the transmit frame.
	TBAP1		When the current descriptor is owned by the DMA, these bits indicate the physical address of the buffer 1.

TDES3: Transmit descriptor word 3

TDES3 contains the address pointer to the second buffer of the descriptor.

Bit	Name	Type	Description
Bit 31: 0	TBAP2/TTSH	rw	Transmit buffer 2 address pointer (Next descriptor address) / Transmit frame time stamp high This field has two functions: 1: The application indicates to the DMA the location of the Ethernet data in system memory. 2: After all data are transferred, the DMA can use these bits to store the 32 most significant bits of the time stamp for the frame.
	TBAP2		When the current descriptor is owned by the DMA, these bits indicate the physical address of buffer2 if a descriptor ring structure is used. If a descriptor chain structure is used, these bits indicate the physical address of the next descriptor.

TXDMA enhanced descriptor

Under the two conditions below, enhanced TXDMA descriptors must be used.

1. Time stamping is activated (by setting TE to 1 in the EMAC_PTPTSCTRL register)
2. IPV4 checksum is activated (by setting IPC to 1 in the EMAC_MACCTRL register)

Enahced TXDMA descriptor feature is enabled by setting EDE to 1 in the EMAC_DMABM register. Enhanced TX descriptors include TDES0,TDES1,TDES2,TDES3,TDES4,TDES5,TDES6 and TDES7, 8x 32-bit words in total.

Within them, TDES0~3 have the same functions as regular TX descriptors; TDES4~5 are unused; TDES6~7 are to store 64-bit time stamp.

TDES6: Transmit descriptor word 6

TDES6 contains the 32 least significant bits of the time stamp captured

Bit	Name	Type	Description
Bit 31: 0	TTSL	rw	The DMA updates this field with the 32 least significant bits of the time stamp captured for the corresponding transmit frame to this filed before releasing descriptors to CPU. The DMA updates this field only after the time stamp is enabled (setting bit 25 TTSE to 1 in the TDES0) and LS bit is to 1 (which means the completion of the current frame transmission).

TDES7: Transmit descriptor word 7

TDES7 contains the 32 most significant bits of the time stamp captured.

Bit	Name	Type	Description
			Transmit frame time stamp high The DMA updates this field with the 32 most significant bits of th time stamp captured for the corresponding transmit frame. Th DMA updates this field only after the time stamp is enable (setting bit 25 TTSE to 1 in the TDES0) and LS bit is to 1 (whic means the completion of the current frame transmission).

RXDMA configuration

3. The application sets up receive descriptors (RDES0~RDES3), sets the OWN bit and then releases the descriptors to the DMA.
4. When the SSR bit (EMAC_DMAOPM[1]) is set, the DMA enters run state and attempts to acquire receive descriptors. If the fetched descriptor is not free (owned by the CPU), the DMA enters suspend state and jumps to Step 9.
5. The DMA decodes the receive buffer address from the acquired receive descriptors.
6. The DMA writes the frame data in the RXFIFO to the receive buffer.
7. When the buffer is full or the frame transfer ends, the receive controller will fetch the next descriptor from the descriptor queue.
8. If the current frame transfer is complete, the DMA jumps to Step 7. If the OWN bit of the next receive descriptor is cleared while the current frame is not complete (EOF is not received), when the frame flushing function is enabled, the DMA sets the descriptor error bit in the RDES0, closes the current descriptor (OWN=0) and sets the LS bit in the RDES1, and then jumps to Step 8 (Note that the LS bit in the RDES1 will not be set if the frame flushing feature is disabled). When the OWN bit of the next descriptor is set while the current frame transfer is not complete, the DMA then closes the current descriptor, marks it as intermediate and jumps to Step 4.
9. If IEEE1588 time stamp is enabled, the DMA writes the time stamp to the current descriptor's RDES2 and RDES3 while it writes the status word to the RDES0, with the OWN bit cleared and the LS bit set.
10. The receive controller checks the latest receive descriptors, if the DMA owns the descriptor, the receive controller will return to Step 4. If the CPU owns the descriptor, the RXDMA will enter suspend state and set the receive buffer unavailable bit, and the controller will flush the received frames if the receive frame flushing feature is enabled.
11. The DMA exits the suspend state when a receive poll demand is received or the start of the next frame is available in the receive FIFO. The receive controller fetches the next descriptor and jumps to Step 2.

Receive descriptor acquisition

The RXDMA always attempts to acquire another descriptor. Receive descriptor is attempted if any of the following conditions is satisfied:

- The DMA enters the run state (SSR=1 in the EMAC_DMAOPM).
- The buffer of the current descriptor is full before a full or part of the frame being transferred.
- The controller has completed the current frame reception, but the current receive descriptor has not yet been closed.
- A new frame is received but the receive process is suspended because the descriptor is owned by the CPU.
- A receive poll demand received.

Receive frame processing

The MII/RMI interface receives the receive frame and writes it to the RXFIFO. When a programmed threshold is reached, the RXDMA begins transferring the frame data to the receive buffer pointed to by the current descriptor. The DMA sets the LS bit in the RDES0 to indicate that this is the first segment of the frame in the buffer. The descriptors are released by clearing the OWN bit when the data buffer fills up or at the end of the frame reception. If the current descriptor buffer is enough to accommodate the complete frame, the current description RDES0 LS and FS bits are set.

The DMA fetches the next descriptor, sets the LS bit in the previous descriptor, writes the receive status word to the previous descriptor and then closes the previous descriptor. Then the DMA sets the receive interrupt bit (EMAC_DMASTS[6]). The same process repeats unless the DMA finds a descriptor as being owned by the CPU. If this occurs, the receive process sets the receive buffer unavailable bit (EMAC_DMASTS[7]) and then enters the suspend state. Before the suspend state is being entered, the current pointer value in the descriptor list is retained, which is used as the start address of a descriptor after exiting the suspend state.

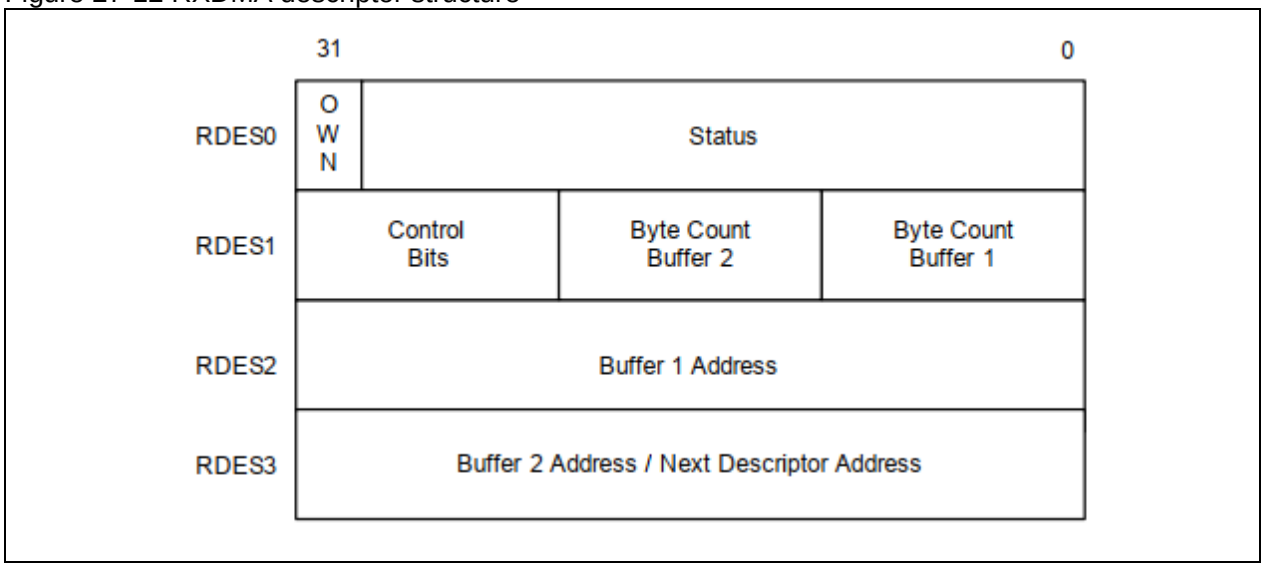
Receive process suspended

If a new receive frame arrives in the RXFIFO while the RXDMA is in suspend state, the DMA re-fetches the current descriptor in the AT32F457xx memory. If the OWN bit of the fetched descriptor is set, the receive process re-enters the run state. If the OWN bit is cleared, the DMA discards the current frame at the top of the RXFIFO and increments the missed frame counter. If more than one frame is stored in the RXFIFO, the process repeats. The flushing or discarding of the frame at the top of the receive FIFO can be avoided by setting the bit 24 (DFRF bit) in the EMAC_DMAOPM register. In this case, the receive process sets the receive buffer unavailable bit and returns to the suspend state.

RXDMA descriptors

The descriptor structure consists of four 32-bit words: RDES0, RDES1, RDES2 and RDES3.

Figure 27-12 RXDMA descriptor structure



RDES0: Receive descriptor word0

RDES0 contains the receive frame state, the frame length and the descriptor ownership information.

Bit	Name	Type	Description
Bit 31	OWN	rw	Own bit 0: The descriptor is owned by the CPU 1: The descriptor is owned by the DMA This bit is cleared by the DMA when the DMA completes the frame transmission or when the buffers associated with this descriptor are full. The descriptor is released to the CPU.
Bit 30	AFM	rw	Destination address filter fail When set, this bit indicates that a frame failed the DA filter in the MAC core.
Bit 29: 16	FL	rw	Frame length These bits indicate the byte length of the received frame that was transferred to the system memory by the DMA. This field is valid only when the LS bit (RESD0[8]) is set and descriptor error bit is cleared (RDES0[14). If the LS bit is cleared, this field indicates the accumulated number of bytes that have been transferred to the system memory. Whether the frame length includes CRC depends on the bit 7 and bit 25 in the EMAC_MACCTRL register.
Bit 15	ES	rw	Error summary This bit indicates the logical OR of the following bits: RDES0[1]: CRC error RDES0[3]: Receive error RDES0[4]: Watchdog timeout RDES0[6]: Late collision RDES0[7]: Giant frame or IP checksum error RDES0[11]: Overflow error RDES0[14]: Descriptor error
Bit 14	DE	rw	Descriptor error When set, this bit indicates a frame truncation caused by a frame that does not fit with the current descriptor buffers, and that the DMA does not own the next descriptor. This bit is valid only when the LS bit (RDES0[8])) is set.
Bit 13	SAF	rw	Source address filter fail When set, this bit indicates that the received frame failed the SA filter in the MAC core.
Bit 12	LE	rw	Length error When set, this bit indicates that the actual length of the received frame does not match the value in the Ethernet length/type field. This bit is valid only when the RDES0[5] bit is cleared. It is invalid when a CRC error occurs.
Bit 11	OE	rw	Overflow error When set, this bit indicates that the received frame was damaged due to receive FIFO overflow.
Bit 10	VLAN	rw	VLAN tag When set, this bit indicates that the frame pointed to by the current descriptor is marked as a VLAN frame by the MAC.
Bit 9	FS	rw	First descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the buffer of the next descriptor contains the beginning of the frame.
Bit 8	LS	rw	Last descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame.
Bit 7	IPHCE	rw	IPv header checksum error When set, this bit indicates an error in the IPv4 or IPv6 header. This error can be due to mismatched Ethernet type filed and IP version field, IPv4 header checksum error or an Ethernet frame lacking the desired number of IP header bytes.
Bit 6	LC	rw	Late collision When set, this bit indicates that a late collision has occurred while receiving the frame in half-duplex mode.
Bit 5	FT	rw	Frame type When set, this bit indicates that the received frame is an Ethernet frame (the LT field is greater than or equal to 1536). When this bit is cleared, it indicates that the received frame is an IEEE802.3 frame. This bit is invalid when the received frame

			is a runt frame shorter than 14 bytes.
Bit 4	RWT	rw	Receive watchdog timeout When set, this bit indicates the receive watchdog timeout has occurred while receiving the current frame. The current frame is truncated after the watchdog timeout.
Bit 3	RE	rw	Receive error When set, this bit indicates that the RX_ER signal is valid while the RX_DV is valid during frame reception.
Bit 2	DE	rw	Dribble bit error When set, this bit indicates that the received frame is not an integer multiple of bytes (odd nibbles) This bit is valid only in MII mode.
Bit 1	CE	rw	CRC error When set, this bit indicates a CRC error occurred on the received frame. This bit is valid only when the LS bit is set.
Bit 0	PCE	rw	Payload checksum error When set, this bit indicates that the TCP, UDP or ICMP checksum calculated by the MAC core does not match the received TCP, UDP or ICMP checksum field. This bit is also set when the received payload size does not match the length filed value of the IPv4 or IPv6 datagram in the received Ethernet frame.

Table 27-7 shows the definitions of bit 5, 7 and 0.

Table 27-7 Receive descriptor 0

Bit 5: Frame type	**Bit 7: Checksum error**	**Bit 0: Payload checksum error**	**Frame status**
0	0	0	IEEE802.3 type frame (length filed value is less than 0x0600)
1	0	0	IPv4/IPv6 type frame, no checksum error detected
1	0	1	IPv4/IPv6 type frame with payload checksum error detected (PCE bit)
1	1	0	IPv4/IPv6 type frame with an IP header checksum error detected (IPC CE bit)
1	1	1	IPv4/IPv6 type frame with both IP header and payload checksum errors detected
0	0	1	IPv4/IPv6 type frame with no IP header checksum error detected and the payload check bypassed due to an unsupported payload
0	1	1	A type frame is neither IPv4 nor IPv6 (the checksum offload bypasses checksum inspection)
0	1	0	Reserved

RDES1: Receive descriptor 1

Bit	Name	Type	Description
Bit 31	DIC	rw	Disable interrupt on completion When set, this bit prevents setting the Ethernet DMA status register's RECV bit (EMAC_DMASTS) for the received frame pointed to by this descriptor. As a result, this disables the interrupt triggered by the RECV bit. This bit is valid only when the RDES0[8] is set.
Bit 30: 29	Reserved	resd	Kept at its default value.
Bit 28: 16	RBS2	rw	Receive buffer 2 size This field indicate the receive buffer 2 size, in bytes. It is invalid if the RDES1[14 bit is set.
Bit 15	RER	rw	Receive end of ring When set, this bit indicates that the current descriptor is the last one in the descriptor list. The DMA returns to the base address to fetch the next descriptor, creating a descriptor ring.
Bit 14	RCH	rw	Second address chained When set, this bit indicates that the second address in the descriptor is the next descriptor address rather than the second buffer address. When this bit is set, RBS2(TDES1[28: 16]) value can be ignored. RDES0[15] takes precedence over RDES0[14].
Bit 13	Reserved	resd	Kept at its default value.
Bit 12: 0	RBS1	rw	Receive buffer 1 size This field indicates the receive buffer 1 size, in bytes. If this field is 0, the DMA ignores this buffer and uses buffer 2 or next descriptor depending on the RDES[14] value.

RDES2: Receive descriptor word 2

RDES2 contains the address pointer to the first data buffer in the descriptor,.

Bit	Name	Type	Description
Bit 31: 0	RBAP1/RTSL	rw	Receive buffer 1 address pointer /Receive frame time stamp low These bits have two functions. The application uses them to indicate to the DMA where to store the data in memory. After completing data reception, the DMA may use these bits to store frame time stamp.
	RBAP1		When this descriptor is owned by the DMA, these bits indicate the physical address of buffer 1.

RDES3: Receive descriptor word 3

RDES3 contains the address pointer to the second data buffer in the descriptor,.

Bit	Name	Type	Description
Bit 31: 0	RBAP2/RTSH	rw	Receive buffer 2 address pointer (next descriptor address)/Receive frame time stamp high Bit [31:0] These bits have two functions. The application uses them to indicate to the DMA where to store the data in memory. After completing data reception, the DMA may use these bits to store frame time stamp.
	RBAP2		When this descriptor is owned by the DMA, these bits indicate the physical address of buffer 2 when a descriptor ring structure is used. If the second address chained bit (RDES0[14]) is set, these bits point to the physical address of the next descriptor while the next descriptor is present.

RXDMA enhanced descriptor

Under the two conditions below, enhanced RXDMA descriptors must be used.

1. Time stamping is activated (by setting TE to 1 in the EMAC_PTPTSCTRL register)
2. IPV4 checksum is activated (by setting IPC to 1 in the EMAC_MACCTRL register)

Enahced RXDMA descriptor feature is enabled by setting EDE to 1 in the EMAC_DMABM register. Enhanced RX descriptors include RDES0,RDES1,RDES2,RDES3,RDES4,RDES5,RDES6 and RDES7, 8x 32-bit words in total.

Within them, RDES0~3 have the same definitions as for normal RX descriptors; RDES4 contains extended status; RDES6~7 hold 64-bit time stamp.

RDES4: Receive descriptor word 4

Bit	Name	Type	Description
Bit 31: 14	Reserved	resd	Kept at default value
Bit 13	PV	ro	PTP version 0: IEEE 1588 version 1 1: IEEE 1588 version 2
Bit 12	PFT	ro	PTP frame type When this bit is set to 1, it indicates that the PTP message is sent directly over Ethernet. When this bit is cleared and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information on IPv4 or IPv6 can be obtained from bit 6 and bit 7.
Bit 11: 8	PMT	ro	PTP message type 0000: No PTP message received 0001: SYNC 0010: Follow_Up 0011: Delay_Req 0100: Delay_Resp 0101: Pdelay_Req/Announce 0110: Pdelay_Resp/Management 0111: Pdelay_Resp_Follow_Up/Signaling 1xxx: Reserved
Bit 7	IPV6PR	ro	IPv6 packet received When set to 1, this bit indicates that the received packet is an IPv6 packet.
Bit 6	IPV4PR	ro	IPv4 packet received When set to 1, this bit indicates that the received packet is an IPv4 packet.
Bit 5	IPCB	ro	IP checksum bypassed When set to 1, this bit indicates that the checksum offload engine is bypassed.
Bit 4	IPPE	ro	IP payload error When set to 1, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP or ICMP checksum) that the core calculated does not match the corresponding checksum filed in the received segment. It is also set to 1 when the TCP, UDP or ICMP segment length does not match the payload length value in the IP header field.
Bit 3	IPHE	ro	IP header error When set to 1, this bit indicates that the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or that the IP datagram version is not consistent with the Ethernet Type value.
Bit 2: 0	IPPT	ro	IP payload type If IPv4 checksum offload is activated (IPCO=1, ETH_MACCR bit 10), these bits indicate the type of payload encapsulated in the IP datagram. These bits are “00” if there is an IP header orrr or fragmented IP. 000: Unkown or did not process IP payload 001: UDP 010: TCP 011: ICMP 1xx: Reserved

RDES6: Receive descriptor word 6

RDES6 contains the 16 least significant bits of the time stamp

Bit	Name	Type	Description
Bit 31: 0	RTSL	rw	The DMA updates this field with the 32 least significant bits of the time stamp captured for the corresponding receive frame. The DMA updates this field only when the time stamp is enabled and the LS bit is set to 1 (that is, the last segemtn of the frame is stored in the buffer area)

RDES7: Receive descriptor word 7

RDES7 contains the 16 most significant bits of the time stamp

Bit	Name	Type	Description
Bit 31: 0	RTSH	rw	The DMA updates this field with the 32 most significant bits of the time stamp captured for the corresponding receive frame. The DMA updates this filed only when the time stamp is enabled and the LS bit is set to 1 (that is, the last segment of the frame is stored in the buffer area)

27.2.4 Enter and wake up EMAC power-down mode

The EMAC enters power-off mode when the PD bit is enabled in the EMAC_MACPMTCTRLSTS register. In this mode, all received frames are dropped by the EMAC and they are not forwarded to the application.

PMT supports the reception of remote wakeup frames and AMD Magic Packet frames and uses them to wake up the EMAC from power-off mode. This is done by setting the ERWF and EMP bits in the EMAC_MACPMTCTRLSTS register.

Remote wakeup frame filter register

There are eight wakeup frame filter registers, each of which requires to be configured one by one. The desired values of the wakeup frame filter are loaded by sequentially loading eight times the wakeup frame filter register. The read operation is identical to the write operation. To read the eight values, the user has to read the wakeup frame filter register for consecutive eight times.

Figure 27-13 Wakeup frame filter register

Wkuppktfilter_reg0	Filter 0 Byte Mask							
Wkuppktfilter_reg1	Filter 1 Byte Mask							
Wkuppktfilter_reg2	Filter 2 Byte Mask							
Wkuppktfilter_reg3	Filter 3 Byte Mask							
Wkuppktfilter_reg4	RESD	Filter 3 Cmd	RESD	Filter 2 Cmd	RESD	Filter 1 Cmd	RESD	Filter 0 Cmd
Wkuppktfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
Wkuppktfilter_reg6	Filter 1 CRC-16				Filter 0 CRC-16			
Wkuppktfilter_reg7	Filter 3 CRC-16				Filter 2 CRC-16			

Filter i byte mask

This register defines which bytes of the filter i (i=0~3) are used to determine whether or not the frame is a wakeup frame. The bit 31 must be zero. The bit j[30: 0] is the byte mask. If the bit j is set, then filter i offset + j of the incoming frame will be processed by the CRC block; otherwise filter i offset + j is ignored.

Filter i command

This is a 4-bit command. Bit 3 defines the address type. When the bit is set, this feature applies to only multicast addresses. When the bit is cleared, this feature applies to only unicast addresses. Bit 2 and bit 1 are reserved. Bit 0 is the filter enable bit. This filter is disabled if the bit 0 is cleared.

Filter i offset

This is an 8-bit register that defines the offset for the filter i first byte to be examined by filter i. The minimum allowed is 12, which refers to the 13th byte of the frame (offset value 0 means the first byte of the frame)

Filter i CRC-16

This register contains the CRC_16 value calculated by the filter, and the byte mask value programmed in the wakeup frame filter register block.

Remote wakeup frame detection

This mode is enabled by setting the RRWF bit in the EMAC_MACPMTCTRLSTS register.

PMT supports four programmable filters. If the incoming frame passed the address filtering of the filter, and if the filter CRC_16 matches the examined incoming frame, then the wakeup frame is received. PMT is only responsible for checking length error, FCS error, Dribble bit error, MII error, collision and ensuring that the wakeup frame is not a runt frame.

When a remote wakeup frame is received, the EMAC will move from sleep mode to normal mode. At the same time, the RRWF bit (bit 6) is set in the EMAC_MACPMTCTRLSTS register, indicating that a remote wakeup frame is received. If a remote wakeup interrupt is enabled, an interrupt will be generated when the PMT receives the remote wakeup frame.

Magic Packet detection

Magic Packet detection is enabled by setting the EMP bit in the EMAC_MACPMTCTRLSTS register. The Magic Packet frame contains a specific packet of information that is used to wakeup the stations on the LAN.

Magic Packet frame format: 6 bytes are all 1, followed by a MAC address repeating 16 times, for instance, MAC address of a device is 0x11aabb22cc33, then the Magic Packet used to wake up this frame is shown as follows:

Destination address source address ……FFFF FFFF FFFF

11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33

11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33

11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33

11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33 11aa bb22 cc33

… CRC

In Sleep mode, the PMT will constantly detect each frame transferred to the station to determine whether or not the frame meets the Magic Packet format.

When the Magic Packet is received, the RMP bit will be updated in the EMAC_MACPMTCTRLSTS register. Upon the reception of Magic Packet, the PMT will generate an interrupt if enabled.

Precautions on system in DEEPSLEEP mode

In DEEPSLEEP mode, the Ethernet PMT block is still able to detect frames as long as the EXTI 19 is enabled. However, the RE bit has to be set in the EMAC_MACCTRL register because the EMAC needs to detect Magic Packet or a remote wakeup frame.

DEEPSLEEP and wakeup sequences are recommended as follows:

1. Disable the TXDMA and wait for all data transmission to complete. These transmissions can be checked by polling the TI bit in the EMAC_DMASTS register.
2. Disable the MAC transmitter and MAC receiver by clearing the TE and RE bits in the EMAC_MACCTRL register.
3. Poll the RI bit in the EMAC_DMASTS register, and wait for the RXDMA to real all the frames in the RXFIFO, and then disable the RXDMA.
4. Configure and enable the EXTI19 to generate either an event or an interrupt.
5. Enable Magic Packet/remote wakeup frame detection by setting the EMP/ERWF bit in the EMAC_MACPMTCTRLSTS register.
6. Enable power-down mode by setting the PD bit in the MAC_MACPMTCTRLSTS register.
7. Enable the EMAC receiver by setting the RE bit in the EMAC_MACCTRL register.

8. MCU enters DEEPSLEEP mode.
9. Upon receiving Magic Packet/a remote wakeup frame, the Ethernet exits the power-down mode.
10. Read the EMAC_MACPMTCTRLSTS register to clear RRWF/RMP, enable the EMAC transmit state machine, and the TXDMA and RXDMA.
11. Configure the MCU system clock.

27.2.5 IEEE1588 precision time protocol

The PTP is used to synchronize systems that include clocks of varying precision, resolution and stability (not limited to Ethernet). Refer to IEEE1588 standard for more information.

The PTP block captures the accurate time when a PTP packet is transmitted or received from Ethernet port, and the captured time is returned to the application.

Reference clock source

According to IEEE158 standard, the system requires a reference time in a 64-bit format as the current time record, with the upper 32 bits time information in seconds, and the lower 32 bits time information in nanoseconds.

The PTP reference clock is used to generate the system time and to capture time stamps. The frequency of this reference clock must be greater or equal to the resolution of time stamp counter. The synchronization accuracy between the master node and the slave node is around 100ns.

The time synchronization accuracy depends on the PTP reference clock input period, the frequency drift of the crystal oscillator and that of the synchronization procedure.

Transmission and reception of frames with PTP feature

When the bit 0 is set in the EMAC_PTPTSCTRL register and the bit 25 is also set in the TDES0 register, a frame's SFD is output on the MII, and then a time stamp is captured. The upper 32 bits and the lower 32 bits of the time stamp are stored in the TDES3 and TDES2, respectively so that the time stamp and transmit status work will be returned to the application all together.

When the bit 0 is set in the EMAC_PTPTSCTRL register and the bit 8 is also set in the EMAC_PTPTSCTRL register, the EMAC will capture the time stamp of all the received frames on the MII. The upper 32 bits and the lower 32 bits of the time stamp are stored in the RDES3 and RDES2, respectively so that the time stamp and receive status work will be returned to the application all together.

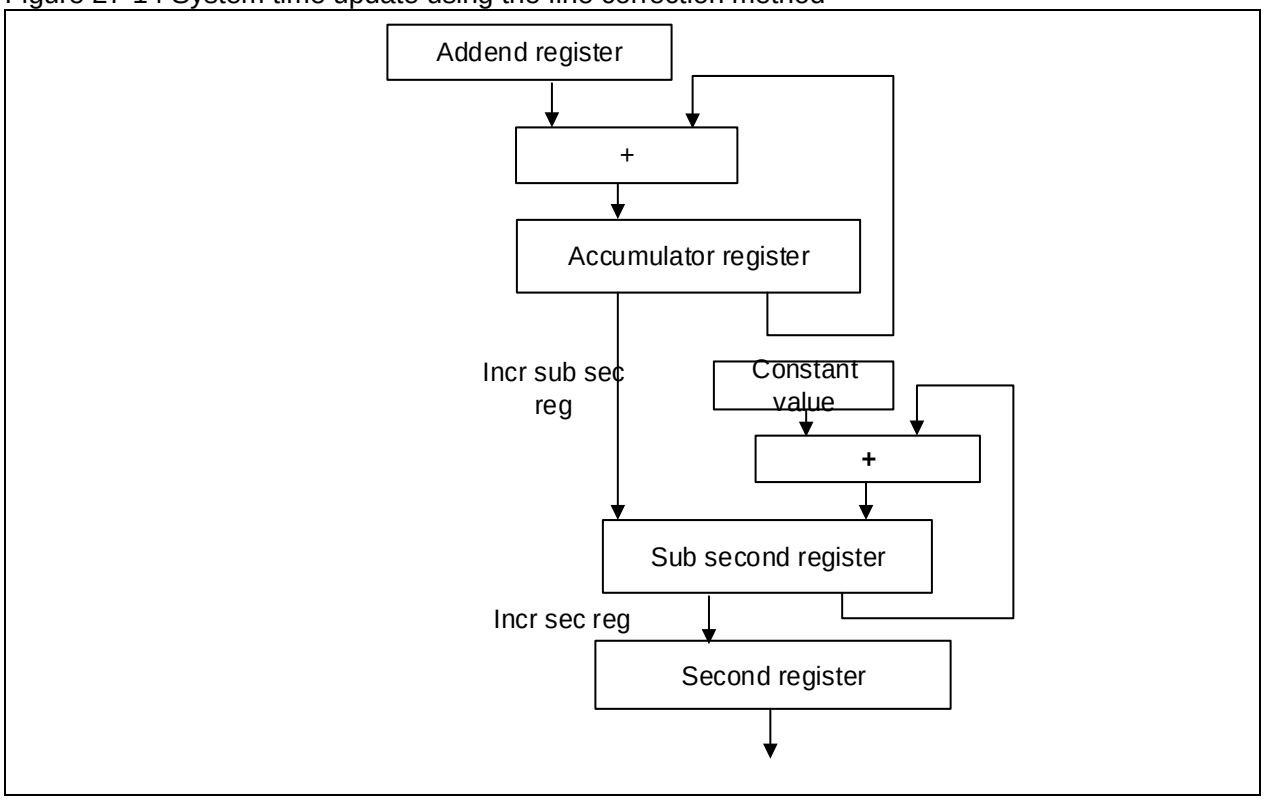
System time correction methods

The PTP input reference clock is the system clock SYSCLK, which is used to update the 64-bit time stamp of the Ethernet frames being transmitted or received. There are two correction methods: coarse and fine correction.

Coarse correction: the initial value/the offset value is written to the time stamp update registers (EMAC_PTPTSHUD and EMAC_ PTPTSLUD). When the TI bit is set in the EMAC_PTPTSCTRL register, an initialization process starts, and system clock counter is updated with the value in the time stamp update register. If the TU bit is set in the EMAC_PTPTSCTRL register, a correction process starts, and the time stamp update register value is uses as the offset value, and such offset value is added or subtracted from the system time.

Fine correction: the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE1588) is corrected over a period of time. An accumulator sums up the value of the addend register as shown in Figure 26-15.The pulse generated by the arithmetic carry of the accumulator is used to increment the system time counter. The addend register value depends on the system clock frequency. Both the accumulator and the addend are 32-bit registers.

Figure 27-14 System time update using the fine correction method



The subsecond register update frequency requires 50 MHz to achieve 20 ns accuracy for the system clock update circuit. Therefore, if the system clock frequency is 70 MHz, the ratio is calculated as 70/50=1.4. Thus the value written to the addend register is 2^{32}/1.4, which is equal to 0XB6DB6DB6.

The value in the addend register has to be updated according to the drift of the system clock frequency. If the subsecond register update frequency is 50 MHz, the constant value used to increment the subsecond register is $2^{31}/(50*10^6)$=43.

The software has to calculate the drift of the frequency by means of Sync, and to update the accumulator register accordingly. Initially, the slave clock addend register is programmed to be FreqCompensationValue0:

FreqCompensationValue0 = 2^{32}/frequency division ratio. If the MasterToSlaveDelay is assumed to be the same for consecutive Sync messages. The algorithm below mentioned must be applied. After a few Sync cycles, the frequency is locked. The slave clock can then determine an accurate MasterToSlaveDela value and synchronize the master with the salve clock using the new value. The algorithm is shown as follows:

When the master clock is $MasterSyncTime_n$, the master node sends the slave node a Sync message. The slave node receives this message when its local clock is $SlaveClockTime_n$, and computes $MasterClockTime_n$ as

$MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$

The master clock count for the current Sync cycle, $MasterClockCount_n$ is:

$MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$, assuming that the MasterToSlaveDelay is the same for Sync cycles n Sync cycles n-1

The slave clock count for the current Sync cycle, $SlaveClockCount_n$ is:

$SlaveClockCount_n = SlaveClockTime_n - SlaveClockTime_{n-1}$

The difference between the master clock count and slave clock count for the current Sync cycle, $ClockDiffCount_n$ is

$ClockDiffCount_n = MasterClockCount_n - SlaveClockCount_n$

The frequency-ratio factor for a slave clock, $FreqScaleFactor_n$ is

$FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$

The frequency compensation value for the addend register, $FreqCompensationValue_n$ is

$$FreqCompensationValue_n = FreqScaleFactor_n \times FreqCompensationValue_{n-1}$$

This algorithm comes with a self-correction feature. In theory, the frequency can be locked at a synchronized cycle. However, it may makes several cycles to synchronize the slave device.

System time initialization procedure

1. Mask the time stamp trigger interrupt by setting the bit 9 in the EMAC_MAIMR register.
2. Enable the time stamp by setting the bit 0 in the EMAC_PTPTSCTRL register.
3. Program the subsecond increment register based on the system time update precision.
4. If the fine correction method is being used, program the EMAC_PTPTSAD register, set the bit 5 in the EMAC_PTPTSCTRL register, poll the EMAC_PTPTSCTRL register until the bit 5 becomes 0. For the coarse correction method, skip this step and directly jump to step 6.
5. To select the fine correction method, program the bit 1 in the EMAC_PTPTSCTRL register.
6. Write the system time value to be configured to the EMAC_PTPTSHUD and EMAC_PTPTSLUD) registers.
7. Set the bit 2 in the EMAC_PTPTSCTRL register, and start initializing the time stamp and polling this bit until this bit becomes 0 (initialization complete).
8. The time stamp counter starts running as soon as the initialization is complete.
9. Enable the MAC transmitter and receiver for proper time stamping.

Programming steps for system time update using coarse correction method

1. Write the offset value (positive or negative) to the Ethernet PTP time stamp high update register (EMAC_PTPTSHUD) and the Ethernet PTP time stamp low update register (EMAC_PTPTSLUD).
2. Set the bit 3 (TU) in the Ethernet PTP time stamp control register (EMAC_PTPTSCTRL).
3. When the TU bit is cleared, the value in the time stamp update registers is added to or subtracted from the system time.

Programming steps for system time update using fine correction method

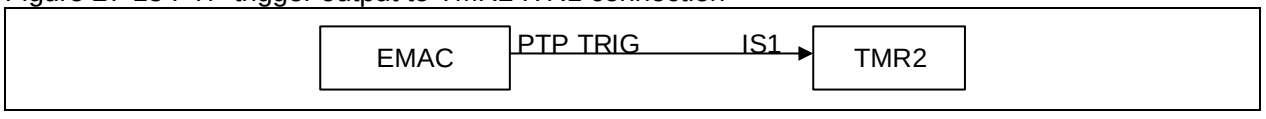
1. Use the algorithm explained in System time correction methods to calculate the value of the addend register.
2. Update the addend register.
3. Write the target value you want to the target time high and target time low registers, and clear the bit 9 in the Ethernet MAC interrupt mask register (EMAC_MAIMR) to activate the time stamp interrupt.
4. Set the bit 4 (TITE) in the Ethernet PTP time stamp control register (EMAC_PTPTSCTRL).
5. When this event generates an interrupt, read the EMAC_MAIMR register to clear the corresponding interrupt flag bits.
6. Reprogram the EMAC_PTPTSAD register with the old value and set the bit 5 in the EMAC_PTPTSCTRL register to update the addend register.

PTP trigger internal connection with TMR2

The EMAC provides a trigger interrupt when the system time is greater than the target time. Using an interrupt introduces an interrupt latency. To obtain an accurate interrupt latency time, a PTP will output a high signal to the TMR2 when the system time becomes greater than the target value. An accurate interrupt latency can be calculated since the clock of the timer and PTP reference clock are synchronous.

Set the bit [11:10]=01 in the TMR2_RMP register to enable the connection between the PTP trigger signal and the TMR2 IS1.

Figure 27-15 PTP trigger output to TMR2 ITR1 connection



PTP second pulse output signal

Refer to the EMAC_PTPPPSCR register descriptor for more information about PTP pulse second output. The following contents are based on the fact when the emac_pps_sel bit (bit 9) is cleared in the CRM_MISC2 register.

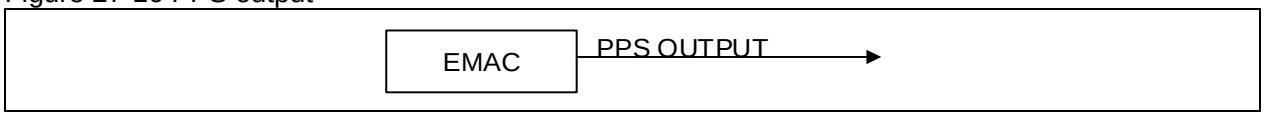
The PPS output frequency is 1 Hz by default, which can be configured to 2PPSFREQ Hz through the PPSFREQ[3: 0] in the EMAC_PTPPPSCR register.

If it is 1 Hz, the pulse width of the PPS is 125 ms when the binary rollover control is selected (TSSSR=0 in the EMAC_PTPTSCTRL register); the pulse width of the PPS is 100 ms when the digital rollover control is selected (TSSSR=1).

If it is 2 Hz or over, the duty cycle of the PPS output is 50% when the binary rollover control is selected.

Configure the PB5 multiplexed feature to enable PPS output feature.

Figure 27-16 PPS output



27.2.6 EMAC interrupts

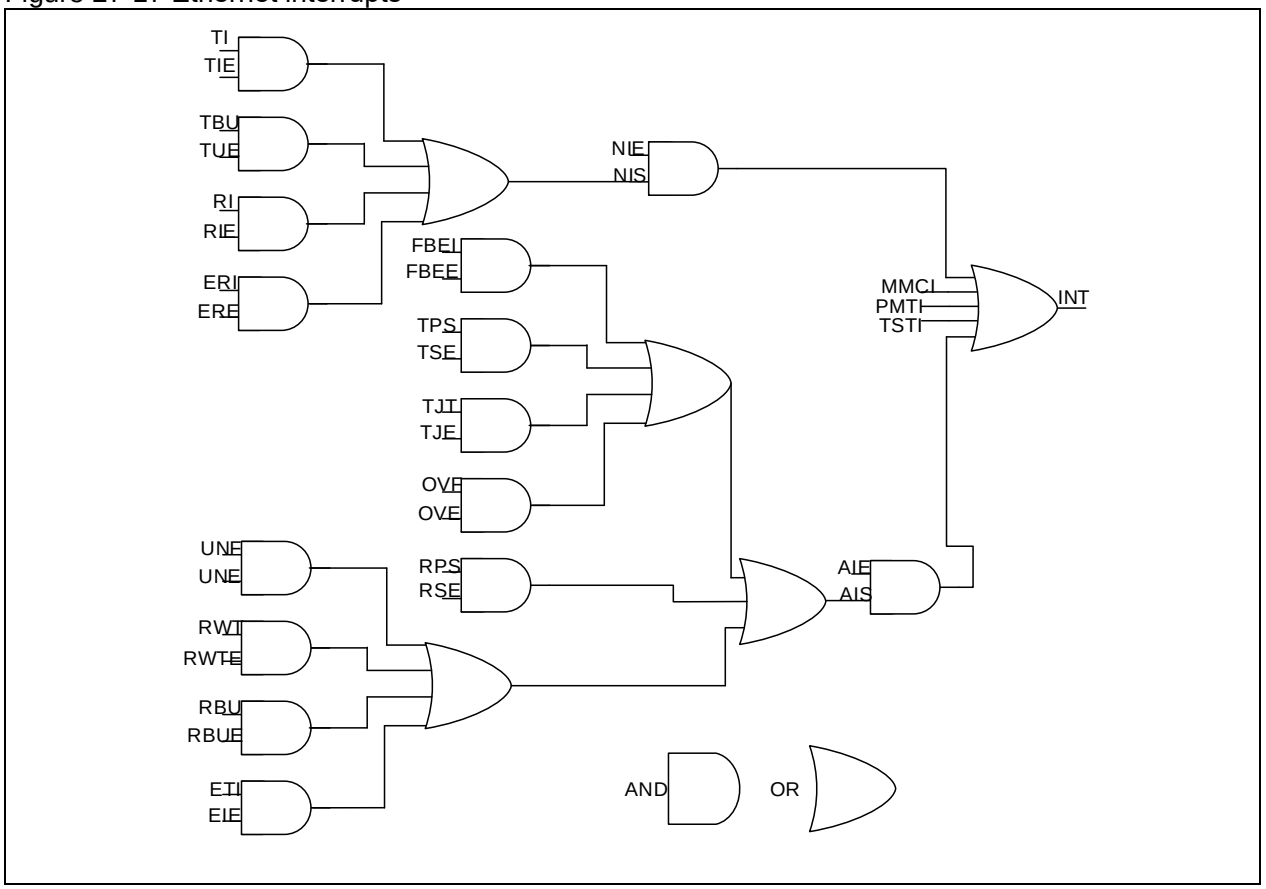
The EMAC has two interrupt vectors: one is used for normal Ethernet operations and the other for the Ethernet wakeup event (remote wakeup frame or Magic Packet detection) when it is mapped on EXINT line 19.

The first interrupt vector is used for interrupts generated by the MAC and the DMA.

The second interrupt vector is used for interrupts generated by the PMT block on wakeup events. It can cause the AT32F457xx to exit the low-power mode, and generate an interrupt.

When an Ethernet wakeup event is mapped on the EXINT line19 and the EMAC PMT interrupt is enabled and the EXINT line 19 interrupt, detected on its rising edge, is also enabled, both interrupts are generated at the same time.

Figure 27-17 Ethernet interrupts



27.3 EMAC registers

Table 27-8 shows the Ethernet register map and its reset values.

The peripheral registers can be accessed by bytes (8-bit), half words (16-bit) or words (32-bit).

Table 27-8 Ethernet register map and its reset values

Register	Offset	Reset value
EMAC_MACCTRL	0x00	0x0000 8000
EMAC_MACFRMF	0x04	0x0000 0000
EMAC_MACHTH	0x08	0x0000 0000
EMAC_MACHTL	0x0C	0x0000 0000
EMAC_MACMIIADDR	0x10	0x0000 0000
EMAC_MACMIIDT	0x14	0x0000 0000
EMAC_MACFCTRL	0x18	0x0000 0000
EMAC_MACVLT	0x1C	0x0000 0000
EMAC_MACRWFF	0x28	0x0000 0000
EMAC_MACPMTCTRLSTS	0x2C	0x0000 0000
EMAC_MACISTS	0x38	0x0000 0000
EMAC_MAIMR	0x3C	0x0000 0000
EMAC_MACA0H	0x40	0x0010 FFFF
EMAC_MACA0L	0x44	0xFFFF FFFF
EMAC_MACA1H	0x48	0x0000 FFFF
EMAC_MACA1L	0x4C	0xFFFF FFFF
EMAC_MACA2H	0x50	0x0000 FFFF
EMAC_MACA2L	0x54	0xFFFF FFFF
EMAC_MACA3H	0x58	0x0000 FFFF
EMAC_MACA3L	0x5C	0xFFFF FFFF
EMAC_MMCCTRL	0x100	0x0000 0000
EMAC_MMCRI	0x104	0x0000 0000
EMAC_MMCTI	0x108	0x0000 0000
EMAC_MMCRIM	0x10C	0x0000 0000
EMAC_MMCTIM	0x110	0x0000 0000
EMAC_MMCTFSCC	0x14C	0x0000 0000
EMAC_MMCTFMSCC	0x150	0x0000 0000
EMAC_MMCTFCNT	0x168	0x0000 0000
EMAC_MMCRFCECNT	0x194	0x0000 0000
EMAC_MMCRFAECNT	0x198	0x0000 0000
EMAC_MMCRGUFCNT	0x1C4	0x0000 0000
EMAC_PTPTSCTRL	0x700	0x0000 2000
EMAC_PTPSSINC	0x704	0x0000 0000
EMAC_PTPTSH	0x708	0x0000 0000
EMAC_PTPTSL	0x70C	0x0000 0000
EMAC_PTPTSH	0x708	0x0000 0000

EMAC_PTPTSL	0x70C	0x0000 0000
EMAC_PTPTSHUD	0x710	0x0000 0000
EMAC_PTPTSLUD	0x714	0x0000 0000
EMAC_PTPTSAD	0x718	0x0000 0000
EMAC_PTPTTH	0x71C	0x0000 0000
EMAC_PTPTTL	0x720	0x0000 0000
EMAC_PTPTSSR	0x728	0x0000 0000
EMAC_PTPPPSCR	0x72c	0x0000 0000
EMAC_DMABM	0x1000	0x0002 0101
EMAC_DMATPD	0x1004	0x0000 0000
EMAC_DMARPD	0x1008	0x0000 0000
EMAC_DMARDLADDR	0x100C	0x0000 0000
EMAC_DMATDLADDR	0x1010	0x0000 0000
EMAC_DMASTS	0x1014	0x0000 0000
EMAC_DMAOPM	0x1018	0x0000 0000
EMAC_DMAIE	0x101C	0x0000 0000
EMAC_DMAMFBOCNT	0x1020	0x0000 0000
EMAC_DMACTD	0x1048	0x0000 0000
EMAC_DMACRD	0x104C	0x0000 0000
EMAC_DMACTBADDR	0x1050	0x0000 0000
EMAC_DMACRBADDR	0x1054	0x0000 0000

27.3.1 Ethernet MAC configuration register (EMAC_MACCTRL)

The Ethernet MAC configuration register defines the receive and transmit operation modes.

A delay greater than 4μs is required for two consecutive write accesses to this register.

Bit	Name	Reset value	Type	Description
Bit 31: 24	Reserved	0x00	resd	Kept at its default value.
Bit 25	CST	0x0	rw	CRC Stripping for Type Frames When this bit is set, the last four bytes (FCS) of Ethernet type frame is stripped.
Bit 23	WD	0x0	rw	Watchdog Disable When this bit is set, the MAC disables the watchdog timer on the receiver, and can receive frames of up to 16,384 bytes. When this bit is cleared, the MAC allows no more than 2048 bytes of the frames being received.
Bit 22	JD	0x0	rw	Jabber Disable When this bit is set, the MAC disables the Jabber timer on the transmitter, and can transfer frames of up to 16,384 bytes. When this bit is cleared, the MAC cuts of the transmitter if the application sends out more than 2048 bytes of data during transmission.
Bit 21: 20	Reserved	0x0	resd	Kept at its default value.
Bit 19: 17	IFG	0x0	rw	InterFrame Gap These bits are used to define the minimum interframe gap between frames during transmission. 000: 96 bit times 96 bit times 001: 88 bit times 88 bit times 010: 80 bit times 80 bit times ...

				111: 40 bit times 40 bit times In half-duplex mode, the minimum IFG can be configured as 64 bit times (IFG=100). Lower values are not allowed.
Bit 16	DCS	0x0	rw	Disable Carrier Sense When this bit is set, the MAC transmitter will ignore the MII CRS signal during frame transmission in half-duplex mode. No error is reported due to loss of carrier or no carrier during transmission. When this bit is cleared, the MAC transmitter will report errors due to carrier sense and even abort the transmission. This bit is reserved in full-duplex mode.
Bit 15	Reserved	0x1	resd	Kept at its default value.
Bit 14	FES	0x0	rw	Fast EMAC Speed This bit indicates the speed of the MII, RMII interface. 0: 10 Mbps 1: 100 Mbps
Bit 13	DRO	0x0	rw	Disable Receive Own When this bit is set, the MAC disables the frame reception in half-duplex mode if the phy_txen_o is enabled. When this bit is cleared, the MAC will receive all packets that are given by the PHY during transmission. This bit is not applicable when the MAC is in full-duplex mode. This bit is reserved (with default value RO) when the MAC is configured as “For full-duplex mode only” mode.
Bit 12	LM	0x0	rw	Loopback Mode When this bit is set, the MAC MII operates in loopback mode. The MII receive clock input (clk_rx_i) is required for the loopback mode to work normally, for the transmit clock is not looped-back internally.
Bit 11	DM	0x0	rw	Duplex Mode When this bit is set, the MAC operates in full-duplex mode, in which it can transmit and receive simultaneously.
Bit 10	IPC	0x0	rw	IPv4 Checksum When this bit is set, the MAC calculates the 16-bit complement sum of all received Ethernet frames and enables IPv4 header checksum (assuming it is bytes 26-26 or 29-30 (VLANtagged)) for received frames, and gives the status in the receive status information. The MAC also appends the 16-bit checksum of the calculated IP header packets (bytes after the IPv4header), and adds it to the Ethernet frame that has been sent out to the application (when Type 2 COE is deselected). When this bit is cleared, this feature is disabled. When this bit is set, IPv4 header checksum feature and IPv4 or IPv6 TCP, UDP or ICMP payload checksum feature is enabled while the Type 2 COE is selected. When this bit is cleared, the COE function in the receiver is disabled, and the corresponding PCE and IP HCE status bits (see Table 3-10) are always 0. This bit is reserved (with default value RO) if the IP checksum mechanism is disabled during the core configuration.
Bit 9	DR	0x0	rw	Disable Retry When this bit is set, the MAC attempts only 1 transmission. When a collision occurs on the MII interface, the MAC will ignore the current frame transmission and report a frame abort because of excessive collision error in the transmit frame status. When this bit is cleared, the MAC attempts retries based on the settings of BL ([6: 5]). This bit is applicable only in half-duplex mode. It is reserved (with default value RO) in “For full-duplex mode only” mode.

Bit 8	Reserved	0x0	resd	Kept at its default value.
Bit 7	ACS	0x0	rw	Automatic pad/CRC Stripping When this bit is set, the MAC strips the pad/FCS field on received frames only when the frame length is shorter than 1536 bytes. All received frame with length field greater than or equal to 1536 bytes are passed on to the application without stripping the Pad or FCS field. When this bit is cleared, the MAC will forward all received frames to the master without changing its contents.
Bit 6: 5	BL	0x0	rw	Back-off Limit The Back-off limit defines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) the MAC waits before retries after a collision. This field is applicable only in the half-duplex mode. It is reserved (RO) in “For full-duplex mode only” mode. 00: k= min (n, 10) 01: k = min (n, 8) 10: k = min (n, 4) 11: k = min (n, 1) Where n = the number of slot time delays for retransmission attempt, and r takes the random integer value in the range 0 ≤ r < 2k.
Bit 4	DC	0x0	rw	Deferral Check When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a frame abort status and sets the excessive deferral error flag bit in the transmit frame status when the transmit state machine is delayed for more than 24288 bit times in 10/100 Mbit/s mode. If the Jumbo frame mode is enabled in 10/100 Mbps mode, the deferral threshold is 155680 bit times. Deferral begins when the transmitter is ready to transmit, but is prevented when an active carrier sense signals is detected on the MII. Deferral time is not cumulative. For instance, if the transmitter is deferred for 10000 bit times because that the CRS signals is active first, but then becomes inactive, then transmits, collides, backs off because of collision, and then has to defer again after the completion of back-off, the deferral times resets to 0 and restarts. When this bit is cleared, the deferral check function is disabled. The MAC defers until the CRS signal becomes inactive. This bit is applicable only in the half-duplex mode. It is reserved (RO) in “For full-duplex mode only” mode.
Bit 3	TE	0x0	rw	Transmitter Enable When this bit is set, the transmit state machine of the MAC is enabled. when this bit is cleared, the MAC disables the transmit state machine after the completion of the current frame transmission, and does not transmit any further frames (To modify this bit through consecutive commands, if needed, a deferral value greater than 4us is required between two consecutive operations)
Bit 2	RE	0x0	rw	Receiver Enable When this bit is set, the receive state machine of the MAC is enabled. when this bit is cleared, the MAC disables the receive state machine after the completion of the current frame reception, and does not receive any further frames (To modify this bit through consecutive commands, if needed, a deferral value greater than 4us is required between two consecutive operations).
Bit 1: 0	Reserved	0x0	resd	Kept at its default value.

27.3.2 Ethernet MAC frame filter register (EMAC_MACFRMF)

The Ethernet MAC frame filter register contains the filter control bits for receiving frames. Some of the control bits got to the address check block of the MAC to perform the first level of address filtering.

The second level of filtering is performed on the incoming frames based on other control bits (such as

pass bad frames and pass control frames).

Bit	Name	Reset value	Type	Description
Bit 31	RA	0x0	rw	Receive All When this bit is set, the MAC passes all received frames onto the application, irrespective of whether they have passed through the address filter. The result (pass or fail) of the source address or destination address filtering is updated in the corresponding bits of the receive status word. When this bit is cleared, the MAC passes on to the application only those frames that have passed the source address or destination address filtering.
Bit 30: 11	Reserved	0x00000	resd	Kept at its default value.
Bit 10	HPF	0x0	rw	Hash or Perfect Filter When this bit is set, the address filter passes frames that match the perfect filter or hash filter set by the HMC or HUC bit. When this bit is cleared, if the HUC or HMC bit is set, only frames that match the hash filter can pass address filter.
Bit 9	SAF	0x0	rw	Source Address Filter When this bit is set, the MAC compares the source address of the received frame with the value programmed in the enabled source address registers. If the comparison mismatches, the MAC will drop this frame. (SAF). When this bit is cleared, the MAC forwards the received frame to the application and updates the source address filter bit (SAF) in the receive status based on the source address comparison.
Bit 8	SAIF	0x0	rw	Source Address Inverse Filtering When this bit is set, the address check block operates in inverse filtering mode. The frame whose source address matches the source address register is marked as failing the source address filter. When this bit is cleared, the frame whose source address does not match the source address register is marked as failing the source address filter.
Bit 7: 6	PCF	0x0	rw	Pass Control Frames These bits control the forwarding of all control frames (including unicast and multicast Pause frames). 00: MAC filters all control frames and prevents them from reaching the application 01: MAC forwards all control frames, except Pause frame, to the application even if they fail the address filter 10: MAC forwards all control frames to the application even if they fail the address filter 11: MAC forwards control frames that pass the address filter to the application The following conditions must be met when dealing with a Pause frame: 1: When the MAC is in full-duplex mode, the bit 2 (REF) is set in the register 6 (flow control register) to enable flow control. 2: When the bit 3 (UP) is set in the register 6 (flow control register), the destination address of the received frames matches the specific multicast address or MAC address 0. 3: Type field of the receive frame is 0x8808, and the OPCODE field is 0x0001.
Bit 5	DBF	0x0	rw	Disable Broadcast Frames When this bit is set, the address filters filter all incoming broadcast frames. In addition, all other filter settings will also be overwritten. When this bit is set, the address filters pass all incoming broadcast frames.
Bit 4	PMC	0x0	rw	Pass MultiCast When this bit is set, all frames with a multicast destination

				address (first bit in the destination address is set) are passed. When this bit is cleared, the filtering of a multicast frame depends on the HMC bit.
Bit 3	DAIF	0x0	rw	Destination Address Inverse Filtering When this bit is set, the address check block operates in inverse filtering mode for the destination address comparison for both unicast and multicast frames. When this bit is cleared, the filter work normally.
Bit 2	HMC	0x0	rw	Hash MultiCast When this bit is set, the MAC performs destination address filtering of the received multicast frames according to the hash table. When this bit is cleared, the MAC performs a perfect destination address filtering for multicast frames, that is, it compares the destination address field with the values programmed in the destination registers. This bit is reserved if Hash filter is not selected during core configuration.
Bit 1	HUC	0x0	rw	Hash UniCast When this bit is set, the MAC performs destination address filtering for unicast frames according to the hash table. When this bit is cleared, the MAC performs a perfect destination address filtering for unicast frames, that is, it compares the destination address field with the values programmed in the destination registers.
Bit 0	PR	0x0	rw	Promiscuous Mode When this bit is set, the address filters pass all incoming frames regardless of their destination or source address. When the PR is set, the source address or destination address error bits in the receive status word are always 0.

27.3.3 Ethernet MAC Hash table high register (EMAC_MACHTH)

The 64-bit Hash table is used for group address filtering. For Hash filtering, the contents of the destination address of the incoming frame pass through the CRC logic, and the upper 6 bits in the CRC register are used to index the Hash table. The most significant bit of the CRC determines the register to be used (EMAC_MACHTH or EMAC_MACHTL), and the other 5 bits determine which bit in the register is to be used. The Hash value 5b'00000 uses the bit 0 in the selected register, while the Hash value 5b'11111 uses the bit 31 in the selected register.

The Hash value of the destination address is calculated according to the following steps:

1. Calculate a 32-bit CRC value of the destination address (see IEEE 802.3, and refer to 3.2.8 section for more details)
2. Bit invert the value obtained in Step 1
3. Take the upper 6 bits from the values obtained in Step 2

For example, if the destination address of the incoming frame is 0x1F52419CB6AF (0x1F is the first byte received on the MII interface), the calculated 6-bit Hash value is 0x2C and the bit 12 in the EMAC_MACHTH registers checked for filtering. If the destination address of the incoming frame is 0xA00A98000045, the calculated 6-bit Hash value is 0x07, and the bit 7 in the EMAC_MACHTL register is checked for filtering.

Bit	Name	Reset value	Type	Description
Bit 31	HTH	0x0000 0000	rw	This bit contains the upper 32 bits of the Hash table.

27.3.4 Ethernet MAC Hash table low register (EMAC_MACHTL)

The EMAC_MACHTL register contains the lower 32 bits of the Hash table. If the Hash filter is disabled or either 128-bit or 256-bit Hash table is selected, both register 2 and register 3 are reserved.

Bit	Name	Reset value	Type	Description
Bit 31	HTL	0x0000 0000	rw	Hash Table Low This bit contains the lower 32 bits of the Hash table.

27.3.5 Ethernet MAC MII address register (EMAC_MACMIIADDR)

The Ethernet MAC MII address register controls the external PHY through the management interface.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 11	PA	0x00	rw	PHY Address This field indicates which of the 32 possible PHY devices are being accessed.
Bit 10: 6	MII	0x00	rw	MII Register This field select the desired MII register in the PHY device.
Bit 5: 2	CR	0x0	rw	Clock Range The CSR clock range selection determines the MDC clock frequency based on the used CSR clock frequency. Each value (when bit 5=0) has its corresponding CSR clock frequency range in order to ensure that the MDC clock frequency is roughly between 1.0 MHz and 2.5 MHz. 0000: CSR clock frequency is 60–100 MHz, and MDC clock frequency is CSR clock/42 0001: CSR clock frequency is 100–150 MHz, and MDC clock frequency is CSR clock/62 0010: CSR clock frequency is 20–35 MHz, and MDC clock frequency is CSR clock/16 0011: CSR clock frequency is 35–60 MHz, and MDC clock frequency is CSR clock/26 0100: CSR clock frequency is 150–250 MHz, and MDC clock frequency is CSR clock/102 0101: CSR clock frequency is 250–288 MHz, and MDC clock frequency is CSR clock/124 0110, 0111: Reserved
Bit 1	MW	0x0	rw	MII Write When this bit is set, it indicates that the EMAC_MACMIIDT register is used for a write operation to the PHY. When this bit is not set, it is a read operation, and the data is loaded to the EMAC_MACMIIDT register.
Bit 0	MB	0x0	rw	MII Busy This bit should read a logic 0 before writing to the EMAC_MACMIIADDR and EMAC_MACMIIDT register. During a PHY register access, this bit is set to 1'b1 by software, indicating that a read or write access is in progress. The EMAC_MACMIIDT register is invalid before this bit is cleared by the MAC. Thus, the MII data should be kept valid until this bit is cleared by the MAC during a PHY write operation. Similarly, the EMAC_MACMIIDT value is invalid until this bit is cleared by the MAC during a PHY read operation. The previous operation must be completed before performing subsequent read or write operations. This is because that there will be no acknowledgement from PHY to MAC after the completion of a read or write operation, the function of this bit will not change even if the PHY is not present.

27.3.6 Ethernet MAC MII data register (EMAC_MACMIIDT)

The Ethernet MAC MII data register stores data to be written to the PHY register located at the address specified in the EMAC_MACMIIADDR register. EMAC_MACMIIDT register also stores data read out from the PHY registers.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 0	MD	0x0000	rw	MII Data This field contains the 16-bit value from the PHY after a read operation, or the 16-bit value to be written to the PHY before a write operation.

27.3.7 Ethernet MAC flow control register (EMAC_MACFCTRL)

The Ethernet MAC flow control register controls the generation and reception of the control frames by the MAC flow control block. Writing 1 to the Busy bit triggers the flow control block to generate a Pause frame. The field of the control frame is selected as defined in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control frame. The Busy bit remains set before the control frame is transferred onto the cable. The host must make sure that the Busy bit is cleared before writing to the register.

Bit	Name	Reset value	Type	Description
Bit 31: 16	PT	0x0000	rw	Pause Time This field contains the value to be used in the Pause Time field of the control frame. If the Pause Time bit is configured to be double-synchronized to the MII clock domain, then consecutive write operations to this register should be performed only after at least four clock cycles in the destination clock domain.
Bit 15: 8	Reserved	0x00	resd	Kept at its default value.
Bit 7	DZQP	0x0	rw	Disable Zero-Quanta Pause When this bit is set, it disables the automatic generation of Zero-quanta Pause frame while the flow control signal of the FIFO layer is disabled. When this bit is cleared, normal operation resumes. The automatic generation of Zero-quanta Pause frame is enabled.
Bit 6	Reserved	0x0	resd	Kept at its default value.
Bit 5: 4	PLT	0x0	rw	Pause Low Threshold This field defines the threshold of the Pause timer. The threshold values should always be less than the Pause time defined in the [31: 16] bit. For example, if PT = 100H (256 slot times), and PLT = 01, then a second Pause frame is automatically transmitted if initiated at 228 (256-28) slot times after the first Pause frame is transmitted. Threshold selection as follows: 00: Pause time minus 4 slot times (PT minus 4 slot times) 01: Pause time minus 28 slot times (PT minus 28 slot times) 10: Pause time minus 144 slot times (PT minus 144 slot times) 11: Pause time minus 256 slot times (PT minus 256 slot times) Slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface.
Bit 3	DUP	0x0	rw	Detect Unicast Pause Frame The Pause frame with a unique multicast address as specified in the IEEE 802.3 will be processed. When this bit is set, the MAC detects the Pause frames with a unicast address specified in the MAC address0 high and MAC address0 low registers. When this bit is cleared, the MAC detects only a Pause frame with a unique multicast address. Note: If the multicast address of the received frame does not match the unique multicast address, the MAC will not process the Pause frame.
Bit 2	ERF	0x0	rw	Enable Receive Flow control When this bit is set, the MAC decodes the received Pause frame and disables the transmitter for a period of time. When this bit is cleared, the decode function of the Pause frame is disabled.
Bit 1	ETF	0x0	rw	Enable Transmit Flow control In full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is cleared, the flow control of the MAC is disabled,

				and the MAC does not transmit any Pause frames. In half-duplex mode, when this bit is set, the MAC enables the back-pressure feature. When this bit is cleared, the back-pressure feature is disabled.
Bit 0	FCB/BPA	0x0	rw1c/rw	Flow Control Busy/Back Pressure Activate In full-duplex mode, this bit initiates a Pause frame; in half-duplex mode, the back-pressure feature is activated if the TFE bit is set. In full-duplex mode, this bit is read as 1'b0 before writing to the EMAC_MACFCTRL register. The application must set this bit to 1'b1 to initiate a Pause frame. During a control frame transmission, this bit remains set, indicating that a frame transmission is in progress. After the completion of the Pause frame, the MAC resets this bit to 1'b0. The Ethernet MAC flow control register (EMAC_MACFCTRL) should not be written until this bit is cleared. In half-duplex mode, when this bit is set (and the TFE is set), the back-pressure feature is activated by the MAC. During back pressure, when the MAC receives a new frame, the transmitter starts sending a JAM mode, resulting a collision. When the MAC is configured to full-duplex mode, the back-pressure (BPA) function is automatically disabled.

27.3.8 Ethernet MAC VLAN tag register (EMAC_MACVLT)

The Ethernet MAC VLAN tag register contains the IEEE 802.1Q VLAN tag to identify the VLAN frames. The MAC compares the 13th and 14th bytes of the received frame (length/type) with 16'h8100, and the following 2 bytes are compared with the VLAN tag. If the comparison matches, the VLAN bit is set in the receive frame status. The legal length of the VLAN frame is increased from 1518 bytes to 1522 bytes.

If the EMAC_MACVLT register is configured to be double-synchronized to the (G)MII clock domain, then consecutive write operations to this register should be performed at least four clock cycles in the destination clock domain.

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	ETV	0x0	ro	Enable 12-bit VLAN tag comparison When this bit is set, a 12-bit VLAN identifier, rather than a 16-bit VLAN tag, is used for comparison and filtering. The bit [11: 0] of the VLAN tag is compared with the corresponding filed in the received VLAN-tagged frame. Similarly, if enabled, only a 12-bit VLAN tag is used for hash VLAN filtering. When this bit is cleared, the 16 bits of the received VLAN frame's 15th and 16th bytes are used for comparison and VLAN hash filtering.
Bit 15: 0	VTI	0x0000	rw	VLAN Tag Identifier (for receive frames) This field contains the 802.1Q VLAN tag to identify VLAN frames, which is compared with the 15th and 16th bytes of the received VLAN frames, described as follows: Bit [15: 13]: User priority Bit 12: Canonical format indicator (CFI) or drop eligible indicator (DEI) Bit [11: 0]: VLAN tag's VLAN identifier field When the ETV bit is set, only the VID ([11: 0]) is used for comparison. If the VL is all zero (if the ETV is set, then VL[11: 0] is all zero), the MAC does not check the 15th and 16th bytes for VLAN tag comparison, and treats all frames with a type field value of 0x8100 or 0x88a8 as VLAN frames.

27.3.9 Ethernet MAC remote wakeup frame filter register (EMAC_MACRWFF)

The PMT CSR sets the request wakeup events and detects the wakeup events.

Figure 27-18 Ethernet MAC remote wakeup frame filter register (EMAC_MACRWFF)

Wkuppktfilter_reg0	Filter 0 Byte Mask							
Wkuppktfilter_reg1	Filter 1 Byte Mask							
Wkuppktfilter_reg2	Filter 2 Byte Mask							
Wkuppktfilter_reg3	Filter 3 Byte Mask							
Wkuppktfilter_reg4	RESD	Filter 3 Cmd	RESD	Filter 2 Cmd	RESD	Filter 1 Cmd	RESD	Filter 0 Cmd
Wkuppktfilter_reg5	Filter 3 Offset		Filter 2 Offset		Filter 1 Offset		Filter 0 Offset	
Wkuppktfilter_reg6	Filter 1 CRC-16				Filter 0 CRC-16			
Wkuppktfilter_reg7	Filter 3 CRC-16				Filter 2 CRC-16			

27.3.10 Ethernet MAC PMT control and status register (EMAC_MACPMTCTRLSTS)

The Ethernet MAC PMT control and status register sets the request wakeup events and detects the wakeup events.

Bit	Name	Reset value	Type	Description
Bit 31	RWFFPR	0x0	rw1s	Remote Wakeup Frame Filter Register Pointer Reset When this bit is set, it resets the remote frame filter register pointer to 3'b000. This bit is automatically cleared after one clock cycle.
Bit 30: 10	Reserved	0x000000	resd	Kept at its default value.
Bit 9	GUC	0x0	rw	Global UniCast When this bit is set, it enables all unicast packets filtered by the MAC address filtering to be remote wakeup frames.
Bit 8: 7	Reserved	0x0	resd	Kept at its default value.
Bit 6	RRWF	0x0	rrc	Received Remote Wakeup Frame When this bit is set, it indicates that the power management event was generated because of the reception of a remote wakeup frame. This bit is cleared by a read access to this register.
Bit 5	RMP	0x0	rrc	Received Magic Packet When this bit is set, it indicates that the power management event is generated because of the reception of a Magic packet. This bit is cleared by a read access to this register.
Bit 4: 3	Reserved	0x0	resd	Kept at its default value.
Bit 2	ERWF	0x0	rw	Enable Remote Wakeup Frame When this bit is set, it indicates that the power management event is generated due to a remote wakeup frame reception.
Bit 1	EMP	0x0	rw	Enable Magic Packet When this bit is set, it indicates that the power management event is generated due to a Magic packet reception.
Bit 0	PD	0x0	rw1s	Power Down When this bit is set, the MAC receiver will drop all received frames after receiving the expected Magic packet or a remote wakeup frame. Then this bit is automatically cleared and power-down mode is disabled. This bit can also be cleared by software before the expected Magic packet or a remote wakeup frame is received. After this bit is cleared, the MAC forwards the receive frames to the application. This bit must only be set when either the Magic Packet enable bit, global unicast bit or the remote wakeup frame enable bit is set high.

27.3.11 Ethernet MAC interrupt status register (EMAC_MACISTS)

The Ethernet MAC interrupt status register identify the events in the MAC that can generate an interrupt.

Bit	Name	Reset value	Type	Description
Bit 15: 10	Reserved	0x00	resd	Kept at its default value.
Bit 9	TIS	0x0	rrc	Timestamp Interrupt Status When this bit is set, it indicates that the system time value equals or exceeds the value programmed in the destination time registers. This bit is cleared after the completion of a read operation to this bit.
Bit 8: 7	Reserved	0x0	resd	Kept at its default value.
Bit 6	MTIS	0x0	ro	MMC Transmit Interrupt Status This bit is set when an interrupt event is generated in the EMAC_MMCTI register. This bit is cleared when all bits in the transmit interrupt register are cleared.
Bit 5	MRIS	0x0	ro	MMC Receive Interrupt Status This bit is set when an interrupt is generated in the

				EMAC_MMCRI register. This bit is cleared when all bits in the receive interrupt register are cleared.
Bit 4	MIS	0x0	ro	MMC Interrupt Status This bit is set whenever any bit of the [7: 5] bit is set high. This bit is cleared only when these bits are set low.
Bit 3	PIS	0x0	ro	PMT Interrupt Status This bit is set when a Magic packet or a remote wakeup event is received in power-down mode (see bits 5 and 6 in the EMAC_MACPMTCTRLSTS register). This bit is cleared when both bits [6: 5] are cleared due to a read access to the EMAC_MACPMTCTRLSTS register.
Bit 2: 0	Reserved	0x0	resd	Kept at its default value.

27.3.12 Ethernet MAC interrupt mask register (EMAC_MAIMR)

The Ethernet MAC interrupt mask register is used to mask the interrupt signal generated due to the corresponding event in the EMAC_MACISTS register.

Bit	Name	Reset value	Type	Description
Bit 15: 10	Reserved	0x00	resd	Kept at its default value.
Bit 9	TIM	0x0	rw	Timestamp Interrupt Mask When this bit is set, it masks the interrupt signal generated in the time stamp interrupt status bit of the EMAC_MACISTS register. This bit is applicable only when the IEEE1588 time stamp is enabled. This bit is reserved in other modes.
Bit 8: 4	Reserved	0x00	resd	Kept at its default value.
Bit 3	PIM	0x0	rw	PMT Interrupt Mask. When this bit is set, it masks the interrupt signal generated in the MPT interrupt status bit of the EMAC_MACISTS register.
Bit 2: 0	Reserved	0x0	resd	Kept at its default value.

27.3.13 Ethernet MAC address 0 high register (EMAC_MACA0H)

The EMAC_MACA0H register contains the upper 6 bits of the first 6-byte MAC address of the station. The first DA byte received on the MII interface corresponds to the LS byte (bit [7: 0]) of the MAC address low register. For example, if the 0x112233445566 (0x11 in channel 0 of the first column) is received on the MII interface as the destination address, then the MacAddress0 register [47: 0] is compared with 0x665544332211.

If the MAC address register is configured to be double-synchronized with the MII domain, the synchronization can be enabled only by writing the bit [31: 24] (in little endian mode) or the bit [7: 0] (in big-endian mode) in the Ethernet MAC address 0 low register (EMAC_MACA0L). Consecutive write operations to this address low register must be performed after at least 4 cycles in the destination clock domain so as to achieve an accurate synchronous update.

Bit	Name	Reset value	Type	Description
Bit 31	AE	0x0	rrc	Adrress Always 1.
Bit 30: 16	Reserved	0x0010	resd	Kept at its default value.
Bit 15: 0	MA0H	0xFFFF	rw	MAC Address0 [47: 32] This field contains the upper 16 bits of the first 6-byte MCU address. This is used by the MAC for filtering received frames, and for inserting the MAC address in the transmit flow control frames (Pause).

27.3.14 Ethernet MAC address 0 low register (EMAC_MACA0L)

The Ethernet MAC address 0 low register contains the lower 32 bits of the 6-byte first MAC address.

Bit	Name	Reset value	Type	Description
Bit 31: 0	MA0L	0xFFFF FFFF	rw	MAC Address0 [31: 0] This field contains the lower 16 bits of the first 6-byte MCU address. This is used by the MAC for filtering received frames, and for inserting the MAC address in the transmit flow control frames (Pause).

27.3.15 Ethernet MAC address 1 high register (EMAC_MACA1H)

The Ethernet MAC address 1 high register holds the upper 16 bits of the 6-byte second MAC address.

If the MAC address register is configured to be double-synchronized with the MII domain, the synchronization can be enabled only by writing the bit [31: 24] (in little endian mode) or the bit [7: 0] (in big-endian mode) in the Ethernet MAC address 1 low register (EMAC_MACA1L) . Consecutive write operations to this address low register must be performed after at least 4 cycles in the destination clock domain so as to achieve an accurate synchronous update.

Bit	Name	Reset value	Type	Description
Bit 31	AE	0x0	rw	Address Enable When this bit is set, the address filter uses the second MAC address for a perfect filtering. When this bit is cleared, the address filter will ignore the address for filtering.
Bit 30	SA	0x0	rw	Source Address When this bit is set, the MAC address 1 [47: 0] is used for comparison with the source address field of the received frame. When this bit is cleared, the MAC address 1 [47: 0] is used for comparison with the destination address field of the received frame.
Bit 29: 24	MBC	0x00	rw	Mask Byte Control These bits are mask control bits for comparison with each of the MAC address bytes. When this bit is set, the MAC does not compare the corresponding byte of the received DA/SA with the contents of the MAC address 1 register. Each control bit is used for controlling the mask of the bytes as follows: Bit 29: EMAC_MACA1H [15: 8] Bit 28: EMAC_MACA1H [7: 0] Bit 27: EMAC_MACA1L[31: 24] ... Bit 24: EMAC_MACA1L[7: 0] It is possible to filter group addresses (that is, group address filtering) by masking one or more bytes of the address.
Bit 23: 16	Reserved	0x00	resd	Kept at its default value.
Bit 15: 0	MA1H	0xFFFF	rw	MAC Address1 [47: 32] These bits contain the upper 16 bits (47: 32) of the 6-byte second MAC address.

27.3.16 Ethernet MAC address 1 low register (EMAC_MACA1H)

The Ethernet MAC address 1 low register contains the lower 32 bits of the 6-byte second MAC address.

Bit	Name	Reset value	Type	Description
Bit 31: 0	MA1L	0xFFFF FFFF	rw	MAC Address1 [31: 0] These bits contain the lower 32 bits of the 6-byte second MAC address. The contents of this field is undefined until loaded by the application after the initialization process.

27.3.17 Ethernet MAC address 2 high register (EMAC_MACA2H)

The Ethernet MAC address 2 high register holds the upper 16 bits of the 6-byte second MAC address.

If the MAC address register is configured to be double-synchronized with the MII domain, the synchronization can be enabled only by writing the bit [31: 24] (in little endian mode) or the bit [7: 0] (in big-endian mode) in the Ethernet MAC address 2 low register (EMAC_MACA2L). Consecutive write operations to this address low register must be performed after at least 4 cycles in the destination clock domain so as to achieve an accurate synchronous update.

Bit	Name	Reset value	Type	Description
Bit 31	AE	0x0	rw	Address Enable When this bit is set, the address filter uses the second MAC address for a perfect filtering. When this bit is cleared, the address filter will ignore the address for filtering.
Bit 30	SA	0x0	rw	Source Address When this bit is set, the MAC address2 [47: 0] is used for comparison with the source address field of the received frame. When this bit is cleared, the MAC address 2 [47: 0] is used for comparison with the destination address field of the received frame.
Bit 29: 24	MBC	0x00	rw	Mask Byte Control These bits are mask control bits for comparison with each of the MAC address bytes. When this bit is set, the MAC does not compare the corresponding byte of the received DA/SA with the contents of the MAC address 2 register. Each control bit is used for controlling the mask of the bytes as follows: Bit 29: EMAC_MACA2H [15: 8] Bit 28: EMAC_MACA2H [7: 0] Bit 27: EMAC_MACA2L[31: 24] ... Bit 24: EMAC_MACA2L[7: 0] It is possible to filter group addresses (that is, group address filtering) by masking one or more bytes of the address.
Bit 23: 16	Reserved	0x00	resd	Kept at its default value.
Bit 15: 0	MA2H	0xFFFF	rw	MAC Address2 High [47: 32] These bits contain the upper 16 bits (47: 32) of the 6-byte second MAC address.

27.3.18 Ethernet MAC address 2 low register (EMAC_MACA2L)

The Ethernet MAC address 2 low register holds the lower 32 bits of the 6-byte second MAC address.

Bit	Name	Reset value	Type	Description
Bit 31: 0	MA2L	0xFFFF FFFF	rw	MAC Address2 Low [31: 0] These bits contain the lower 32 bits of the 6-byte second MAC address. The contents of this field is undefined until loaded by the application after the initialization process.

27.3.19 Ethernet MAC address 3 high register (EMAC_MACA3H)

The Ethernet MAC address 3 high register holds the upper 16 bits of the 6-byte second MAC address.

If the MAC address register is configured to be double-synchronized with the MII domain, the synchronization can be enabled only by writing the bit [31: 24] (in little endian mode) or the bit [7: 0] (in big-endian mode) in the Ethernet MAC address 3 low register (EMAC_MACA3L). Consecutive write operations to this address low register must be performed after at least 4 cycles in the destination clock domain so as to achieve an accurate synchronous update.

Bit	Name	Reset value	Type	Description
Bit 31	AE	0x0	rw	Address Enable When this bit is set, the address filter uses the second MAC address for a perfect filtering.

				When this bit is cleared, the address filter will ignore the address for filtering.
Bit 30	SA	0x0	rw	Source Address When this bit is set, the MAC address 3 [47: 0] is used for comparison with the source address field of the received frame. When this bit is cleared, the MAC address 3 [47: 0] is used for comparison with the destination address field of the received frame.
Bit 29: 24	MBC	0x00	rw	Mask Byte Control These bits are mask control bits for comparison with each of the MAC address bytes. When this bit is set, the MAC does not compare the corresponding byte of the received DA/SA with the contents of the MAC address 3 register. Each control bit is used for controlling the mask of the bytes as follows: Bit 29: EMAC_MACA3H [15: 8] Bit 28: EMAC_MACA3H [7: 0] Bit 27: EMAC_MACA3L[31: 24] ... Bit 24: EMAC_MACA3L[7: 0] It is possible to filter group addresses (that is, group address filtering) by masking one or more bytes of the address.
Bit 23: 16	Reserved	0x00	resd	Kept at its default value.
Bit 15: 0	MA3H	0xFFFF	rw	MAC Address3 High [47: 32] These bits contain the lower 16 bits (47: 32) of the 6-byte second MAC address.

27.3.20 Ethernet MAC address 3 low register (EMAC_MACA3L)

The Ethernet MAC address 3 low register holds the lower 32 bits of the 6-byte second MAC address.

Bit	Name	Reset value	Type	Description
Bit 31: 0	MA3L	0xFFFF FFFF	rw	MAC Address3 Low [31: 0] These bits contain the lower 32 bits of the 6-byte second MAC address. The contents of this field is undefined until loaded by the application after the initialization process.

27.3.21 Ethernet DMA bus mode register (EMAC_DMABM)

The Ethernet DMA bus mode register defines the bus operation modes for the DMA.

Bit	Name	Reset value	Type	Description
Bit 31: 26	Reserved	0x00	resd	Kept at its default value.
Bit 25	AAB	0x0	rw	Address-Aligned Beats When this bit is set and the FB bit equals 1, the AHB interface generates burst transfers aligned to the start address LS bits. If the FB bit equals 0, the first burst transfer (accessing the data buffer's start address) is not aligned, but subsequent burst transfers are aligned to the address. This bit is applicable to GMAC-AHB and GMAC-AXI configurations only. It is reserved in other configurations.
Bit 24	PBLx8	0x0	rw	PBLx8 Mode When this bit is set, this bit multiples the PBL value programmed (bits [22: 17] and bits [13: 8]) by 8. Thus the DMA transfers data at 8, 16, 32, 64, 128 and 256 beats depending on the PBL value.
Bit 23	USP	0x0	rw	Use separate PBL When this bit is set, the Rx DMA uses the value programmed in bit [22: 17] as PBL. The PBL value in bit [13: 8] is applicable to Tx DMA operations only. When this bit is cleared, the PBL value in bit [13: 8] is applicable to both Tx DMA and Rx DMA operations.
Bit 22: 17	RDP	0x01	rw	Rx DMA PBL

				This field indicates the maximum number of beats to be transferred in one Rx DMA operation. This is the maximum value that is used for a single write or read operation. The Rx DMA always attempts to perform burst transfer as specified in RPBL each time it starts a burst transfer on the host bus. The RPBL can be programmed with 1, 2, 4, 8, 16 and 32. Any other value result in unexpected behavior. These bits are applicable only when the USP bit is set.
Bit 16	FB	0x0	rw	Fixed Burst This bit controls whether the AHB master interface performs fixed burst transfers or not. When this bit is set, the AHB uses only SINGLE, INCR4, INCR8 or INCR16 during start of normal burst transfers. When this bit is cleared, the AHB or AXI interface uses SINGLE and INCR burst transfer operations.
Bit 15: 14	PR	0x0	rw	Priority Ratio These bits control the priority ratio of the round-robin arbitration between Rx DMA and Tx DMA. These bits are valid only when the bit 1 (destination address) is reset. The priority ratio is either Rx: Tx or Tx: Rx, depending on whether the bit 27 (TXPR) is set or reset. 00: 1: 1 01: 2: 1 10: 3: 1 11: 4: 1
Bit 13: 8	PBL	0x01	rw	Programmable Burst Length These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum that is used for a single write or read operation. The DMA always attempts to perform burst transfer as specified in PBL each time it starts a burst transfer on the host bus. The RPBL can be programmed with 1, 2, 4, 8, 16 and 32. Any other value result in unexpected behavior. When the USP is set, the PBL value is applicable to Tx DMA operations only. If the number of beats to be transferred is greater than 32, the following steps are required: 1. Set PBLx8 mode 2. Set PBL For example, if the maximum value to be transferred is greater than 64, then the PBLx8 should be set first, and then the PBL is set to 8. The PBL values have the following limitations: The maximum number of beats possible is limited by the size of the Tx FIFO and Rx FIFO on the MTL layer, as well as the data bus width on the DMA. FIFO constraint: The maximum beat supported by the FIFO is half the depth of the FIFO, unless otherwise specified.
Bit 7	EDE	0x0	rw	Enhanced descriptor enable When this bit is set to 1, the enhanced descriptor format is enabled and the descriptor size is increased to 8 words. For details, refer to TX enhanced descriptors and RX enhanced descriptors.
Bit 6: 2	DSL	0x00	rw	Descriptor Skip Length These bits define the number of words to skip between two unchained descriptors. The address skip starts from the end of the current descriptor to the start of next descriptor. When the DSL value equals 0, the descriptor is regarded as contiguous by the DMA in ring mode.
Bit 1	DA	0x0	rw	DMA Arbitration These bits specify the arbitration scheme between the transmit path and receive path of channel 0. 0: Rx: Tx or Tx: Rx The priority between round-robin channels depends on the

				priority as specified in the bit [15: 14] (PR) and the priority weight as specified in bit 27(TXPR). 1: Fixed priority When the bit 27 (TXPR) is set, Tx has priority over Rx. Otherwise, Rx has priority over Tx.
Bit 0	SWR	0x1	rw	Software Reset When this bit is set, the MAC DMA controller resets all internal registers and MAC logic. This bit is automatically cleared after all reset operations have been completed.

27.3.22 Ethernet DMA transmit poll demand register (EMAC_DMATPD)

The EMAC_DMATPD register enables the Tx DMA to check whether or not the current descriptor is owned by the DMA. The Transmit Poll Demand is used to wake up the Tx DMA from suspend mode. The Tx DMA can go into suspend mode due to an underflow error in a transmitted frame or due to the unavailability of descriptors owned by transmit DMA. The Poll demand can be issued at any time, and the Tx DMA will reset this command once it starts re-fetching the current descriptor from the host memory. This register is always read 0.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TPD	0x0000 0000	rrc	Transmit Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by the EMAC_DMACTD. If the descriptor is not available (owned by host), the transmission suspends, and the bit 2 (TU) is set in the status register. If the descriptor is available, the transmission resumes.

27.3.23 Ethernet DMA receive poll demand register (EMAC_DMARPD)

The EMAC_DMARPD register enables the Rx DMA to check new descriptors. The Receive Poll Demand is used to wake up the Rx DMA from suspend mode. The Rx DMA can enter suspend mode due to the unavailability of descriptors owned by it.

Bit	Name	Reset value	Type	Description
Bit 31: 0	RPD	0x0000 0000	rrc	Receive Poll Demand When these bits are written with any value, the DMA reads the current descriptor pointed to by the EMAC_DMACRD. If the descriptor is not available (owned by host), the reception suspends, and the bit 7 (RU) is set in the status register. If the descriptor is available, the reception resumes.

27.3.24 Ethernet DMA receive descriptor list address register (EMAC_DMARDLADDR)

The EMAC_DMARDLADDR register points to the start of the receive descriptor list. The descriptor list is located in the host's physical memory and must be word-aligned. The DMA enables bus-width aligned address by making the corresponding LS bit low. Writing to the register is permitted only when the Rx DMA stops. After the Rx DMA stops, this register must be written before the receive start command is given.

Writing to the register is permitted only when the Rx DMA stops. In other words, the bit 1 (SR) is set 0 in the operation mode register. After the Rx DMA stops, this register can be written with a new descriptor list address.

When the SR bit is set, the DMA uses the newly programmed descriptor base address.

If the SR is cleared and this register remains unchanged, then the DMA will use the previous descriptor address when the Rx DMA stops.

Bit	Name	Reset value	Type	Description
Bit 31: 0	SRL	0x0000 0000	rw	Start of Receive List

				These bits contain the base address of the first descriptor in the receive descriptor list. The LSB bits (1: 0, 2: 0 or 3: 0) for 32/64/128-bit bus width are ignored and taken as zero by the DMA. Therefore these LSB bits are read only.

27.3.25 Ethernet DMA transmit descriptor list address register (EMAC_DMATDLADDR)

The EMAC_DMATDLADDR register points to the start of the transmit descriptor list. The descriptor list is located in the host's physical memory and must be word-aligned. The DMA enables bus-width aligned address by making the corresponding LS bit low.

Writing to the register is permitted only when the Tx DMA stops. In other words, the bit 13 (ST) is set 0 in the register 6 (operation mode register). After the Tx DMA stops, this register can be written with a new descriptor list address.

When the SR bit is set, the DMA uses the newly programmed descriptor base address.

If the SR is cleared and this register remains unchanged, then the DMA will use the previous descriptor address when the Tx DMA stops.

Bit	Name	Reset value	Type	Description
Bit 31: 0	STL	0x0000 0000	rw	Start of Transmit List These bits contain the base address of the first descriptor in the transmit descriptor list. The LSB bits (1: 0, 2: 0 or 3: 0) for 32/64/128-bit bus width are ignored and taken as zero by the DMA. Therefore these LSB bits are read only.

27.3.26 Ethernet DMA status register (EMAC_DMASTS)

The EMAC_DMASTS register contains all the status bits the DMA reports to the host. This register is read by the software driver during an interrupt service routine or polling. Most of the bits in this register can trigger the host to be interrupted. The bits in this register cannot be cleared when read. Writing 1'b1 to the bit [16: 0] (unreserved) in this register clears them. Writing 1'b0 has no effect. Each bit (bit [16: 0]) can be masked through the corresponding bit in the interrupt enable mask register.

Bit	Name	Reset value	Type	Description
Bit 31: 30	Reserved	0x0	resd	Kept at its default value.
Bit 29	TTI	0x0	ro	Timestamp Trigger Interrupt This bit indicates an interrupt event in the time stamp generator block. The software must read the corresponding register to get interrupt sources. This bit is applicable only when the IEEE1588 time stamp feature is enabled. Otherwise, this bit is reserved.
Bit 28	MPI	0x0	ro	MAC PMT Interrupt This bit indicates an interrupt even in the PMT. The software must read the Ethernet PMT control and status register (EMAC_MACPMTCTRLSTS) to get the interrupt sources and clear them in order to reset this bit to 1'b0. This bit is applicable only when the PMT function is enabled. Otherwise, this bit is reserved.
Bit 27	MMI	0x0	ro	MAC MMC Interrupt This bit indicates an interrupt event in the MMC. The software must read the corresponding register to get interrupt sources and clear them in order to reset this bit to 1'b0. This bit is applicable only when the MAC MMC is enabled. Otherwise, this bit is reserved.
Bit 26	Reserved	0x0	resd	Kept at its default value.
Bit 25: 23	EB	0x0	ro	Error Bits These bits indicate the type of error that caused a bus error. They are applicable only when the bit 13 (FBI) is set. This filed does not generate an interrupt. 000: Error during data transfer by Rx DMA 011: Error during read transfer by Tx DMA 100: Error during Rx DMA descriptor write access

				101: Error during Tx DMA descriptor write access 110: Error during Rx DMA descriptor read access 111: Error during Tx DMA descriptor read access Note: 001 and 010 are reserved.
Bit 22: 20	TS	0x0	ro	Transmit Process State This field indicates the Tx DMA FSM state. This field does not generate an interrupt. 3'b000: Stopped; Rest or Stop transmit command issued 3'b001: Running; Fetching transmit descriptor 3'b010: Running; Waiting for status 3'b011: Running; Reading data from host memory buffer and queuing it to Tx FIFO 3'b100: Time stamp write status 3'b101: Reserved for future use 3'b110: Suspended; Transmit descriptor unavailable or transmit buffer underflow 3'b111: Running; Closing transmit descriptor
Bit 19: 17	RS	0x0	ro	Receive Process State This field indicates the Rx DMA FSM state. This field does not generate an interrupt. 3'b000: Stopped; Rest or Stop transmit command issued 3'b001: Running; Fetching receive descriptor 3'b010: Reserved for future use 3'b011: Running; Waiting for receive packet 3'b100: Suspended; Receive descriptor unavailable 3'b101: Running; Closing receive descriptor 3'b110: Time stamp write status 3'b111: Running; Transferring the receive buffer data to host memory
Bit 16	NIS	0x0	rw1c	Normal Interrupt Summary The normal interrupt summary value is the logic OR of the following bits when the corresponding interrupt bits are enabled in the interrupt enable registers. EMAC_DMASTS[0]: Transmit interrupt EMAC_DMASTS[2]: Transmit buffer unavailable EMAC_DMASTS[6]: Receive interrupt EMAC_DMASTS[14]: Early receive interrupt Only unmasked bits affect the normal interrupt summary. This is a sticky bit and it must be cleared (by writing 1 to this bit) each time a corresponding bit (causes NIS to be set) is cleared.
Bit 15	AIS	0x0	rw1c	Abnormal Interrupt Summary The abnormal interrupt summary value is the logic OR of the following bits when the corresponding interrupt bits are enabled in the interrupt enable registers. EMAC_DMASTS[1]: Transmit process stopped EMAC_DMASTS[3]: Transmit Jabber timeout EMAC_DMASTS[4]: Receive FIFO overflow EMAC_DMASTS[5]: Transmit data underflow EMAC_DMASTS[7]: Receive buffer unavailable EMAC_DMASTS[8]: Receive process stopped EMAC_DMASTS[9]: Receive watchdog timeout EMAC_DMASTS[10]: Early transmit interrupt EMAC_DMASTS[13]: Fatal bus error Only unmasked bits affect the abnormal interrupt summary. This is a sticky bit and it must be cleared (by writing 1 to this bit) each time a corresponding bit (causes AIS to be set) is cleared.
Bit 14	ERI	0x0	rw1c	Early Receive Interrupt This bit indicates that the DMA has filled the first data buffer of the packet. This bit is cleared when the software writes 1 to this bit or when the bit 6 (RI) bit is set in this register. (Whichever occurs first)
Bit 13	FBEI	0x0	rw1c	Fatal Bus Error Interrupt

				This bit indicates that a bus error occurred as defined in bit [25: 23]. When this bit is set, the corresponding DMA will disable all its bus accesses.
Bit 12: 11	Reserved	0x0	resd	Kept at its default value.
Bit 10	ETI	0x0	rw1c	Early Transmit Interrupt This bit indicates that the frame to be transmitted was fully sent to the MTL Tx FIFO.
Bit 9	RWT	0x0	rw1c	Receive Watchdog Timeout When this bit is set, it indicates that the receive watchdog timer timeout occurs while receiving the current frame, and the current frame is cut off after the watchdog timeout happens.
Bit 8	RPS	0x0	rw1c	Receive Process Stopped This bit is set when the receive process enters the stop state.
Bit 7	RBU	0x0	rw1c	Receive Buffer Unavailable This bit indicates that the next descriptor in the receive list is owned by the host and cannot be acquired by the DMA. Thus the receive process is suspended. The host should change the ownership of the descriptor and release the receive poll demand command in order to resume receive process. If no receive poll demand command is issued, the receive process resumes when the DMA receives the next incoming frame. This bit is set only when the previous receive descriptor is owned by the DMA.
Bit 6	RI	0x0	rw1c	Receive Interrupt This bit indicates the completion of a frame reception. After the completion of a frame reception, the bit 31 of the RDES1 (interrupt disabled after reset operation) is reset in the last descriptor. Specific frame status information will be posted in the descriptor. Receive process remains in the running state.
Bit 5	UNF	0x0	rw1c	Transmit Underflow This bit indicates that the transmit buffer has an underflow during a frame transmission. Transmit process is suspended and the underflow error bit TDES0[1] is set.
Bit 4	OVF	0x0	rw1c	Receive Overflow This bit indicates that the receive buffer has an overflow during a frame reception. If the partial frame has been transferred to the application, the overflow status is set in the RDES0[11].
Bit 3	TJT	0x0	rw1c	Transmit Jabber Timeout This bit indicates that the transmit Jabber timer will expire when the current frame is greater than 2048 bytes (it is 10240 bytes if Jumbo frame is enabled). After the Jabber is expired, the transmit process is aborted and enters stop state, which causes the transmit Jabber timeout flag bit TDES0[14] to be set.
Bit 2	TBU	0x0	rw1c	Transmit Buffer Unavailable This bit indicates that the next descriptor in the transmit list is owned by the host and cannot be acquired by the DMA. Then the transmit process is suspended. Bit [22: 20] explains the transmit process state. To resume transmit process, the host should change the ownership of the descriptor by setting the TDES0[31] and issue the transmit poll demand command
Bit 1	TPS	0x0	rw1c	Transmit Process Stopped This bit is set when the transmit process stops.
Bit 0	TI	0x0	rw1c	Transmit Interrupt This bit indicates the completion of a frame transmission. The bit 31 (OWN) is reset in the TDES0. Specific frame status information will be posted in the descriptor.

27.3.27 Ethernet DMA operation mode register (EMAC_DMAOPM)

The EMAC_DMAOPM register defines the receive and transmit operation modes and commands. This register should be the last CSR to be written during DMA initialization. This register is also applicable to GMAC-MTL configuration where the unused and reserved bits are 24, 13, 2 and 1. A delay value greater than 4us is required between two consecutive write accesses to this register.

Bit	Name	Reset value	Type	Description
Bit 31: 27	Reserved	0x00	resd	Kept at its default value.
Bit 26	DT	0x0	rw	Disable Dropping of TCP/IP Checksum Error Frames When this bit is set, the MAC does not drop the frames that only have errors detected by the receive checksum offload engine. Such frames have errors in the encapsulated payload only but do not have errors (including FCS error) in the Ethernet frames received by the MAC. When this bit is cleared, all error frames are dropped if the FEF bit is reset.
Bit 25	RSF	0x0	rw	Receive Store and Forward When this bit is set, the MTL reads the Rx FIFO only after a full frame is written to the Rx FIFO, ignoring the RTC bit. When this bit is cleared, the Rx FIFO operates in cut-through mode and will be subject to the threshold defined by the RTC.
Bit 24	DFRF	0x0	rw	Disable Flushing of Received Frames When this bit is set, the Rx DMA does not flush any receive frame due to the unavailability of receive descriptors or receive buffers. When this bit is cleared, the DMA will flush receive frames in case of the above-mentioned circumstances.
Bit 23: 22	Reserved	0x000	resd	Kept at its default value.
Bit 21	TSF	0x0	rw	Transmit Store and Forward When this bit is set, transmission starts when a full frame resides in the Tx FIFO, and the TTC values specified in the bit [16: 14] are ignored. This bit can be changed only when the transmit process stops.
Bit 20	FTF	0x0	rw	Flush Transmit FIFO When this bit is set, the Tx FIFO controller logic is reset of its default values and thus all data in the Tx FIFO are either lost or flushed. This bit is cleared after the completion of the flushing operation. The operation mode register should not be written before this bit is cleared. The data that has been received by the MAC transmitter is not flushed and is going to be transferred, causing data underflow and runt frame transfer (If you want to change this bit through consecutive commands, a delay value greater than 4us is required between two consecutive operations.)
Bit 19: 17	Reserved	0x0	resd	Kept at its default value.
Bit 16: 14	TTC	0x0	rw	Transmit Threshold Control These bits control the threshold of the Tx FIFO. Transmission starts when the frame size in the Tx FIFO is greater than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are applicable only when the bit 21 (TSF) is reset. 000: 64 001: 128 010: 192 011: 256 100: 40 101: 32 110: 24 111: 16
Bit 13	SSTC	0x0	rw	Start or Stop Transmission Command When this bit is set, transmission is in the running state, and the DMA checks the transmit list at the current location

				and determines the frame to be transmitted. The DMA acquires the descriptor either from the current position in the list (the transmit list base address set by the transmit descriptor list address register) or from the position where the transmit process was stopped previously. If the current descriptor is owned by the DMA, the transmit process enters suspend state, and the bit 2 (transmit buffer unavailable) is set in the statue register. Transmission command is valid only when the transmission is stopped. If the transmit command were issued before setting the transmit descriptor list address register, the DMA will show unpredictable behavior. When this bit is cleared, transmit process enters stop state after the completion of a frame transmission. The next descriptor position in the transmit list is saved, and becomes the current position when transmission gets started. To change the list address, write a new value to the transmit descriptor list address register when this bit is reset. The newly written value becomes effective only when this bit is set again. The Stop Transmission Command is effective only when the current frame transmission is complete or transmit process enters suspend state.
Bit 12: 8	Reserved	0x00	resd	Kept at its default value.
Bit 7	FEF	0x0	rw	Forward Error Frames 1: All frames except runt error frames are forwarded to the DMA 0: Rx FIFO drops error frames (CRC error, collision error, giant frame, watchdog timeout and overflow). However, if the frame's start byte point has already been transferred to the application in Threshold mode, then the frames are not dropped. The Rx FIFO drops the error frames whose start bytes have not been transferred to the AHB bus.
Bit 6	FUGF	0x0	rw	Forward Undersized Good Frames When this bit is set, the Rx FIFO forwards undersized good frames including pad bytes and CRC (with no error and length less than 64 bytes). When this bit is cleared, the Rx FIFO drops all frames with a length less than 64 bytes, unless such a frame has already been transferred to the application due to a lower value than the receive threshold (e.g. RTC=01).
Bit 5	Reserved	0x0	resd	Kept at its default value.
Bit 4: 3	RTC	0x0	rw	Receive Threshold Control These two bits control the threshold of the Rx FIFO. Transfer to DMA starts when the frame in the Rx FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also automatically transferred. Value 11 is not applicable if the Rx FIFO size is configured to be 128 bytes. These bits are applicable only when the RSF bit equals 0. These bits are ignored when the RSF bit is set. 00: 64 01: 32 10: 96 11: 128
Bit 2	OSF	0x0	rw	Operate on Second Frame When this bit is set, it instructs the DMA to process a second frame of transmit data even before the status of the first frame is obtained.
Bit 1	SSR	0x0	rw	Start or Stop Receive When this bit is set, the receive process is in the running state, and the DMA attempts to acquire the descriptor from the receive list and processes incoming frames. The DMA acquires the descriptor either from the current position in

				the list (the receive list base address set by the receive descriptor list address register) or from the position where the receive process was stopped previously. If the current descriptor is owned by the DMA, the receive process enters suspend state, and the bit 7 (receive buffer unavailable) is set in the statue register. Reception command is valid only when the reception is stopped. If the reception command were issued before setting the receive descriptor list address register, the DMA will show unpredictable behavior. When this bit is cleared, Rx DMA operation is stopped after the completion of a frame reception. The next descriptor position in the receive list is saved, and becomes the current position when reception process is restarted. The Stop Rece[topm Command is effective only when the receive process enters in the running state (waiting for receive packet) or the suspend state.
Bit 0	Reserved	0x0	resd	Kept at its default value.

27.3.28 Ethernet DMA interrupt enable register (EMAC_DMAIE)

The EMAC_DMAIE register enables the interrupts reported by the status register. Setting a bit to 1'b1 enables a corresponding interrupt. All interrupts are disabled after a software or hardware reset.

Bit	Name	Reset value	Type	Description
Bit 31: 17	Reserved	0x0000	resd	Kept at its default value.
Bit 16	NIE	0x0	rw	Normal Interrupt enable When this bit is set, a normal interrupt summary is enabled. When this bit is cleared, a normal interrupt summary is disabled. This bit enables the following bits (in the statue register) EMAC_DMASTS[0]: Transmit interrupt EMAC_DMASTS[2]: Transmit buffer unavailable EMAC_DMASTS[6]: Receive interrupt EMAC_DMASTS[14]: Early receive interrupt
Bit 15	AIE	0x0	rw	Abnormal interrupt enable When this bit is set, an abnormal interrupt summary is enabled. When this bit is cleared, an abnormal interrupt summary is disabled. This bit enables the following bits (in the status register) EMAC_DMASTS[1]: Transmit process stopped EMAC_DMASTS[3]: Transmit Jabber timeout EMAC_DMASTS[4]: Transmit overflow EMAC_DMASTS[5]: Transmit data underflow EMAC_DMASTS[7]: Transmit buffer u unavailable EMAC_DMASTS[8]: Receive process stopped EMAC_DMASTS[9]: Receive watchdog timeout EMAC_DMASTS[10]: Early transmit interrupt EMAC_DMASTS[13]: Fatal bus error
Bit 14	ERE	0x0	rw	Early Receive interrupt Enable When this bit is set with the normal interrupt summary enable bit, the early receive interrupt is enabled. When this bit is cleared, the early receive interrupt is disabled.
Bit 13	FBEE	0x0	rw	Fatal Bus Error Enable When this bit is set with the abnormal interrupt summary enable bit, the fatal bus error interrupt is enabled. When this bit is cleared, the fatal bus error enable interrupt is disabled.
Bit 12: 11	Reserved	0x0	resd	Kept at its default value.
Bit 10	EIE	0x0	rw	Early transmit Interrupt Enable When this bit is set with the abnormal interrupt summary enable bit, the early transmit interrupt is enabled. When this bit is cleared, the early transmit interrupt is disabled.
Bit 9	RWTE	0x0	rw	Receive Watchdog Timeout Enable When this bit is set with the abnormal interrupt summary

				enable bit, the receive watchdog timeout interrupt is enabled. When this bit is cleared, the receive watchdog timeout interrupt is disabled.
Bit 8	RSE	0x0	rw	Receive Stopped Enable When this bit is set with the abnormal interrupt summary enable bit, the receive stopped interrupt is enabled. When this bit is cleared, the receive stopped interrupt is disabled.
Bit 7	RBUE	0x0	rw	Receive Buffer Unavailable Enable When this bit is set with the abnormal interrupt summary enable bit, the receive buffer unavailable interrupt is enabled. When this bit is cleared, the receive buffer unavailable interrupt is disabled.
Bit 6	RIE	0x0	rw	Receive Interrupt Enable When this bit is set with the normal interrupt summary enable bit, the receive interrupt is enabled. When this bit is cleared, the receive interrupt is disabled.
Bit 5	UNE	0x0	rw	Underflow Interrupt Enable When this bit is set with the abnormal interrupt summary enable bit, the underflow interrupt is enabled. When this bit is cleared, the underflow interrupt is disabled.
Bit 4	OVE	0x0	rw	Overflow Interrupt Enable When this bit is set with the abnormal interrupt summary enable bit, the overflow interrupt is enabled. When this bit is cleared, the overflow interrupt is disabled.
Bit 3	TJE	0x0	rw	Transmit Jabber Timeout Enable When this bit is set with the abnormal interrupt summary enable bit, the transmit Jabber timeout interrupt is enabled. When this bit is cleared, the transmit Jabber timeout interrupt is disabled.
Bit 2	TUE	0x0	rw	Transmit Buffer Unavailable Enable When this bit is set with the normal interrupt summary enable bit, the transmit buffer unavailable interrupt is enabled. When this bit is cleared, the transmit buffer unavailable interrupt is disabled.
Bit 1	TSE	0x0	rw	Transmit Stopped Enable When this bit is set with the abnormal interrupt summary enable bit, the transmit stopped interrupt is enabled. When this bit is cleared, the transmit stopped interrupt is disabled.
Bit 0	TIE	0x0	rw	Transmit Interrupt Enable When this bit is set with the normal interrupt summary enable bit, the transmit interrupt is enabled. When this bit is cleared, the transmit interrupt is disabled.

The Ethernet interrupt is generated only when the TST or PMT bit is set in the DMA status register with other interrupts unmasked, ow when the NIS/AIS is enabled with other interrupts enabled.

27.3.29 Ethernet DMA missed frame and buffer overflow counter register (EMAC_DMAMFBOCNT)

The DMA contains two counters to track the number of missed frames during reception. Tis register reports the current value of the counter. The counter is used for the purpose of diagnosis. The bit [15: 0] indicates the number of missed frames due to the host buffer being unavailable. The bit [27: 17] indicate the number of missed frames due to buffer overflow (MTL and MAC) and runt frames dropped by the MTL.

Bit	Name	Reset value	Type	Description
Bit 31: 29	Reserved	0x0	resd	Kept at its default value.
Bit 28	OBFOC	0x0	rrc	Overflow Bit for FIFO Overflow Counter This bit is set whenever an overflow occurs on the overflow frame counter ([27: 17]), that is, the Rx FIFO overflows, and the overflow frame counter reaches its maximum value. In this case, the overflow frame counter is reset to all zero, and this bit indicates that a toggle has occurred.
Bit 27: 17	OFC	0x000	rrc	Overflow Frame Counter

				These bits indicate the number of frames missed by the application.
Bit 16	OBMFC	0x0	rrc	Overflow Bit for Missed Frame Counter This bit is set whenever an overflow occurs on the missed frame counter ([15: 0]), that is, the DMA ignores incoming frames due to the host receive buffer being unavailable, and the missed frame counter reaches its maximum value. In this case, the missed frame counter is reset to all zero, and this bit indicates that a toggle has occurred.
Bit 15: 0	MFC	0x0000	rrc	Missed Frame Counter This field indicates the number of frames missed by the controller due to the host receive buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame.

27.3.30 Ethernet DMA current transmit descriptor register (EMAC_DMACTD)

The EMAC_DMACTD register points to the start address of the transmit descriptor being read by the DMA.

Bit	Name	Reset value	Type	Description
Bit 31: 0	HTDAP	0x0000 0000	ro	Host Transmit Descriptor Address Pointer These bits are cleared when reset. The DMA updates the pointer during operation.

27.3.31 Ethernet DMA current receive descriptor register (EMAC_DMACRD)

The EMAC_DMACRD register points to the start address of the receive descriptor being read by the DMA.

Bit	Name	Reset value	Type	Description
Bit 31: 0	HRDAP	0x0000 0000	ro	Host Receive Descriptor Address Pointer These bits are cleared when reset. The DMA updates the pointer during operation.

27.3.32 Ethernet DMA current transmit buffer address register (EMAC_DMACTBADDR)

The EMAC_DMACTBADDR register points to the transmit buffer address being read by the DMA.

Bit	Name	Reset value	Type	Description
Bit 31: 0	HTBAP	0x0000 0000	ro	Host Transmit Buffer Address Pointer These bits are cleared when reset. The DMA updates the pointer during operation.

27.3.33 Ethernet DMA current receive buffer address register (EMAC_DMACRBADDR)

The EMAC_DMACRBADDR register points to the receive buffer address being read by the DMA.

Bit	Name	Reset value	Type	Description
Bit 31: 0	HRBAP	0x0000 0000	ro	Host Receive Buffer Address Pointer These bits are cleared when reset. The DMA updates the pointer during operation.

27.3.34 Ethernet MMC control register (EMAC_MMCCTRL)

The EMAC_MMCCTRL register defines the operating mode of the management counters.

Bit	Name	Reset value	Type	Description
Bit 31: 4	Reserved	0x0000000	resd	Kept at its default value.
Bit 3	FMC	0x0	rw	Freeze MMC Counter When this bit is set, it freezes all the MMC counters to their current value. None of the MMC counters are updated due

				to any transmitted or received frame until this bit is set to 0. If the Reset on Read bit is set while the MMC counter is being read, the counter is also cleared.
Bit 2	RR	0x0	rw	Reset on Read When this bit is set, the MMC counter is reset to 0 after being read. The counter is cleared when the least significant byte bit [7: 0] is read.
Bit 1	SCR	0x0	rw	Stop Counter Rollover When this bit is set, the counter does not roll over to 0 after it reaches the maximum value.
Bit 0	RC	0x0	rw	Reset Counter When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

27.3.35 Ethernet MMC receive interrupt register (EMAC_MMCRI)

The EMAC_MMCRI register contains the interrupts generated in the following conditions:

- Receive statistic counters reaches half their maximum values (32-bit counter corresponds to 0x8000_0000, and 16-bit counter corresponds to 0x8000)
- Receive statistic counters exceed their maximum values (32-bit counter corresponds to 0xFFFF_FFFF, and 16-bit counter corresponds to 0xFFFF)

When the counter stops rolling, an interrupt is set but the counter is still all 1. The EMAC_MMCRI is a 32-bit register. An interrupt bit is cleared when the MMC counter that generates the interrupt is read. The least significant byte bit [7: 0] of the corresponding counter must be read in order to clear the interrupt bit.

Bit	Name	Reset value	Type	Description
Bit 31: 18	Reserved	0x0000	resd	Kept at its default value.
Bit 17	RGUF	0x0	rrc	Received Good Unicast Frames This bit is set when the received good unicast frame counter reaches the maximum value or half the maximum value.
Bit 16: 7	Reserved	0x000	resd	Kept at its default value.
Bit 6	RFAE	0x0	rrc	Received Frames Alignment Error This bit is set when the received frame counter with alignment error reaches the maximum value or half the maximum value.
Bit 5	RFCE	0x0	rrc	Received Frames CRC Error This bit is set when the receive frame with CRC error reaches the maximum value or half the maximum value.
Bit 4: 0	Reserved	0x00	resd	Kept at its default value.

27.3.36 Ethernet MMC transmit interrupt register (EMAC_MMCTI)

The EMAC_MMCTI register contains the interrupts generated in the following conditions: when the transmit statistic counters reach half their maximum values (32-bit counter corresponds to 0x8000_0000, and 16-bit counter corresponds to 0x8000), and when the transmit statistic counters exceed their maximum values (32-bit counter corresponds to 0xFFFF_FFFF, and 16-bit counter corresponds to 0xFFFF). When the counter stops rolling, an interrupt is set but the counter is still all 1. The EMAC_MMCTI is a 32-bit register. An interrupt bit is cleared when the MMC counter that generates the interrupt is read. The least significant byte bit [7: 0] of the corresponding counter must be read in order to clear the interrupt bit.

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value.
Bit 21	TGF	0x0	rrc	Transmitted Good Frames This bit is set when the transmitted good frame counter reaches its maximum value or half its maximum value.
Bit 20: 16	Reserved	0x00	resd	Kept at its default value.
Bit 15	TGFMSC	0x0	rrc	Transmitted Good Frames More Single Collision This bit is set when the transmitted good frame after more than a single collision counter reaches its maximum value or half its maximum value.

Bit 14	TSCGFCI	0x0	rrc	Transmitted Single Collision Good Frame Counter Interrupt This bit is set when the transmitted good frame after a single collision counter reaches its maximum value or half its maximum value.
Bit 13: 0	Reserved	0x0000	resd	Kept at its default value.

27.3.37 Ethernet MMC receive interrupt register (EMAC_MMCRIM)

The EMAC_MMCRIM contains the masks for interrupts generate when the receive statistic counters reach half their maximum values or their maximum values. This register is a 32-bit register.

Bit	Name	Reset value	Type	Description
Bit 31: 18	Reserved	0x0000	resd	Kept at its default value.
Bit 17	RUGFCIM	0x0	rw	Received Unicast Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the received good unicast frame counter reaches half its maximum value or its maximum value.
Bit 16: 7	Reserved	0x000	resd	Kept at its default value.
Bit 6	RAEFACIM	0x0	rw	Received Alignment Error Frame Alignment Counter Interrupt Mask Setting this bit masks the interrupt when the received alignment error frame counter reaches half its maximum value or its maximum value.
Bit 5	RCEFCIM	0x0	rw	Received CRC Error Frame Counter Interrupt Mask Setting this bit masks the interrupt when the received CRC error frame counter reaches half its maximum value or its maximum value.
Bit 4: 0	Reserved	0x00	resd	Kept at its default value.

27.3.38 Ethernet MMC transmit interrupt register (EMAC_MMCTIM)

The EMAC_MMCTIM contains the masks for interrupts generate when the transmit statistic counters reach half their maximum values or their maximum values. This register is a 32-bit register.

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value.
Bit 21	TGFCIM	0x0	rw	Transmitted Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the transmitted good frame counter reaches half its maximum value or its maximum value.
Bit 20: 16	Reserved	0x00	resd	Kept at its default value.
Bit 15	TMCGFCIM	0x0	rw	Transmitted Multiple Collision Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the transmitted good frame after more than a single collision counter reaches half its maximum value or its maximum value.
Bit 14	TSCGFCIM	0x0	rw	Transmitted Single Collision Good Frame Counter Interrupt Mask Setting this bit masks the interrupt when the transmitted good frame after a single collision counter reaches half its maximum value or its maximum value.
Bit 13: 0	Reserved	0x0000	resd	Kept at its default value.

27.3.39 Ethernet MMC transmitted good frame single collision counter register (EMAC_MMCTFSCC)

This register maintains the number of successfully transmitted frames after a single collision in half-duplex mode.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TGFSCC	0x0000 0000	ro	Transmitted Good Frames Single Collision Counter) This field maintains the transmitted good frames after a single collision counter.

27.3.40 Ethernet MMC transmitted good frame more than a single collision counter register (EMAC_MMCTFMSCC)

This register maintains the number of successfully transmitted frames after more than a single collision in half-duplex mode.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TGFMSCC	0x0000 0000	ro	Transmitted Good Frame More Than a Single Collision Counter This field maintains the transmitted good frames after more than a single collision counter.

27.3.41 Ethernet MMC transmitted good frames counter register (EMAC_MMCTFCNT)

This register maintains the number of the transmitted good frames.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TGFC	0x0000 0000	ro	Transmitted Good Frames Counter

27.3.42 Ethernet MMC received frames with CRC error counter register (EMAC_MMCRFCECR)

This register maintains the number of the received good frames with CRC error.

Bit	Name	Reset value	Type	Description
Bit 31: 0	RFCEC	0x0000 0000	ro	Received Frames CRC Error Counter Received frames with CRC error.

27.3.43 Ethernet MMC received frames with alignment error counter register (EMAC_MMCRFAECNT)

This register maintains the number of the received frames with alignment error.

Bit	Name	Reset value	Type	Description
Bit 31: 0	RFAEC	0x0000 0000	ro	Received Frames Alignment Error Counter Received frames with alignment error.

27.3.44 Ethernet MMC received good unicast frames counter register (EMAC_MMCRGUFCNT)

This register maintains the number of the received good unicast frames.

Bit	Name	Reset value	Type	Description
Bit 31: 0	RGUFC	0x0000 0000	ro	Received Good Unicast Frames Counter

27.3.45 Ethernet PTP time stamp control register (EMAC_PTPTSCTRL)

This register controls the generation of system time in the receiver and the generation of time stamp in a PTP packet.

Bit	Name	Reset value	Type	Description
Bit 31: 19	Reserved	0x0000	resd	Kept at its default value.
Bit 18	EMAFPFF	0x0	rw	Enable MAC Address For PTP Frame Filtering When this bit is set, the MAC address (matches any of the MAC address registers) is used for PTP frame filtering while the PTP is directly sent by the Ethernet.
Bit 17: 16	SPPFTS	0x0	rw	Select PTP Packets For Taking Snapshot 00: Normal clock 01: Boundary clock 10: End-to-End Transparent Clock 11: Point-to-Point Transparent Clock
Bit 15	ESFMRTM	0x0	rw	Enable Snapshot For Message Relevant To Master When this bit is set, it enables snapshots for messages relevant to master. Otherwise, it enables snapshots for

				messages relevant to slave.
Bit 14	ETSFEM	0x0	rw	Enable Timestamp Snapshot For Event Messages When this bit is set, it enables time stamp snapshots for event messages only (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is cleared, time stamp snapshots are applicable to all the messages except Announce, Management and Signaling.
Bit 13	EPPFSIP4U	0x1	rw	Enable Processing of PTP Frames Sent over IPv4-UDP When this bit is set, the MAC receiver processes the PTP encapsulated in UDP over IPv4 packet. When this bit is cleared, the MAC ignores the PTP transferred over UDP-IPv4 packet. This bit is set by default.
Bit 12	EPPFSIP6U	0x0	rw	Enable Processing of PTP Frames Sent over IPv6-UDP When this bit is set, the MAC receiver processes the PTP encapsulated in UDP over IPv6 packet. When this bit is cleared, the MAC ignores the PTP sent over UDP-IPv6 packet.
Bit 11	EPPEF	0x0	rw	Enable Processing of PTP over EMAC Frames When this bit is set, the MAC receiver processes the PTP that is directly encapsulated in the Ethernet frames. When this bit is cleared, the MAC ignores the PTP over EMAC frames.
Bit 10	EPPV2F	0x0	rw	Enable PTP packet Processing for Version 2 Format When this bit is set, it enables PTP packet processing in the format of 1588 V2. Otherwise, 1588 V1 format is used for PTP packet processing. Refer to *PTP process and control* on page 155 for more details on IEEE 1588 V1 and V2.
Bit 9	TDBRC	0x0	rw	Timestamp Digital or Binary Rollover Control When this bit is set, time stamp low register starts rolling after the 0x3B9A_C9FF value (1 ns precision), and the time stamp (high) second is incremented. When this bit is cleared, the rollover value of the subsecond register is 0x7FFF_FFFF. The subsecond increment must be configured according to PTP reference clock frequency and the value of this bit.
Bit 8	ETAF	0x0	rw	Enable Timestamp for All Frames When this bit is set, it enables time stamp snapshot for all received frames on the MAC.
Bit 7: 6	Reserved	0x0	resd	Kept at its default value.
Bit 5	ARU	0x0	rw	Addend Register Update When this bit is set, the Ethernet PTP time stamp addend register's contents are updated on the PTP block for fine correction. This bit is cleared when the update is completed. This register bit must be read as 0 before being set.
Bit 4	TITE	0x0	rw	Timestamp Interrupt Trigger Enable When this bit is set, a time stamp interrupt is enabled if the system time becomes greater than the value written in the target time register. This bit is cleared when the time stamp trigger interrupt is generated.
Bit 3	TU	0x0	rw	Timestamp Update When this bit is set, the system time is updated (added or subtracted from) with the value programmed in the system time second update register and system time nanosecond update register. This bit must be read as 0 before being updated. This bit is cleared after the hardware update is completed. Time stamp high word register (if enabled) is not updated.
Bit 2	TI	0x0	rw	Timestamp Initialize When this bit is set, the system time is initialized (overwritten) with the value specified in the system time second update register and system time nanosecond update register.

				This bit must be read as 0 before being updated. This bit is cleared after the initialization. Time stamp high word register (if enabled) is not updated.
Bit 1	TFCU	0x0	rw	Timestamp Fine or Coarse Update When this bit is set, it indicates that the system time is updated using a fine update method. When this bit is cleared, it indicates that the system time is updated using a coarse update method.
Bit 0	TE	0x0	rw	Timestamp Enable When this bit is set, time stamp function is enabled for transmit and receive frames. Once disabled, the time stamp function is not added for transmit and receive frames, and the time stamp generator is suspended as well. Once enabled, the time stamp (system time) should be initialized. On the receive side, the MAC processes 1588 frames only when this bit is set.

Correlation between time stamp snapshot and register bits

SPPFTS Bit 17: 16	ESFMRTM Bit 15	ETSFEM Bit 14	PTP message
00 or 01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00 or 01	1	1	Delay_Req
00 or 01	0	1	SYNC
10	N/A	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
10	N/A	1	SYNC, Follow_Up
11	N/A	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp
11	N/A	1	SYNC, Pdelay_Req, Pdelay_Resp

1 : N/A= Not applicable

2 : X=Irrelevant

27.3.46 Ethernet PTP subsecond increment register (EMAC_PTPSSINC)

This register is present only when the IEEE1588 time stamp function is selected without an external time stamp input. In Coarse Update mode (TSCFUPDT bit), the value in this register is added to the system time every clk_ptp_ref_i clock cycle. In Fine Update mode, the value in this register is added to the system time whenever the accumulator has an overflow.

Bit	Name	Reset value	Type	Description
Bit 31: 8	Reserved	0x000000	resd	Kept at its default value.
Bit 7: 0	SSIV	0x00	rw	Sub-Second Increment Value The value programmed in this field is incremented with the value of the subsecond register at every clock cycle (of clk_ptp_i). For example, if the PTP clock is 50 MHz (20 ns), when the system time nanosecond register is 1 ns accuracy (by setting the bit 9 in the EMAC_PTPTSCTRL register), the value of these bits should be configured to 20 (0x14). When the TSCTRLSSR is cleared, nanosecond register resolution is ~0.465ns accuracy. In this case, the value of these bits should be configured to 43 (0x2B), that is, 20ns/0.465.

27.3.47 Ethernet PTP time stamp high register (EMAC_PTPTSH)

System time second register and system time nanosecond register indicate the current value of the system time maintained by the MAC. This value is updated on a continuous basis.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TS	0x0000 0000	ro	Timestamp Second This field indicates the second value of the current system time maintained by the MAC.

27.3.48 Ethernet PTP time stamp low register (EMAC_PTPTSL)

This register contains the lower 32 time bits. It is a read-only register containing the subsecond system time value.

Bit	Name	Reset value	Type	Description
Bit 31	AST	0x0	ro	Add or Subtract Time When this bit is set, the time value is subtracted from the value of the update register. When this bit is cleared, the time value is added to the value of the update register.
Bit 30: 0	TSS	0x0000 0000	ro	Timestamp Sub Seconds This field indicates the subsecond system time with 0.46 ns accuracy. When the bit 9 is set in the EMAC_PTPTSCTRL register, each bit represents 1 ns, with the value programmed not exceeding the 0x3B9A_C9FF.

27.3.49 Ethernet PTP time stamp high update register (EMAC_PTPTSHUD)

System time second update register and system nanosecond update register initializes or updates the system time maintained by the MAC. It is required to write both registers before setting the TSINIT or TSUPDT bit in the EMAC_PTPTSCTRL register.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TS	0x0000 0000	rw	Timestamp Second This field indicates the second value that is to be initialized or added to the system time.

27.3.50 Ethernet PTP time stamp low update register (EMAC_PTPTSLUD)

This register is present only when the IEEE1588 time stamp function is selected without an external time stamp input.

Bit	Name	Reset value	Type	Description
Bit 31	AST	0x0	rw	Add or Subtract Time When this bit is set, the time value is subtracted from the value of the update register. When this bit is cleared, the time value is added to the value of the update register.
Bit 30: 0	TSS	0x0000 0000	rw	Timestamp Sub Seconds This field indicates the subsecond system time with 0.46 ns accuracy. When the bit 9 is set in the EMAC_PTPTSCTRL register, each bit represents 1 ns, with the value programmed not exceeding the 0x3B9A_C9FF.

27.3.51 Ethernet PTP time stamp addend register (EMAC_PTPTSAD)

This register value is used only when the system time is configured for Fine update mode. This register value is added to a 32-bit accumulator at every clock cycle (of clk_ptp_ref_i). The system time is updated whenever the accumulator overflows.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TAR	0x0000 0000	rw	Timestamp Addend Register This field indicates the 32-bit time value to be added to the accumulator in order to achieve time synchronization.

27.3.52 Ethernet PTP target time high register (EMAC_PTPTTH)

Target time second register and target time subsecond register are used to schedule an interrupt event when the system time exceeds the value programmed in these registers.

Bit	Name	Reset value	Type	Description
Bit 31: 0	TTSR	0x0000 0000	rw	Target Time Seconds Register This register stores the time value in seconds. When the time stamp value equals or exceeds both target time stamp registers, the MAC starts or stops PPS signal output depending on the bit [6: 5] of the PPS control register. An interrupt is generated if enabled.

27.3.53 Ethernet PTP target time low register (EMAC_PTPTTL)

Bit	Name	Reset value	Type	Description
Bit 31: 0	TTLR	0x0000 0000	rw	Target Timestamp Low Register This register stores the time (signed) in nanoseconds. When the value of the time stamp equals both target time stamp registers, the MAC starts or stops PPS signal output depending on the TRGTMODSEL0 (bit [6: 5]) of the PPS control register. An interrupt is generated if enabled. When the bit 9 (TSCTRLSSR) is set in the MAC_PTPTSCTR register, the value of this field cannot exceed the 0x3B9A_C9FF. The actual time that starts or stops PPT signal output may have an error of up to 1 subsecond increment value.

27.3.54 Ethernet PTP time stamp status register (EMAC_PTPTSSR)

Bit	Name	Reset value	Type	Description
Bit 31: 2	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 1	TTTR	0x0	ro	Timestamp Target Time Reached When this bit is set, it indicates the value programmed when the system time equals or exceeds the target time second register and target time nanosecond register.
Bit 0	TSO	0x0	ro	Timestamp Seconds Overflow When this bit is set, it indicates that the time stamp value (V2 format supported) overflows and has exceeded the 32'hFFFF_FFFF.

27.3.55 Ethernet PTP PPS register (EMAC_PTPPPSCR)

Bit	Name	Reset value	Type	Description
Bit 31: 4	Reserved	0x0000000	resd	Kept at its default value.
Bit 3: 0	POFC	0x0	rw	PPS0 Output Frequency Control The output of this field depends on the emac_pps_sel bit (bit 15 in the CRM_MISC2 register) Emac_pps_sel=0: 0000: 1 Hz, use binary rollover control, pulse width is 125 ms; use digital rollover, pulse width is 100 ms 0001: 2 Hz, use binary rollover control, duty cycle is 50% (digital rollover is not recommended) 0010: 4 Hz, se binary rollover control, duty cycle is 50% (digital rollover is not recommended) 0011: 8 Hz, use binary rollover control, duty cycle is 50%(digital rollover is not recommended) 0100: 16 Hz, use binary rollover control, duty cycle is 50% (digital rollover is not recommended) 1111: 32.768 kHz, use binary rollover control, duty cycle is 50% (digital rollover is not recommended) Emac_pps_sel=1: 0000: 1 Hz, pulse width is one clk_ptp cycle 0001: For binary rollover, 2hz, duty cycle 50%; For digital rollover, 1hz (digital rollover is not recommended) 0010: For binary rollover, 4hz, duty cycle 50%; For digital rollover, 2hz (digital rollover is not recommended) 0011: For binary rollover, 8hz, duty cycle 50%; For digital rollover, 4hz (digital rollover is not recommended) 1111: For binary rollover, 32.768khz, duty cycle 50%; For digital rollover, 16.384khz (digital rollover is not recommended) Digital rollover is not recommended when the PPS is non-zero value, because PPS output waveforms will be irregular (although its average frequency is always correct in any one-second window) in these cases.

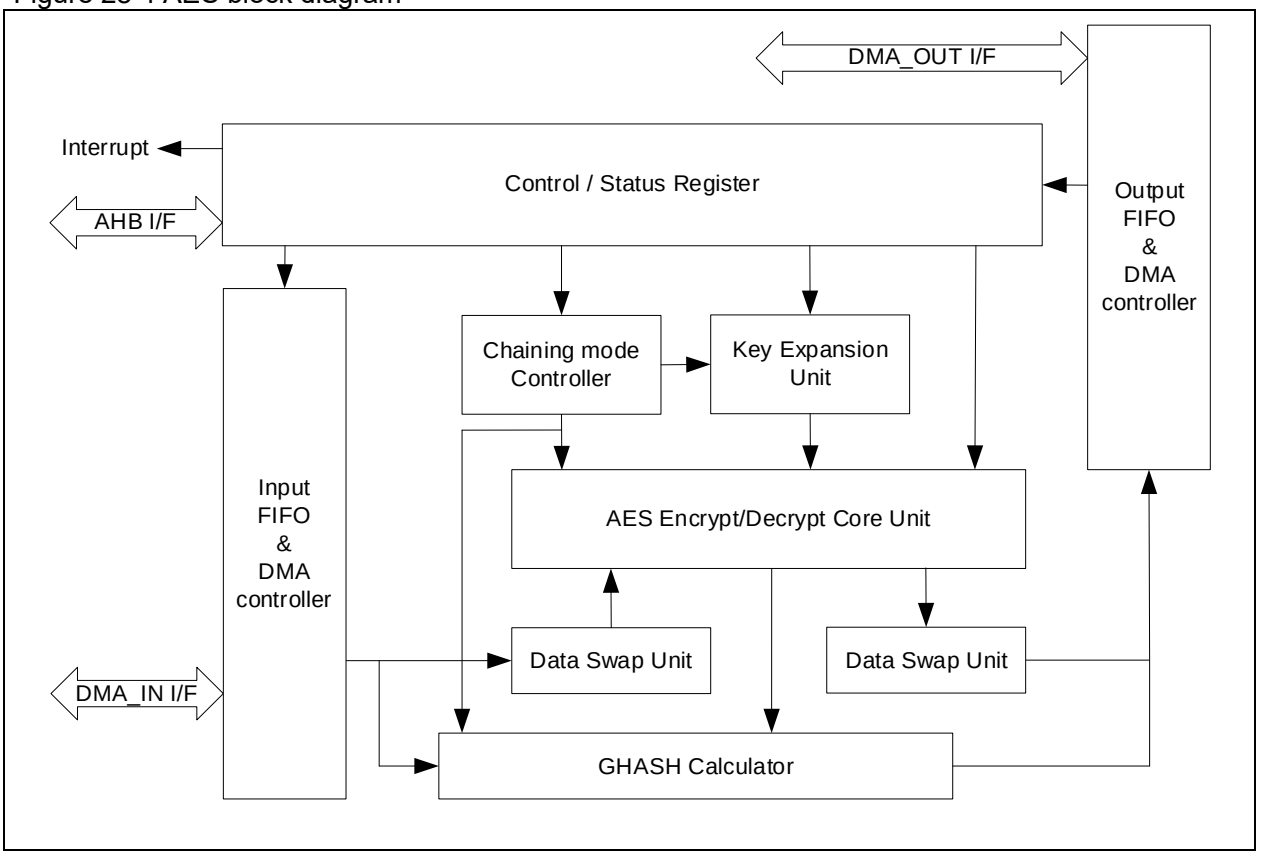
28 AES hardware processor (AES)

28.1 Introduction

The AES hardware accelerator encrypts or decrypts data using an algorithm fully compliant with the NIST advanced encryption standard defined in Fedar information processing standards publication 197.

The AES accelerator works on a 128-bit data block processing. It supports cipher key lengths of 128-bit, 192-bit and 256-bit. It also supports multiple operation modes. Data can be swapped based on a bit, a half-word and a byte level. The AES accelerator can use DMA for data transfers without the intervention of the CPU. It can also be monitored through interrupt status.

Figure 28-1 AES block diagram



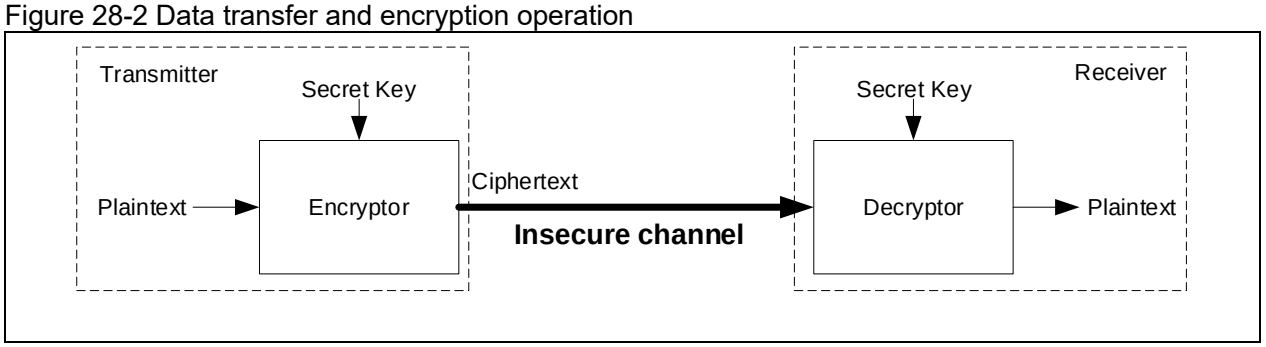
28.2 AES main features

- Compliance with NIST FIPS PUB-197 Advanced Encryption Standard (AES)
- Support for cipher key lengths of 128-bit, 192-bit and 256-bit
- Support the following operation modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Counter with CBC-MAC (CCM) mode
- Support Galois message authentication code (GMAC) mode
- Support four-word initialization vector for CBC, CTR, GCM, CCM
- Data transfer using two-channel DMA capability
- Support data input and output on the half-word, byte and bit level, in terms of word
- Support software to interrupt encryption/decryption operation based on data block level

28.3 Key algorithms

When the transmitter tries to share a message with the receiver for communication purposes via an insecure channel, a shared encryption key – a symmetric key algorithm, can be used to convert plaintext into a ciphertext. After receiving data, the receiver uses this encryption key to decrypt data into the original plaintext. This is also to prevent those potential eavesdropper on insecure channels from accessing plaintext as a result of the encrypted key.

Figure 28-2 Data transfer and encryption operation



28.3.1 Encryption operation

Encryption mode is activated by setting OPR=0 in the AES_CTRL register. In this mode, the input plaintext is encrypted by accessing the AES_IDE register. The encrypted ciphertext is then stored in the output buffer which is obtained by reading the AES_ODT register. Before AES being enabled, it is necessary to first configure the AES_KEYx (key register), AES_IVx (initial vector register) and AES_CTRL (control register) registers. If the chaining mode is Galois counter mode (GCM) or GCM combined with Galois message authentication code (GMAC) mode, that is CHN=3 or CHN4=4, it is also necessary to empty the AES_Six register. For more information, please refer to section 28.4.

28.3.2 Decryption operation

The round key used for decryption is the reversed order arrangement of the cipher round key. Therefore, to decrypt ciphertext into a plaintext, the AES accelerator needs to first extend the secret key into a round key before decryption operation, which is different from encryption. There are two ways available for the AES to perform decryption, based on the number of cipher blocks.

Single block decryption

In Electronic codebook mode or cipher block chaining mode (by setting CHN=0 or CHN=1 respectively), if there is only a single block in the ciphertext, it can be decrypted by setting OPR=3 (key derivation then single decryption) in the AES_CTRL register. In this case, the user needs to read the AES_IDT register four times to input the block ciphertext in a sequence to implement decryption operation. The decrypted plaintext is then stored in the output buffer, which can be obtained by the user by reading the AES_ODT register four times. Note that the AES_KEYx) register, AES_IVx register and AES_CTRL register must be configured before enabling the AES accelerator.

Chaining block decrption

In Electronic codebook mode or cipher block chaining mode (by setting CHN=0 or CHN=1 respectively), if there is more than one blocks of ciphertexts, it is necessary to perform key derivation before switching to decrption mode.

Enable key derivation: configure the AES_KEYx register, set OPR=1 in the AES_CTRL register and then enable AES by setting the AESEN bit. After key derivation configuration, the AES_KEYx is not allowed to be modified.

Next step is to enable decrption mode by setting OPR=2. In this mode, the user reads the AES_IDE register and inputs the ciphertext for decription. The decrypted message is then stored in the output buffer for users to obtain by reading the AES_ODT register. Note that the AES_IVx register and AES_CTRL register must be configured before AES being enabled.

In Counter mode, Galois counter mode, or counter mode combined with cipher block chaining message authentication code (CHN=2, CHN=3 or CHN=4, respectively), if there is more than one blocks of ciphertexts, there is no need to perform key derivation. Instead, enable decryption mode simply by setting OPR=2 in the AES_CTRL register. In this mode, the user inputs the ciphertext for decrption operation by reading the AES_IDT register. The decrypted message is stored into output buffer for the user to obtain by reading the AES_OTD register. Note that before enabling the AES, it is necessary to configure AES_KEYx, AES_IVx and AES_CTRL registers in advance. If Galois counter mode or Counter mode combined with cipher block chaining message authentication mode is selected by setting CHN=3 or CHN=4 respectively, the AES_SIx) register must be cleared as well.

28.4 AES chaining modes

The AES supports multiple typical chaining modes and uses the same set of secret key to perform encryption or decryption operations. For a plaintext or ciphertext that has more than one blocks of message (i.e. 16 bytes) in size, there are the following chaining modes that can be used to encrypt or decrypt them.

28.4.1 Electronic codebook mode (ECB)

The Electronic codebook mode is the most basic chaing mode. It is enabled by setting CHN=0 in the AES_CTRL register. In this mode, a message is split into several blocks based on a 16 bytes level. Each block is then encrypted or decrypted separately, as shonw in Figure 28-3.

Figure 28-3 ECB encryption

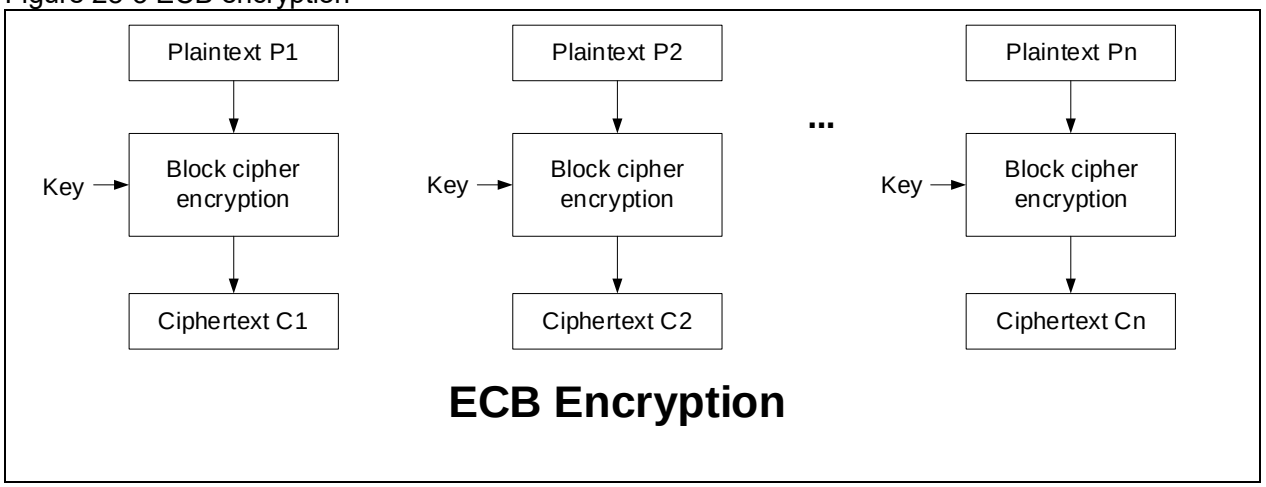
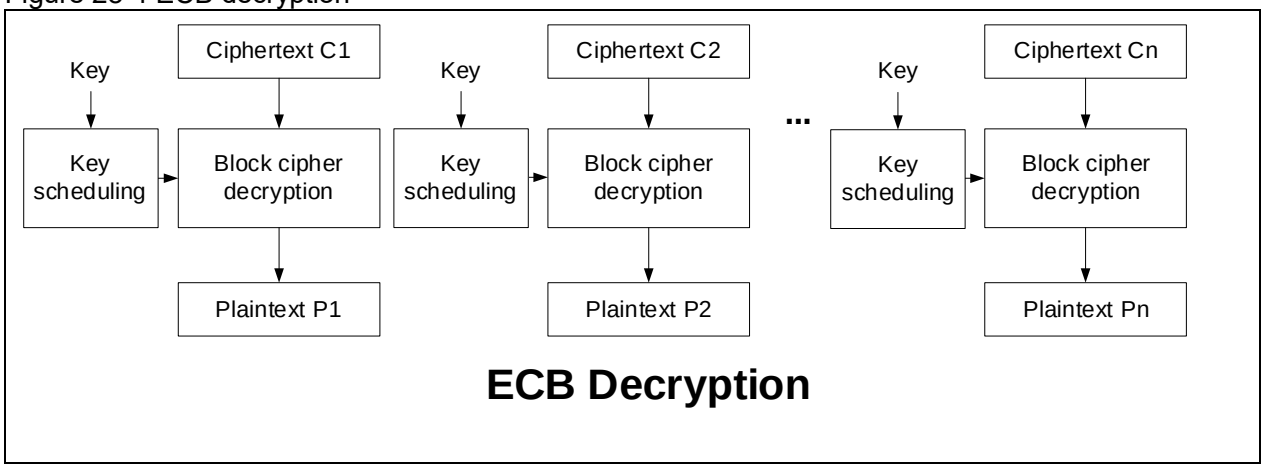


Figure 28-4 ECB decryption



Encryption/decryption configure procecure

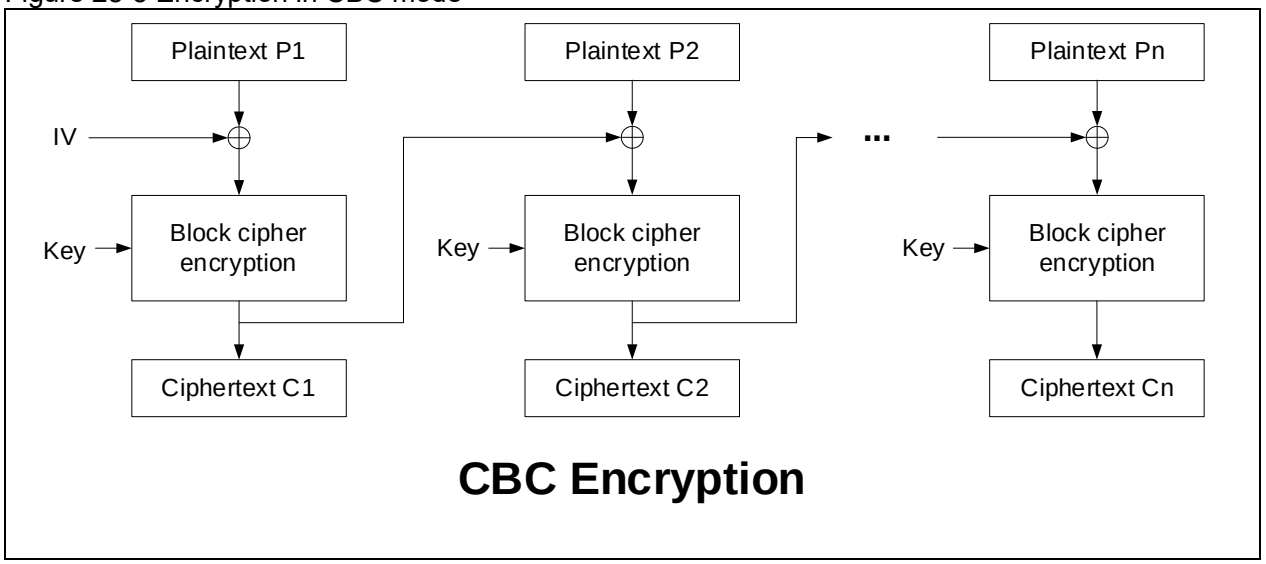
In ECB mode, encryption or decryption are implemented in the following order:

1. **Preparation**
   - Set CHN=0 in the AES_CTRL register
   - Configure AES_KEYx register
2. **Secret key derivation for decryption**
   - Skip this step if it is for encryption operation
   - Set OPR=1 in the AES_CTRL register
   - Enable AES by setting the AESEN bit in the AES_CTRL register, and the AESEN will be automatically reset after the completion of the secret key derivation
   - Wait for the PDFS to be set in the AES_STS register, and then clear PDFS by writing 1 ot the PDFC bit in the AES_FCLR register
3. **Encryption/decription stage**
   - Set OPR=0 or OPR=2 in the AES_CTRL register, depending on encryption or decryption operation
   - Append the ciphertext or plaintext based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
   - After finishing data append, reset AESEN

28.4.2 Cipher block chaining mode (CBC)

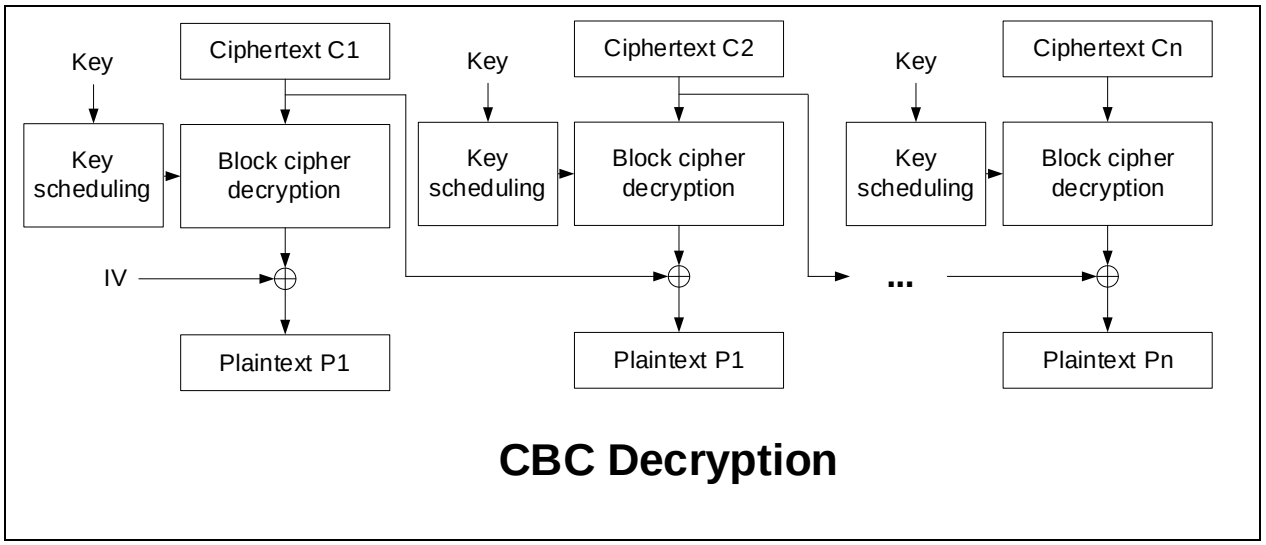
The block ciphertexts in CBC mode are composed of all the previous block plaintexts that are computed. This mode is enabled by setting CHN=1 in the AES_CTRL register. In this mode, the plaintext is split into plaintext blocks which are then XOR-ed with the previous block ciphertext before being encrypted. In order to ensure data uniqueness, the initialization vector is used on the first block to replace the previous ciphertext. The initialization vector is configured through the AES_IVx register. Figure 28-5 shows the encryption flowchart in CBC mode.

Figure 28-5 Encryption in CBC mode



In CBC decryption mode, the incoming ciphertext is split into cipher blocks, which are decrypted through a core unit for encryption/decryption, and then XOR-ed with the previous ciphertext to obtain a plaintext. On the first ciphertext block, the initialization vectors, which must be the same as that of encryption operation, are used to replace the previous ciphertext for decryption operation. The figure below shows the decryption flowchart in CBC mode.

Figure 28-6 Decryption in CBC mode



Encryption/decryption configuration procedure:

1. **Preparation**
   - Set CHN=1 in the AES_CTRL register
   - Configure AES_KEYx register and AES_IVx register
2. **Secret key derivation for decryption**
   - Skip this step if it is for encryption operation
   - Set OPR=1 in the AES_CTRL register
   - Enable AES by setting the AESEN bit in the AES_CTRL register, and the AESEN will be automatically reset after the completion of the secret key derivation
   - Wait for the PDFS to be set in the AES_STS register, and then clear PDFS by writing 1 ot the PDFC bit in the AES_FCLR register
3. **Encryption/decription stage**
   - Set OPR=0 or OPR=2 in the AES_CTRL register, depending on encryption or decryption operation
   - Append the ciphertext or plaintext based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
   - After finishing data append, reset AESEN

28.4.3 Counter mode (CTR)

The Counter mode is based on a stream key for decryption or encryption operation. A key stream block is generated by decrypting a nonce and combining this nonce with the successively accumulated counter values, and it is then XOR-ed with a plaintext block to obtain a ciphertext. Due to the symmetry of XOR-ed operations, the same process is applied to the encryption and decryption. This mode is selected by setting CHN=2 in the AES_CTRL register. The figures below show CTR encryption and decryption.

Figure 28-7 CTR encryption mode

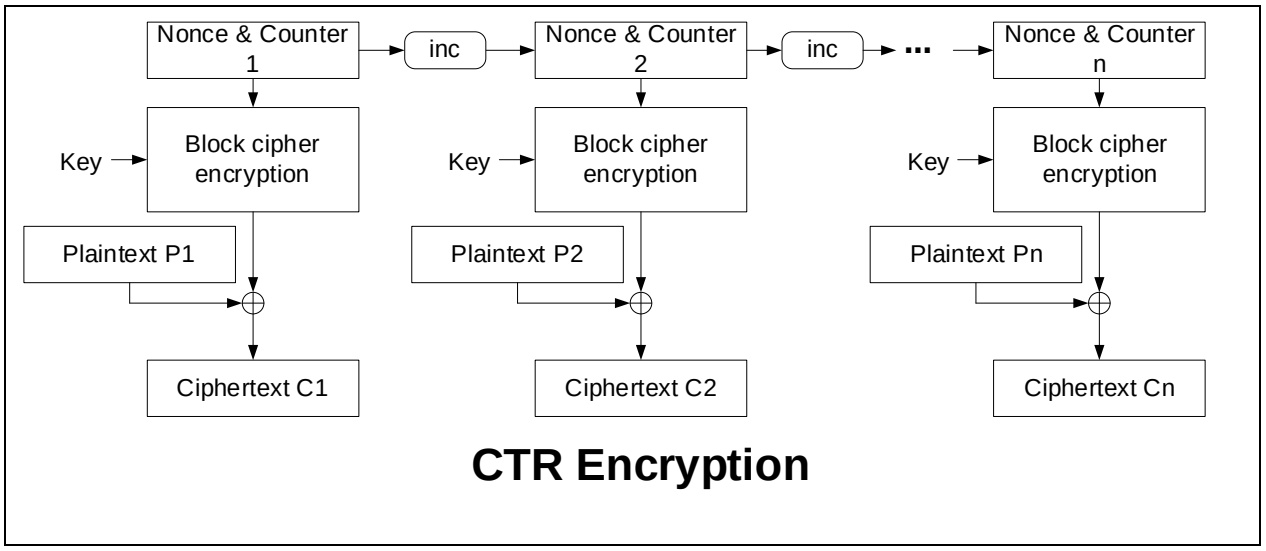
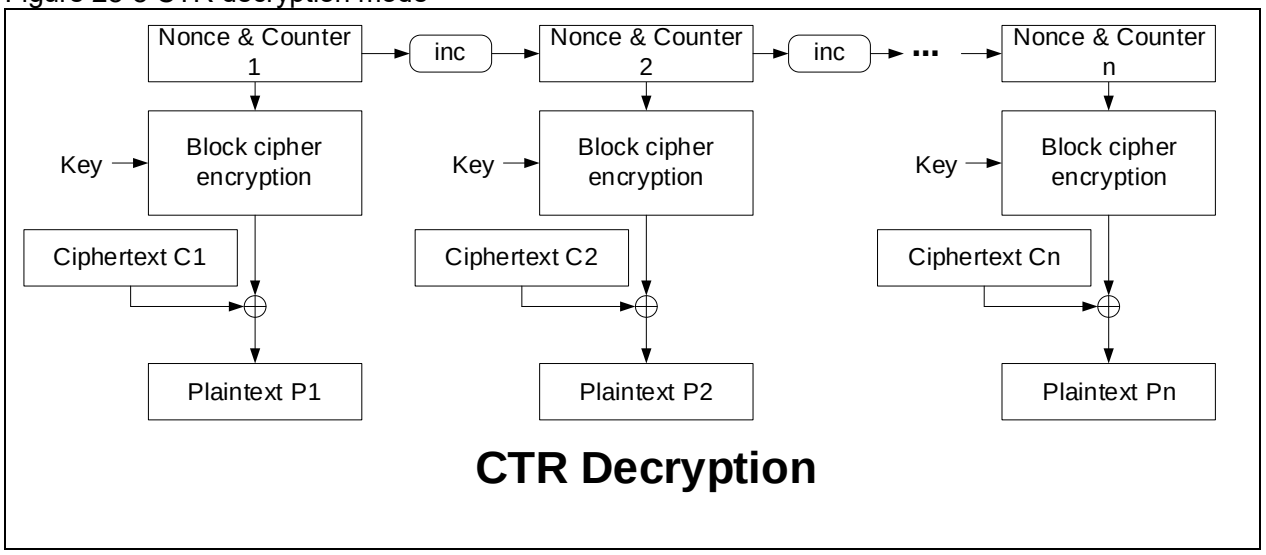


Figure 28-8 CTR decryption mode



Encryption/decryption configuration procedure:

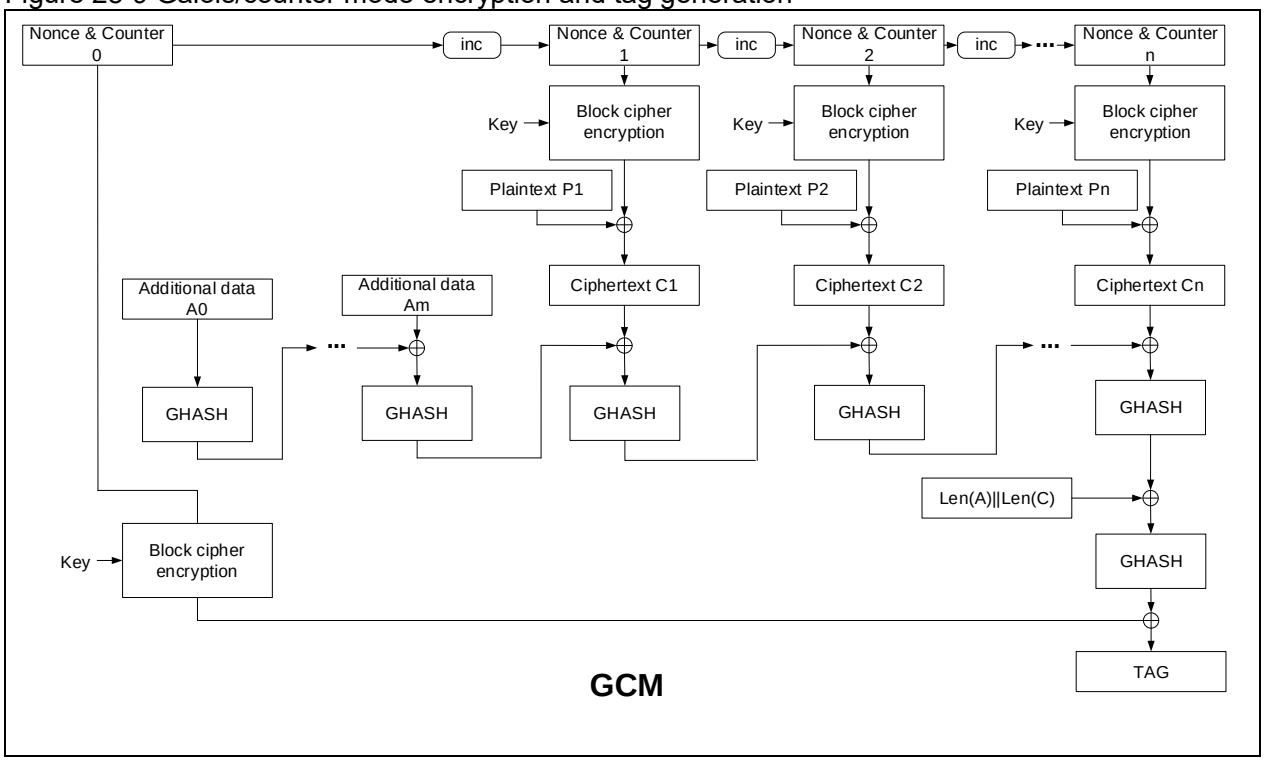
Unlike with ECB mode and CBC mode, in CTR mode, the content in a counter is encrypted to produce a key stream block. Therefore before starting decryption operation, there is no need to perform key derivation.

1. **Preparation**
   - Set CHN=2 in the AES_CTRL register
   - Configure AES_KEYx register and AES_IVx register
2. **Encryption/decription stage**
   - Set OPR=0 or OPR=2 in the AES_CTRL register, depending on encryption or decryption operation
   - Append the ciphertext or plaintext based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
   - After finishing data append, reset AESEN

28.4.4 Galois/counter mode (GCM)

The GCM mode is based on counter mode and Galois Hash functions for data integrity and confidentiality. Data encryption and decryption are implemented in counter mode, as decribed in section 28.4.3. The ciphertext after encryption operation, along with the additional authentication data, are processed in blocks using GHASH to generate a tag. This mode is enabled by setting CHN=3 in the AES_CTRL register.

Figure 28-9 Galois/counter mode encryption and tag generation



Encryption/decryption configuration procedure:

In GCM mode, encryption and decryption are implemented in six stages.

1. **Preparation**
   - Set CHN=3 in the AES_CTRL register
   - Configure AES_KEYx register and AES_IVx register
   - Clear the AES_Six register
2. **Initialization of Galois hash functions**
   - Set PRC=0 in the AES_CTRL register
   - Enable AES by setting the AESEN bit in the AES_CTRL register
   - Wait for the PDFS bit to be set in the AES_STS register, after that, clear PDFS by writing 1 to PDFC in the AES_FCLR register
3. **Additional authentication data phase**
   - Skip this step if there is no such data
   - Set PRC=1 or OPR=2 in the AES_CTRL register
   - Append the additional data based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
4. **Encryption/decryption phase**
   - Skip this step if GMAC mode is being used
   - Set PRC=2 in the AES_CTRL register
   - Append the ciphertext or plaintext based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
5. **Encryption/decryption of the last block**

- Skip this step if GMAC mode is being used
- If the significant data in the last block to process is inferior to 16 bytes, the NDD bit in the AES_CTRL register is used to append the remainder of the last clock with dummy data
- Set LST=1 in the AES_CTRL register
- Input the last block for encryption or decryption in four times to the AES_IDT register. The data that is not part of the payload will be discarded by hardware (the excessive data are written in the NDD field)
- Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register

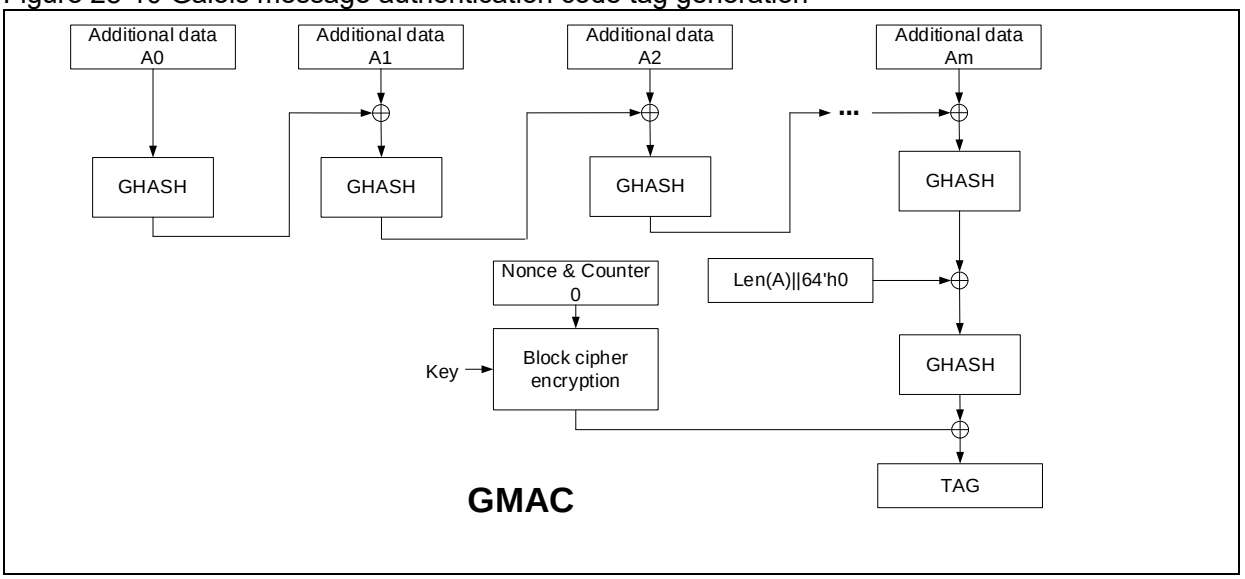
6. Tag phase

- Set PRC=3 in the AES_CTRL register
- Read the AES_IDT register in four times to input the last block. The last block is composed of the additional authentication data and cipher text, totaling 16 bytes
- Wait until the PDFS is set in the AES_STS register. After that, read the AES_ODT register in four times to obtain the tag
- Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register. Reset AESEN bit to disable AES accelerator.

28.4.5 Galois message authentication code (GMAC)

The GMAC mode is a variant of the Galois/counter mode combined with counter mode. it uses the Galois hash functions only to generate a tag for data integrity, with no encryption operation required. For how to generate a tag, please refer to the encryption/decryption configure procedure (skip step 4) in the section 28.4.4 for details

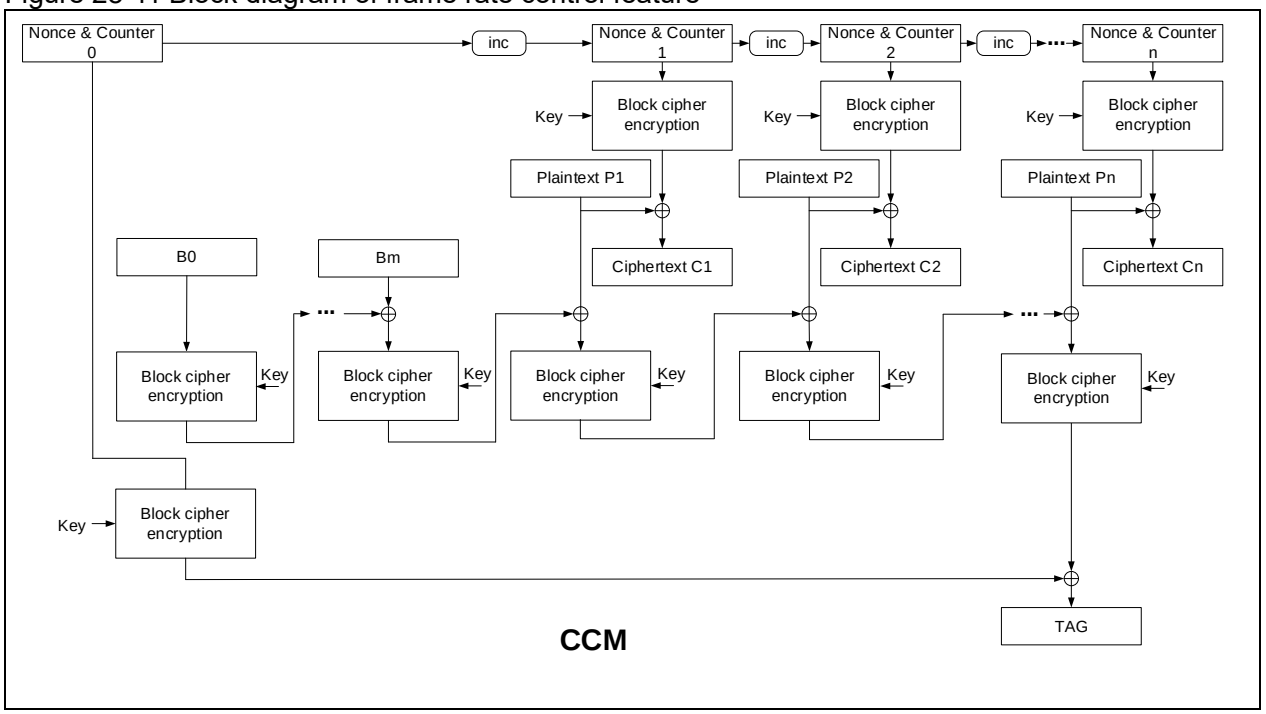
Figure 28-10 Galois message authentication code tag generation



28.4.6 Counter with CBC-MAC (CCM)

The counter with CBC-MAC mode (CCM) is based on counter mode and cipher block chaining mode as well as message authentication mode for data integrity and confidentiality. Data encryption and decryption is processed in counter mode, as described in section 28.4.3. The ciphertext after encryption operation, along with the additional authentication data, are process in blocks to generate a tag using the cipher block chaining mode with message authentication code. CCM mode is enabled by setting CHN=4 in the AES_CTRL register.

Figure 28-11 Block diagram of frame rate control feature



Encryption/decryption configuration procedure:

In GCM mode, encryption and decryption are implemented in five stages.

1. **Preparation**
   - Set CHN=4 in the AES_CTRL register
   - Configure AES_KEYx register and AES_IVx register
   - Clear the AES_Six register
   - Set OPR=0 or OPR=2 in the AES_CTRL register, depending on decryption or encryption
2. **Additional authentication data phase**
   - Skip this step if there is no such data
   - Set PRC=1 in the AES_CTRL register
   - Enable AES peripheral by setting the AESEN bit in the AES_CTRL register, and append the additional authentification data on a 16bytes level, using CPU or DMA, see section 28.5.2 for details
3. **Encryption/decryption phase**
   - Set PRC=2 in the AES_CTRL register
   - Append the ciphertext or plaintext based on 16-byte level, using CPU or DMA, see section 28.5.2 for more infomration
4. **Encryption/decryption of the last block**
   - If the significant data in the last block to process is inferior to 16 bytes, the NDD bit in the AES_CTRL register is used to append the remainder of the last clock with dummy data
   - Set LST=1 in the AES_CTRL register
   - Input the last block for encryption or decryption in four times to the AES_IDT register. The data that is not part of the payload will be discarded by hardware (the excessive data are written in the NDD field)
   - Wait until the PDFS is set in the AES_STS register, and read the AES_ODT register in four times to obtain the operation result of the last block
   - Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register
5. **Tag phase**
   - Set PRC=3 in the AES_CTRL register
   - Read the AES_IDT register in four times to input CTR0

- Wait until the PDFS is set in the AES_STS register. After that, read the AES_ODT register in four times to obtain the tag
- Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register. Reset AESEN bit to disable AES accelerator.

28.5 Data formats and append

28.5.1 Data formats

The AES core works on 128-bit data blocks for encryption or decrption, as specified in AES_CTRL register. Data are organized in small endian order in the AES core. The figures below show the formats of the ciphertext, plaintext, secret key and initialization vector in the registers.

Figure 28-12 Ciphertext and plaintext data formats

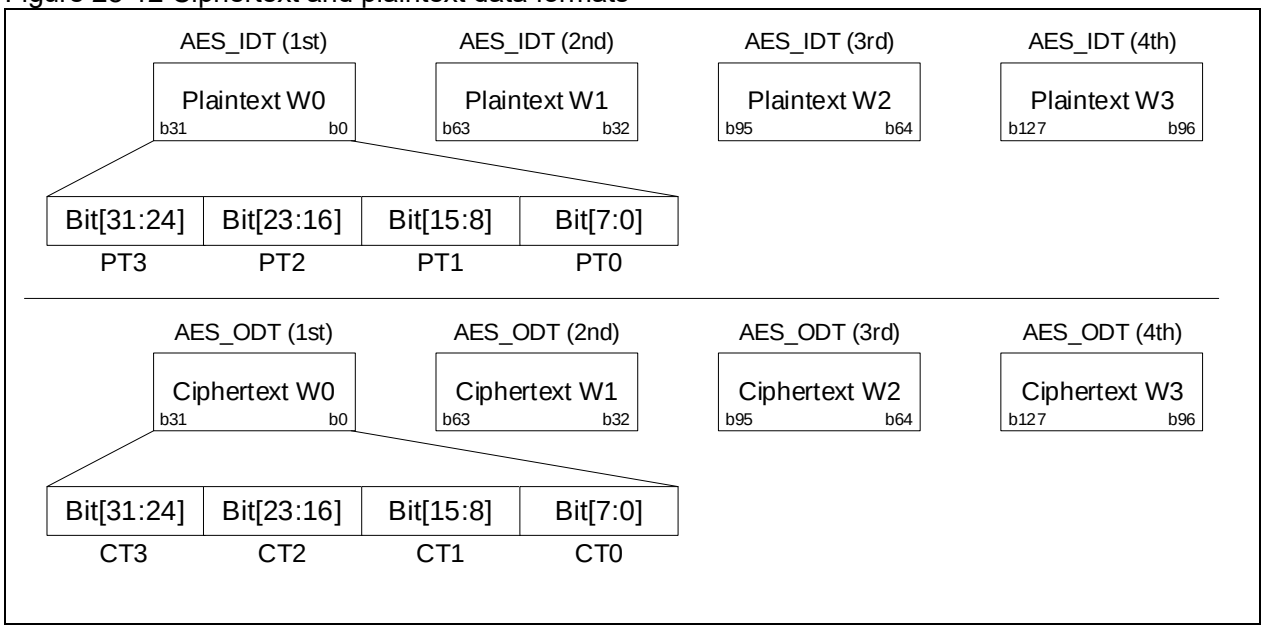
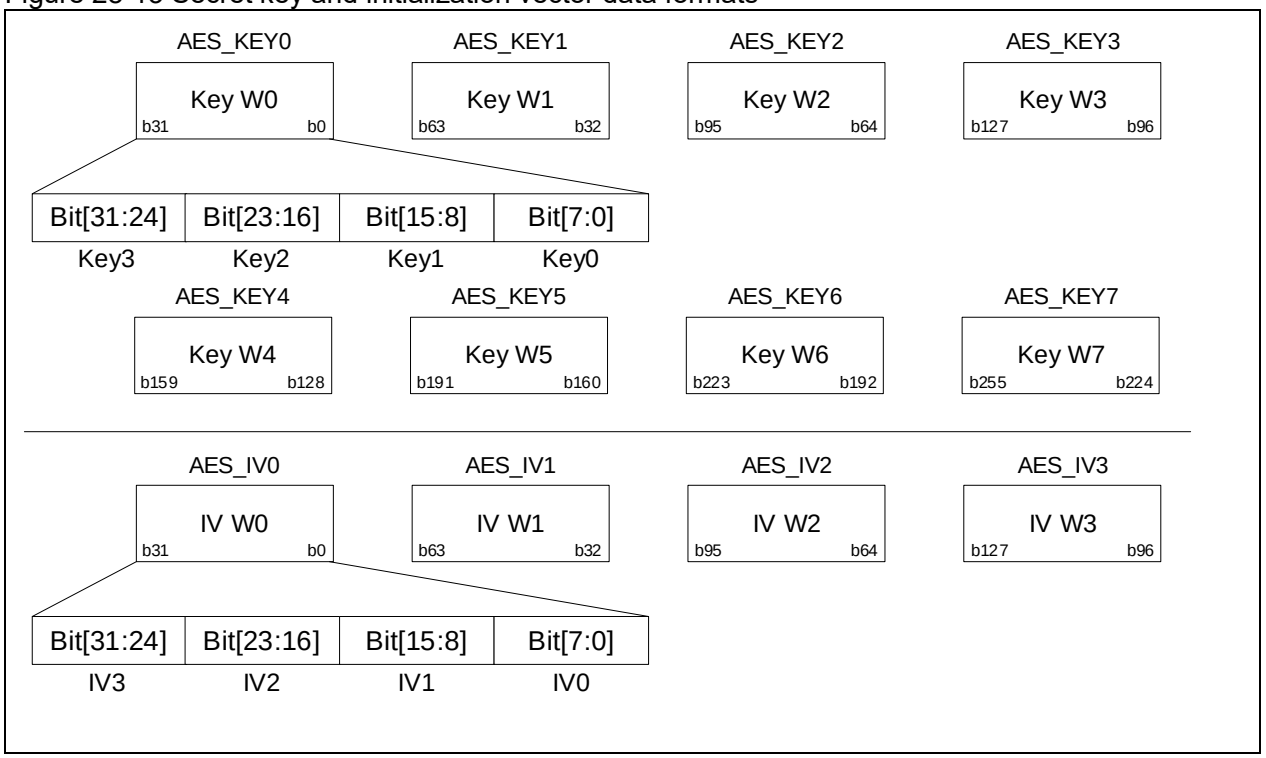
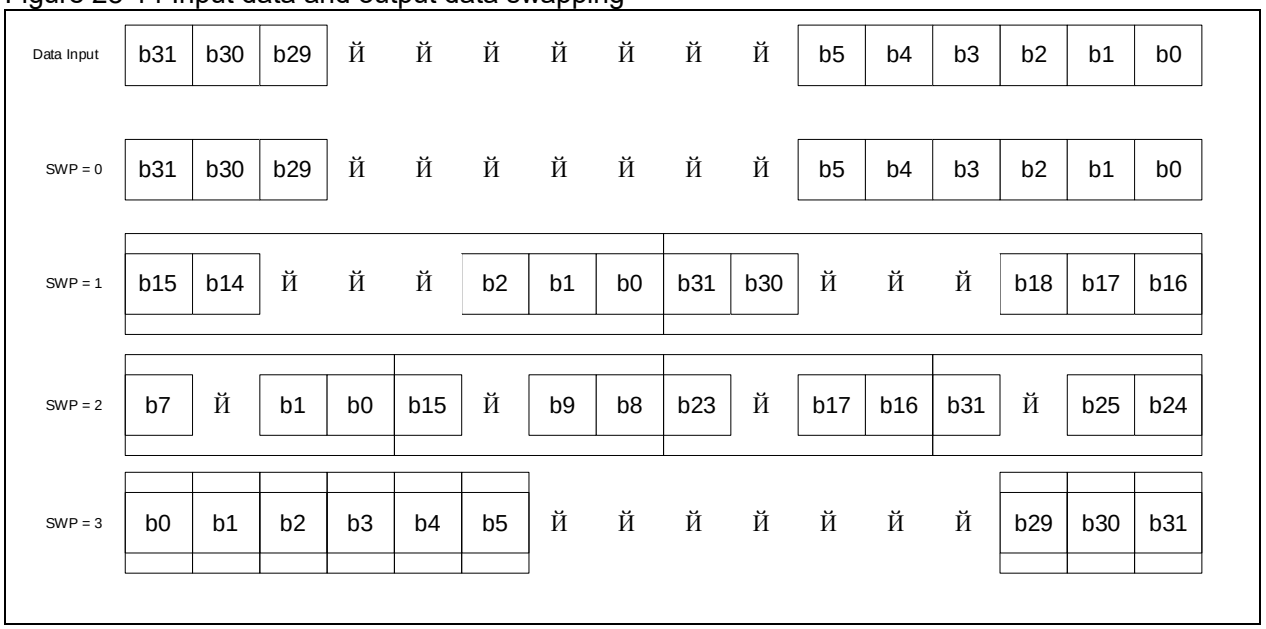


Figure 28-13 Secret key and initialization vector data formats



In order to accommodate data formats in different applications, the AES peripheral supports data swapping based on a bit, a byte or a half-word level for ciphertext and plaintext. The SWP bit in the AES_CTRL register is used to select data that are to be input in the AES_IDT register and data in the AES_ODT register for data swapping.

Figure 28-14 Input data and output data swapping



28.5.2 Data append

In the encryption and decryption phase of each chaining mode, and in the additional authentication data phase in GCM mode and CCM mode, data are split into 128-bit data blocks for processing. Data blocks are input the AES_IDT register one by one until the completion of data processing. Then these data are obtained by reading the AES_ODT register. The whole process is called data append, which can be done using CPU or DMA.

Data append through CPU

The CPU uses flag polling or interrupts to control data append.

1. If using interrupts to receive interrupts, set RWEIE=1 and PDIE=1 in the AES_CTRL register.
2. Configure AES_CTRL register, AES_KEYx register and AES_IVx register. Enable AES through the AESEN bit in the AES_CTRL register.
3. Read the AES_IDT register in four times using CPU to input each block data
4. Wait for the completion of the AES hardware encryption operation, there are some minor differences depending on the control methods
   - If using flag polling to control data append, CPU reads the AES_STS register multiple times until the PDFS is set
   - If using interrupts to control data append, the CPU, after receiving interrupts, reads the AES_STS register to check whether the PDFS is set or not
5. For encryption/decryption stage in chaing modes, read the AES_ODT register in four times to obtain the computational data; for additionl authentication data ste in GCM/CCM mode, skip this step.
6. Clear the PDFS by writing 1 to the PDFC bit in the AES_FCLR register.
7. If the data to process is greater than 16 bytes, that is, there is more than one data block, repeat step 3 to step 7.
8. If the data to process is inferior to 16 bytes, append the data block with zeros. For the additional authentication state in GCM/CCM mode, configure the NDD bit and set LST bit in the AES_CTRL register.
9. Wait until the AES encryption operatio is completed, the same as step 4
10. If the NDD bit in the AES_CTRL register is configured, read the AES_STS register to confirm

whether the NZDFS bit is set or not

11. For encryption/decryption stage in chaining modes, read the AES_ODT register in four times to obtain the computational data; for the additional authentication data stage in GCM/CCM mode, skip this step
12. Clear the PDFS and NZDFS by writing 1 to PDFC and NZDFC in the AES_FCLR register. After that, reset AESEN.

Data append using DMA

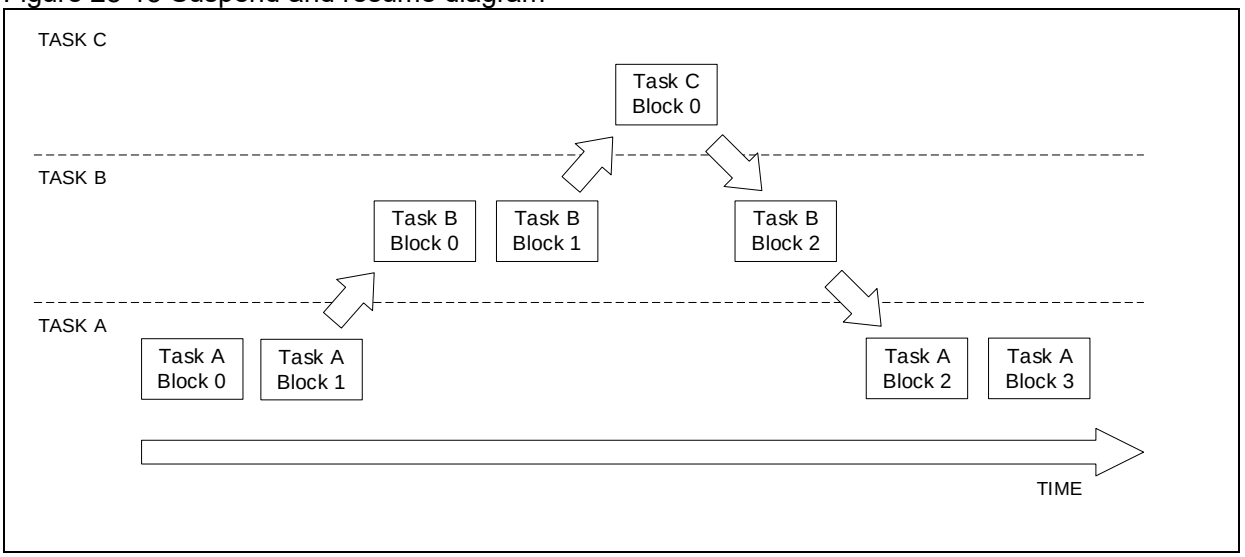
With this method, all the data transfers are managed by DMA and AES. To use this method, proceed as follows:

1. If the data to process ia not a multiple of 16 bytes, pad the remainder of the data block with zeros
2. Configure two DMA controllers and enable them. One DMA controller is used to transfer the data to process from the memory to the AES_IDT register, while the other DMA is to transfer the processed data from the AES_ODT register to the memory. Note that during encryption/decryption stage in GCM or CCM mode and if the data to process is not a multiple of 16 bytes, the DMA size is configured to deduct the last data block (encryption/decryption of the last data block is performed in the next stage)
3. Enable DMA write and read channels by wrting 1 to the DMAWE and DMARE in the AES_CTRL register
4. Enable data processing complete interrupt and read/write error interrupt by setting the PDIE and RWEIE in the AES_CTRL register
5. Configure AES_CTRL, AES_KEYx and AES_IVx reigsters, and enable AES through the AESEN bit in the AES_CTRL register
6. After AES enable, data transfers are automatically managed by DMA and AES. After the completion of data append, the AES will set the PDFS bit in the AES_STS register and issue interrupt to CPU
7. Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register, and then reset AESEN.

28.5.3 Suspend and resume

The AES peripheral supports the suspend of a chaining mode in which data is being appended. This task suspend is performed in data blocks. It is used to suspend and record the tasks in process so as to free resources for other tasks with a higher priority.

Figure 28-15 Suspend and resume diagram



Data suspend and resume procedure using CPU

1. Wait until the PDFS bit is set in the AES_STS register, depending on flag polling method or interrupts
2. For encryption/decryption state in chaining modes, read the AES_ODT register in four times to obtain the processed data; for the additional authentication stage in GCM/CCM mode, skip this step

3. Clear the PDFS by writing 1 to the PDFC in the AES_FCLR register. after that, reset AESEN to disable the AES
4. Save the configurations of the current AES_CTRL register, AES_KEYx register, AES_IVx register and AES_SIx register
5. Configure and perform tasks with a higher priority
6. Resume the previous task and restore the AES_CTRL register, AES_KEYx register, AES_IVx register and AES_SIx register using the saved configurations in step 4
7. Enable AES thoruhg the AESEN in the AES_CTRL register to return to the previous interrupted task.

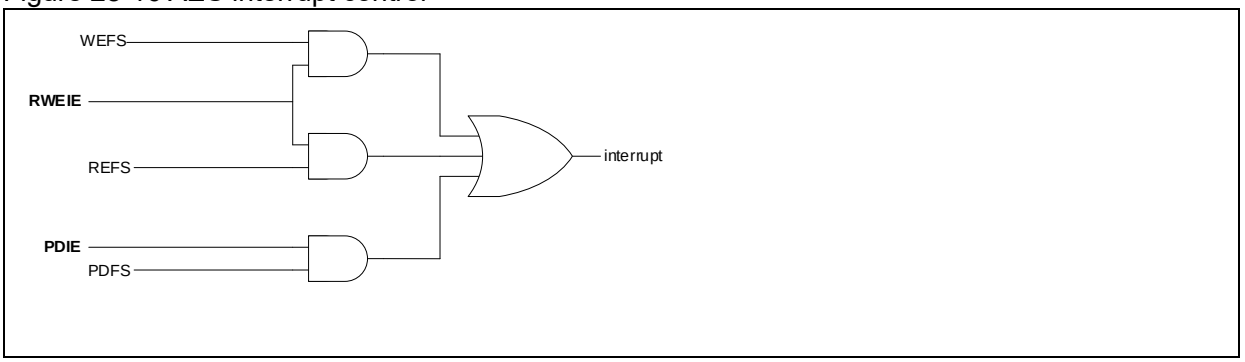
Data suspend and resume procedure using DMA

1. Disable DMA write channel by resetting the DMAWE bit in the AES_CTRL register
2. Wait until the interrupt is sent to CPU, and confirm whether the PDFS is set in the AES_STS register
3. Disable DMA read channel by writing 1 to the PDFC in the AES_FCLR register
4. Clear PDFS by writing 1 to the PDFC in the AES_FCLR register. after that, reset AESEN to disable AES
5. Save the configurations or contents of the current AES_CTRL register, AES_KEYx register, AES_IVx register and AES_SIx register
6. If the DMA controller is also to be release for tasks with a higher priority, save the remaining DMA length
7. Configure and perform tasks with a higher priority
8. Resume the previously interrupted task, and restore the AES_CTRL register, AES_KEYx register, AES_IVx register and AES_SIx register using the saved contents in step 5
9. If the DMA controller is also to be resumed, re-configure and enable DMA with the initial DMA address, length, and the remaining DMA length
10. Enable AES thoruhg the AESEN in the AES_CTRL register to return to the previous interrupted task.

28.6 Interrupts

There are three status flags for the AES peripheral.

Figure 28-16 AES interrupt control



Encryption/decryption computation completed flag (PDFS)

This flag is set by hardware upon the completionof the computation. If the PDIE bit is set in the AES_CTRL register, an interrupt is sent to the CPU. This PDFS bit is cleared by writing 1 to the PDFC in the AES_FCLR register.

Read error flag (REFS)

The REFS bit is set in the AES_STS register when the user is reading the AES_ODT register while the output buffer is in idle state. If the RWEIE bit is set in the AES_CTRL register, an error interrupt is isused to the CPU. This flag is cleared by writing 1 to the REFC bit in the AES_FCLR register.

Write error flag (WEFS)

The WEFS bit is set in the AES_STS register when the user is reading the AES_IDT register while the

inputbuffer is full. If the RWEIE bit is set in the AES_CTRL register, an error interrupt is isused to the CPU. This flag is cleared by writing 1 to the WEFC bit in the AES_FCLR register.

In addition, the NZDFS bit is used to indicate whether the processed data blocks are with dummy data. Dummy data are only used to pad the block which is short of 16 bytes. The NZDFS bit is set upon the completion of the computation when the LST and NDD are configured in the AES_CTRL register. No interrupt is generated as this flag is without interrupt enable settings. The NZDFS bit is cleared by writing 1 to the NZDFC in the AES_FCLR register.

28.7 ASE registers

These peripheral registers have to be accessed by words (32 bits).

Table 28-1 AES register map and reset values

Register	Offset	Reset value
AES_CTRL	0x000	0x0000 0000
AES_STS	0x004	0x0000 0000
AES_IDT	0x008	0x0000 0000
AES_ODT	0x00C	0x0000 0000
AES_KEY0	0x010	0x0000 0000
AES_KEY1	0x014	0x0000 0000
AES_KEY2	0x018	0x0000 0000
AES_KEY3	0x01C	0x0000 0000
AES_IV0	0x020	0x0000 0000
AES_IV1	0x024	0x0000 0000
AES_IV2	0x028	0x0000 0000
AES_IV3	0x02C	0x0000 0000
AES_KEY4	0x030	0x0000 0000
AES_KEY5	0x034	0x0000 0000
AES_KEY6	0x038	0x0000 0000
AES_KEY7	0x03C	0x0000 0000
AES_SI0	0x040	0x0000 0000
AES_SI1	0x044	0x0000 0000
AES_SI2	0x048	0x0000 0000
AES_SI3	0x04C	0x0000 0000
AES_SI4	0x050	0x0000 0000
AES_SI5	0x054	0x0000 0000
AES_SI6	0x058	0x0000 0000
AES_SI7	0x05C	0x0000 0000
AES_FCLR	0x060	0x0000 0000

28.7.1 ASE control register (AES_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31: 25	Reserved	0x00	resd	Kept at its default value.
Bit 24	LST	0x0	rw	Processing last block 0: The processing is not on the last block 1: The last block is being processed This bit is cleared upon the completion of the computation

				when the register is set to 1..
Bit 23:20	NDD	0x0	rw	Number of dummy data 0: No dummy data 1-15: number of dummy data This register is valid only when LST=1.
Bit 19	Reserved	0x0	resd	Kept at its default value.
Bit 18:17	KL	0x0	rw	Key length 0: 128 bits (AES-128) 1: 192 bits (AES-192) 2: 256 bits (AES-256) 3: Reserved
Bit 16: 15	Reserved	0x0	resd	Kept at its default value.
Bit 14: 13	PRC	0x0	rw	Processing stage 0: Galois hash functions initialization state 1: Additional authentication data stage 2: Encryption/decryption stage 3: Tag stage This bit is effective onhy when in CH=3 or CHN=4.
Bit 9: 8	DMARE	0x0	rw	DMA read channel enable 0: DMA read channel disabled. The user reads the AES_ODT register using CPU to obtain data. 1: DMA read channel enabled. The user reads the AES_ODT register using DMA to obtain data.
Bit 11	DMAWE	0x0	rw	DMA write channel enable 0: DMA write channel disabled. The user reads the AES_IDT register using CPU to obtain data. 1: DMA write channel enabled. The user reads the AES_IDT register using DMA to obtain data.
Bit 10	RWEIE	0x0	rw	Read and write error interrupt enable 0: Read and write error interrupt disabled 1: Read and write error interrupt enabled
Bit 9	PDIE	0x0	rw	Encryption/Decryption processing done interrupt enable 0: Encryption/Decryption processing completion interrupt disabled 1: Encryption/Decryption processing completion interrupt enabled
Bit 8	Reserved	0x0	resd	Kept at its default value.
Bit 7:5	CHN	0x0	rw	Chaining mode 0: Electronic codebook mode (ECB) 1: Cipher block chaining mode (CBC) 2: Counter mode (CTR) 3: Galois/counter mode (GCM) 4: Counter mode with cipher block chaining message authentication code (CCM) 5-7: Reserved
Bit 4:3	OPR	0x0	rw	Encryption/Decryption mode 0: Encryption 1: Secret key derivation 2: Decryption 3: Secret key derivation/single decryption
Bit 2:1	SWP	0x0	rw	Data Swapping 0: None 1: Half-word 2: Byte 3: Bit
Bit 0	AESEN	0x0	rw	AES Enable 0: AES disable 1: AES Enable The AES register must be configured before AES enable.

28.7.2 AES status register (AES_STS)

Bit	Name	Reset value	Type	Description
Bit 31：5	Reserved	0x00000000	resd	Kept at its default value.

Bit 4	NZDFS	0x0	ro	Non-zero dummy data processed flag status 0: Non-zero dummy data processed flag 1: Aero dummy data processed flag This register is cleared by writing 1 to the NZDFC bit in the AES_FCLR register.
Bit 3	PBS	0x0	ro	Crypto-processor busy status 0: Crypto-processor computation complete 1: Crypto-processor computation is busy When this register is in CHN=3 (Galois/counter mode) chaining mode, it is likely to happen that encryption/decryption computation is completed (PDFS=1) while the crypto-processor computation is busy (PBS=1). In this case, it suggests that encryption/decryption operation is completed but Galois hash computation is going on. In other chaining modes, this register is the complement of the encryption/decryption computation complete interrupt stage.
Bit 2	WEFS	0x0	ro	Write error flag status 0: No write error is detected 1: Write error is detected This register is cleared by writing 1 to the WEFC bit in the AES_FCLR register.
Bit 1	REFS	0x0	ro	Read error flag status 0: No read error is detected 1: Read error is detected This register is cleared by writing 1 to the REFC bit in the AES_FCLR register.
Bit 0	PDFS	0x0	ro	Encryption/Decryption processing done flag status 0: Encryption/Decryption computation is not completed 1: Encryption/Decryption computation is completed This register is cleared by writing 1 to the PDFC bit in the AES_FCLR register.

28.7.3 AES data input register (AES_IDT)

Bit	Name	Reset value	Type	Description
Bit 31: 0	IDT	0x00000000	wo	Input Data Port This register is used for data input using CPU or DMA.

28.7.4 AES data output register (AES_ODT)

Bit	Name	Reset value	Type	Description
Bit 31: 0	ODT	0x00000000	wo	Output Data Port This register is used to obtain data using CPU or DMA.

28.7.5 AES key register 0 (AES_KEY0)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY0	0x00000000	rw	Key W0 This field contains the bit [31:0] of the AES encryption or decryption kdy.

28.7.6 AES key register 1 (AES_KEY1)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY0	0x00000000	rw	Key W1 This field contains the bit [63:32] of the AES encryption or decryption kdy.

28.7.7 AES key register 2 (AES_KEY2)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY2	0x00000000	rw	Key W2

				This field contains the bit [95:64] of the AES encryption or decryption kdy.

28.7.8 AES key register 3 (AES_KEY3)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY3	0x00000000	rw	Key W3 This field contains the bit [127:96] of the AES encryption or decryption kdy.

28.7.9 AES initialization vector register 0 (AES_IV0)

Bit	Name	Reset value	Type	Description
Bit 31: 0	IV0	0x00000000	rw	Initialization vector register 0 初始向量第 0-31 比特位 This field contains the bit [31:0] of the initialization vector.

28.7.10 AES initialization vector register 1 (AES_IV1)

Bit	Name	Reset value	Type	Description
Bit 31: 0	IV1	0x00000000	rw	Initialization vector register 1 This field contains the bit [63:32] of the initialization vector.

28.7.11 AES initialization vector register 2 (AES_IV2)

Bit	Name	Reset value	Type	Description
Bit 31: 0	IV2	0x00000000	rw	Initialization vector register 2 This field contains the bit [95:64] of the initialization vector.

28.7.12 AES initialization vector register 3 (AES_IV3)

Bit	Name	Reset value	Type	Description
Bit 31: 0	IV3	0x00000000	rw	Initialization vector register 3 This field contains the bit [127:96] of the initialization vector.

28.7.13 AES key register 4 (AES_KEY4)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY4	0x00000000	rw	Key W4 This field contains the bit [159:128] of encryption or decryption key. This field is effective only when the key length is configured as 192 bit or 256 bit (KL=1 or KL=2)

28.7.14 AES key register 5 (AES_KEY5)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY5	0x00000000	rw	Key W5 This field contains the bit [191:160] of encryption or decryption key. This field is effective only when the key length is configured as 192 bit or 256 bit (KL=1 or KL=2)

28.7.15 AES key register 6 (AES_KEY6)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY6	0x00000000	rw	Key W6 This field contains the bit [223:192] of encryption or decryption key. This field is effective only when the key length is configured as 256 bit KL=2)

28.7.16 AES key register 7 (AES_KEY7)

Bit	Name	Reset value	Type	Description
Bit 31: 0	KEY6	0x00000000	rw	Key W7 This field contains the bit [255:224] of encryption or decryption key. This field is effective only when the key length is configured as 256 bit KL=2)

28.7.17 AES suspend message register 0 (AES_SI0)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI0	0x00000000	rw	Suspend information 0 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.18 AES suspend message register 1 (AES_SI1)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI1	0x00000000	rw	Suspend information 1 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.19 AES suspend message register 2 (AES_SI2)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI2	0x00000000	rw	Suspend information 2 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.20 AES suspend message register 3 (AES_SI3)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI3	0x00000000	rw	Suspend information 3 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.21 AES suspend message register 4 (AES_SI4)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI4	0x00000000	rw	Suspend information 4 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.22 AES suspend message register 5 (AES_SI5)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI5	0x00000000	rw	Suspend information 5 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.23 AES suspend message register 6 (AES_SI6)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI6	0x00000000	rw	Suspend information 6 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.24 AES suspend message register 7 (AES_SI7)

Bit	Name	Reset value	Type	Description
Bit 31: 0	SI7	0x00000000	rw	Suspend information 7 This field is effective only in CHN=3 (Galois/counter mode) or CHN=4 (Counter mode with cipher block chaining message authentication code).

28.7.25 AES flag clear register 7 (AES_FCLR)

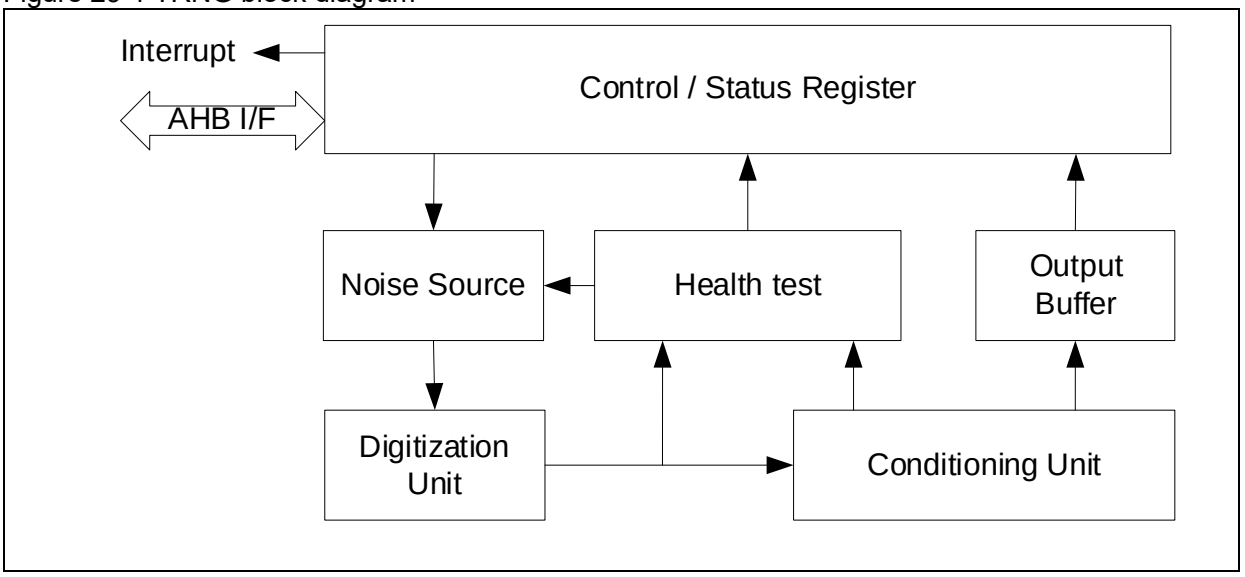
Bit	Name	Reset value	Type	Description
Bit 31: 5	Reserved	0x00000000	resd	Kept at default value.
Bit 4	NZDFC	0x0	wo	Non-zero dummy data processed flag clear The NZDFS flag of the AES_STS register is cleared by writing 1 to this bit.
Bit 3	Reserved	0x0	resd	Kept at default value.
Bit 2	WEFC	0x0	wo	Write error flag clear The WEFS flag of the AES_STS register is cleared by writing 1 to this bit.
Bit 1	REFC	0x0	wo	Read error interrupt clear The REFS flag of the AES_STS register is cleared by writing 1 to this bit.
Bit 0	PDFC	0x0	wo	Encryption/Decryption processing done interrupt clear The PDFS flag of the AES_STS register is cleared by writing 1 to this bit.

29 True random number generator (TRNG)

29.1 Introduction

The RNG is a true random number generator designed for security applications. Its entropy source design and detection is in accordance with the NIST SP800-90B standard. Therefore the TRNG is able to generate the best entropy value under a minimum algorithm, and keep monitoring it through a health check mechanism, so as to ensure the unpredictability of the generated content. It produces 128-bit random data every clock cycle and it has an AHB bus to access 32-bit random numbers each time.

Figure 29-1 TRNG block diagram



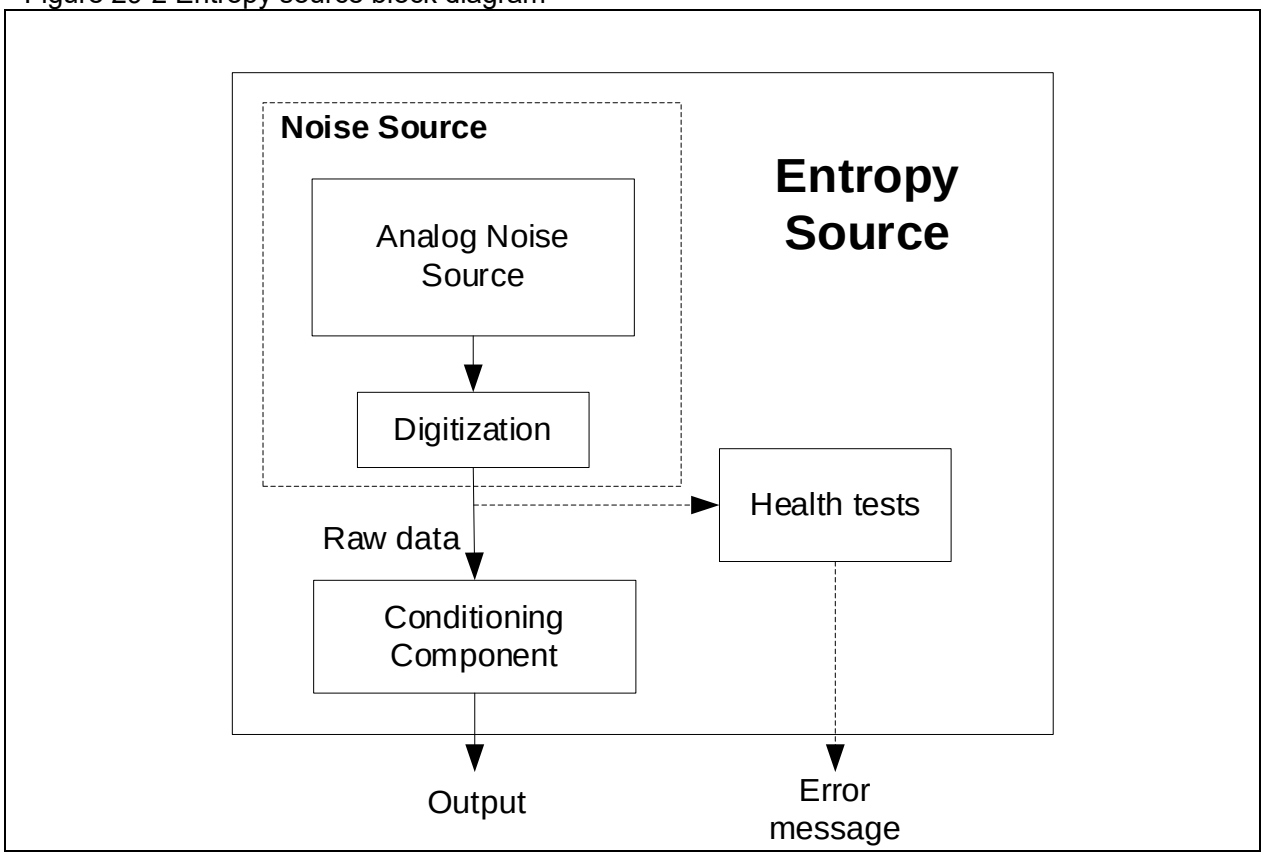
29.2 TRNG main features

- Compliance with the NIST SP800-90B standard
- Embedded health test mechanism
  - Repetition count tests on noise source
  - Adaptive proportional tests on noise source
  - Known answer tests on the conditioning components
- Helath checks on on program run
  - Repetition count tests on noise source
  - Adaptive proportional test on noise source
- Helath checks under specific conditions
  - Repetition tests on the noise source
  - Clock tests on the noise source
- In-circuit test with error flag and interrupt signals
- Noise source automatic reset to eliminate the non-performing random numbers
- Use conditioning components that are validated according to the NIST SP800-90B standard for noise source post-processing
- Access 32-bit random numbers via an AHB
- 128-bit random numbers generated every clock cycle

29.3 Entropy source

The entropy source is viewed as the main component of the TRNG strucutre. Under the NIST SP800-90B standard, it includes the noise source, health test and the conditioning component, as showin in Figure 29-2.

Figure 29-2 Entropy source block diagram



29.3.1 Noise source

The noise source is the basis of a TRNG. Analog noise sources procuded from mutliple free-running ring oscillators are converted via a digitization stage into a bitstring output, which is called Raw data. So the noise source provides the unpredictability necessary for the TRNG.

29.3.2 Conditioning component

The output of the noise source may fluctuate due to statistical variations. The conditioning component is aimed to improving this fluctuating problem. The TRNG uses the CBC_MAC algorithm, which is validated according to the NIST SP800-90B standard, in conjunction with AES peripheral as its conditioning component. The conditioning component processes 128-bit data on each single access. All the processed data are stored into output buffer with the DTRDY set in the TRNG_STS register.

29.3.3 Health test mechanism

This component is designed to monitor the normal operation of the entire entropy source and prevent it from malicious affect from external factors, so as to make sure the reliability of the random number generated. Health test results are indicated by corresponding error flags and interrupt signals which are then used to block noise source outputs. Under the NIST SP800-90B, there are three forms of health tests in all.

Initialization health test:

In order to ensure the normal operation of circuitry, it is necessary to perform initialization health tests before microcontroller power-on, the RNT_RESET asserted in the CRM register or the initial use of the TRNG. Thus it is not possible for users to obtain random numbers through data register (reading the data register returns all zeros only) until the initalzation health test is passed. The health tests are as follows:

- Repetition count test on the noise source
  The repetition count test is to help confirm whether the bitstring output from the noise source gets stucked at a constant location. When the noise source has delivered more than 42 consecutive bits at a constant value (“0” or “1”), the noise source is reset and a health test is re-started. After the noise source is reset, if it has delivered more than 42 consecutive bits at “0” or “1”, the health

test mechanism decides that the noise source gets stuck and then sets the ESFES bit in the TRNG_STS register accordingly.

- Adaptive proportional test on the noise source
  The Adaptive porpoirtion test is used to judge whether the entropy value generated by the noise source is damaged or not.

  When a noise source is being affected with unexpected events, it may deliver an extremely large amount of “0” or “1”. If taking the first 1024 bits out for the noise source output results for analysys, when the total number of “0” or “1” is over 690 bits, it is decided that the noise source has insufficient entropy values and sets the the ESFES bit in the TRNG_STS register accordingly.
- Known answer test on the conditioning component
  The known answer test is aimed at confirming whether the conditioning component is running normal. The health test mechanism inputs three data with constant values and then compares the output of the conditioning component with the pre-stored result. If no matching between them, it is decided that the conditioning component failures and sets the the ESFES bit in the TRNG_STS register accordingly.

Programm run test

In order to prevent TRNG from external impact during runtime, after TRNG gets started, the health test mechanism keepts monitoring the noise source. The content to test is similar to the initialization test, once upon an error detection, the health test mechanism stops TRNG and discards the random numbers that are generated during this period of time.

- Repetition count test on the noise source
  The repetition count test is used to keep monitoring the noise source during TRNG runtime. When the noise source has delivered more than 42 consecutive bits at a constant value (“0” or “1”), the noise source is reset and the generated raw data are discarded. After the noise source reset, if it has delivered more than 42 consecutive bits at “0” or “1”, the health test mechanism decides that the noise source gets stuck and then sets the ESFES bit in the TRNG_STS register accordingly.
- Adaptive proportional test on the noise source
  During TRNG runtime, the health mechanism is responsible for monitoring the 1024 bits in total that are generated by the noise source during its previous and current output period.

  when the total number of “0” or “1” is over 690 bits, it is decided that the noise source has insufficient entropy values and sets the the ESFES bit in the TRNG_STS register accordingly.

Specific condition test

In addition, there are several other test items to allow for a precise evaluation of the noise source.

- Repetition toggle test on the noise source
  The health mechanism keeps monitoring the noise source during TRNG runtime.

  When the noise source has delivered more than 21 consecutive bits at a constant value (“01” or “10”), the noise source is reset and the generated raw data are discarded. After the noise source reset, if it has delivered more than 21 consecutive bits at “01” or “10”, the health test mechanism decides that the noise source error occurs and then sets the ESFES bit in the TRNG_STS register accordingly.
- Clocking test on the noise source
  The health mechanism monitors the original clock of the noise source based on a AHB/32 freuqency. When there is not any toggle action within three clock cycles, the clock of the noise source is deviced as abnormal, TRNG stops and the CKFES bit in the TRNG_STS is set.

29.4 Status and interrupts

The TRNG_STS register is used to save TRNG status and error events. Interrupt events are sent to CPU by setting the DTRIE bit and FIE bit in the TRNG_CTRL register. The TRNG uses global interrupts to connect to CPU NVCI, so the user needs to obtain interrupt details by reading the registers below:

Data port ready

This register is used to obtain the random data that are available in the TRNG output buffer. When NSEN=1 (noise source is enabled) and CUDIS=0 (output buffer is enabled), TRNG gets started generating random numbers. After the completion of the random number generation and data are stored into output buffer , this DTRDY bit is set. After all four 32-bit random data are obtained out of the output buffer, this DTRDY bit is reset. if DTRIE is enabled, an interrupt is issued to CPU when DTRDY bit is set.

Clock failure status

The CKFES bit is used to indicate the clock status of the noise source. When a noise source clock error is detected, this bit is set. When this bit is set, please refers to the section 29.5.3 for error reset procedure. If CKFES bit remains set while TRNG is reset, it indicates that the clock of the noise source is being damaged and cannot be recovered. When the CKFES bit is set, the noise source stops generating raw data.

Entropy source failure status

The ESFES bit indicates the status of the entropy source. The health mechanism decides Entropy source error and sets the ESFES bit in the event of the events below:

- Noise source repetition count test failed in noise source initialization test and program run test
- Noise source adaptive proportion test failed in initialization test and program run test
- Noise source repetition test failed in program run test
- Known answer test of the conditioning component fialured in intializatio test

If this bit is set, please refers to section 29.5.3 for details on error status reset procedure. If the ESFES bit remains set while TRNG is reset, it indicates that the entropy source is damaged and cannot be recovered. When the ESFES bit is set, the noise source stops generating raw data.

If the FIE is enabled, the ESFIS bit is also set upo an entropy source error and an interrupt is generated and issued to CPU. This ESFES bit is cleared by writing 1.

29.5 TRNG configurations

29.5.1 TRNG operation procedure

After TRNG is enabled, it starts to run in two stages.

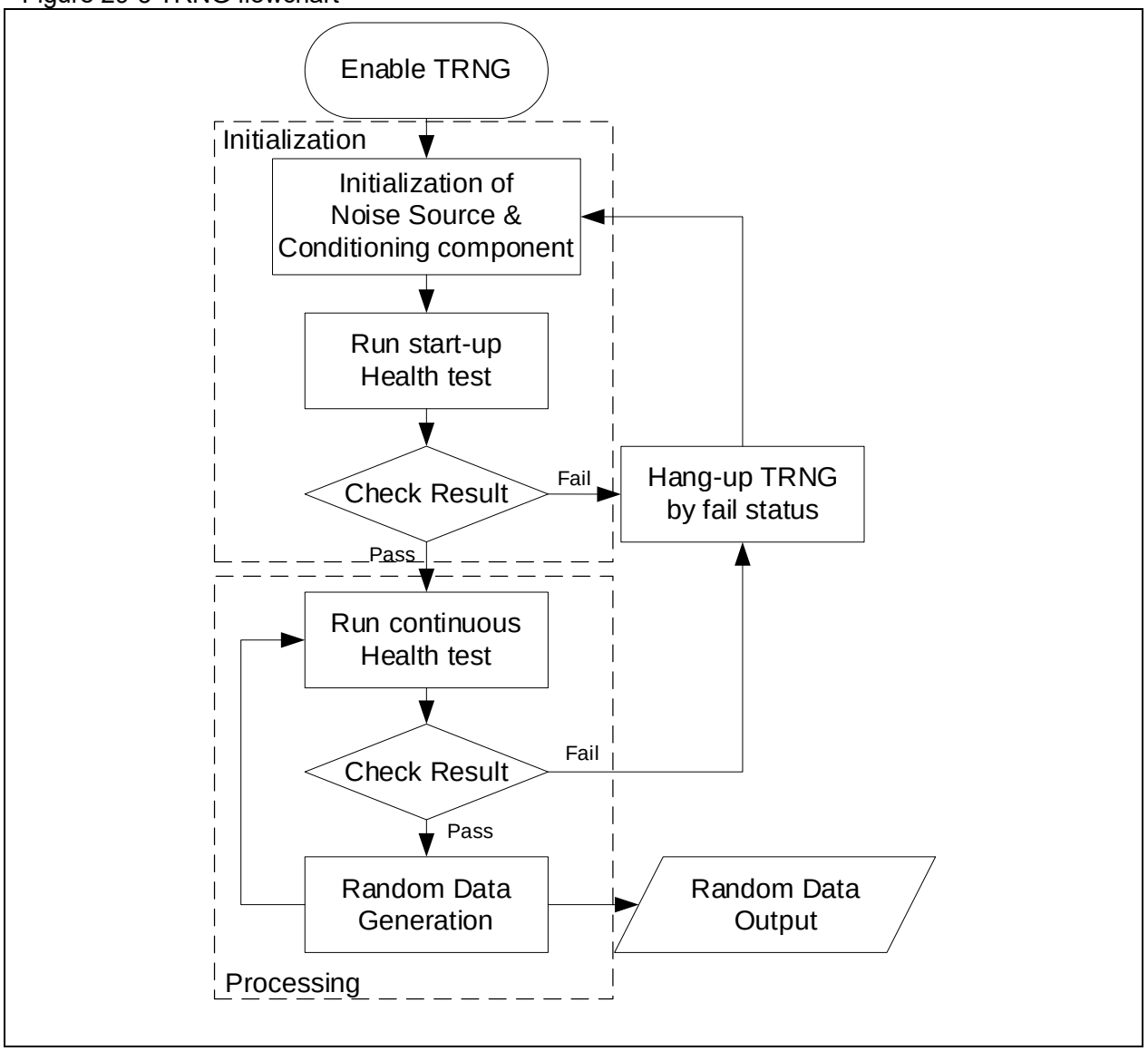
Initialization

When NSEN=1 (noise source enabled) and CUDIS=0 (conditioning component and output buffer enabled), the noise source and the conditioning component is being initialized and the TRNG is enabled. In order to ensure normal operation of circuity, the health test mechanism starts monitoring the initialization process following the completion of the initialization of the noise source and the conditioning component. Under the NIST SP800-90B standard, the bitstring generated by the noise source are not to be used during the initialization testing period. If the test failed, an error flag is set and the output of the noise source is halted.

Processing

After going through the initialization test, the TRNG starts run and generates random numbers while the health test mechanism is responsible for monitoring program run. Upon a noise source error is detected, the raw data thart are being processed via the conditioning component will be discarded, the TRNG stops, so as to avoid the user to obtain wrong raw data. During TRNG normal operation, it can deliver four 32-bit random data every around 450 AHB clock cycles. The clock cyles for generating random data can be extended by setting the CLKDIV bit in the TRNG_CTRL register.

Figure 29-3 TRNG flowchart



29.5.2 TRNG configurations

Follow the steps below to configure TRNG:

1. Set the CUDIS bit to 1 in the TRNG_CTRL register, and configure TRNG (ex. CLKDIV)
2. If needed for interrupts, set the DTRIE and FIE in the TRNG_CTRL register,
3. Set CUDIS=0 in the TRNG_CTRL register, and enable TRNG by setting NSEN=1
4. Wait until the DTRDY is set in the TRNG_STS register, and confirm whether there are error flags to be set in the following ways:
   - If DTRIE=1, wait until the NVIC of the CPU receives an interrupt signal, check if ESFIS and CKFIS are set or not, and whether the DTRDY is set or not
   - If DTRIE=0, the user can read the TRNG_STS register at certain intervals to check if ESFIS and CKFIS are set or not, and whether the DTRDY is set or not
5. Check that the DTRDY is set and without error, it indicates that the TRNG has completed the generation of random data and stored them into the output buffer. In this case, if an interrupt control bit is already enabled, please disable it.
6. When DTRDY is set, it is possible to obtain a 32-bit random data by reading the data register. After four 32-bit random payload are obtained out of the output buffer, the DTRDY is reset automatically.
7. Return to step 2 to start the next cycle.

29.5.3 Error flag reset procedure

TRNG-generated random data are widely used in sensitive scenarios such as secret key or security applications, so it is imperative to guarantee he unpredictability of the content generated.

The TRNG hardware is designed to equip a health test mechanism which sets CKFES and ESFES bits upon an error and then stops a noise source to ensure normal operation and prevent any intentional or unintentional effect from external factors. Some effects may be temporary, and some are persistent. So when an error falg is set, the user needs to follow a procedure to reset and confirm whether this error can be cleared, proceed as follows:

1. Set FESCLR to 1 in the TRNG_CTRL register and confirm whether CKFES or ESFES is cleared or not. If not yet, perform step 2. If yes, the TRNG will re-start the initialization test and resume normal operation
2. Set NSEN=0 and CUDIS=1 to disable TRNG, and then set NSEN=1 and CUDIS=0 to enable TRNG. Check if CKFES or ESFES is cleared or not. If yes, the TRNG will re-start the initialization test and resume normal operation
3. If the error flag cannot be cleared through step 1 and step 2, it indicates that TRNG is still being affected and not able to generate random data. in this case, it is recommened to stop the use of the TRNG.

29.6 TRNG registers

These peripheral registers have to be accessed by words (32 bits).

Table 29-1 TRNG register map and reset values

Register	Offset	Reset value
TRNG_CTRL	0x000	0x0020 0000
TRNG_STS	0x004	0x0000 0000
TRNG_DT	0x008	0x0000 0000
TRNG_HTCTRL	0x00C	0x0004 2ce8

29.6.1 TRNG control register (TRNG_CTRL)

Bit	Name	Reset value	Type	Description
Bit 31	PRGLK	0x0		Programing register lock 0: TRNG_HTCTRL[19:0] an TRNG_CTRL[28:5] can be configured 1: TRNG_HTCTRL[19:0] an TRNG_CTRL[28:5] cannot be configured This bit is set by writing 1. It is not allowe to be cleared by writing 0. If there is a need to unlock this registe reset TRNG through the RNG_RESET b in the CRM register.
Bit 30	CUDIS	0x0	rw	Conditioning unit and output buffer disable 0: Conditioning unit and output buffer enable 1: Conditioning unit and output buffer disable TRNG_HTCTRL[19:0] and TRNG_CTRL[28:5] can be configured only when CUDIS=1).
Bit 29	FESCLR	0x0	wo	Clock fail event and entropy source fail event status clear The CKFES and ESFES in the TRNG_STS register can be cleared by writing 1 to FESCLR bit.
Bit 28:22	Reserved	0x0	resd	Kept at default value.
Bit 21:20	HT0	0x2	rw	Health test parameter 0 Kept at default value.

Bit 19:16	CLKDIV	0x0	rw	Noise source clock divider factor 0: Orignial clock frequency 1: Orignial clock/2 2: Orignial clock/4 3: Orignial clock/8 4: Orignial clock/16 5: Orignial clock/32 … 15: Orignial clock/32768
Bit 15：5	Reserved	0x00	resd	Kept at default value.
Bit 4	DTRIE	0x0	rw	Data port ready interrupt enable 0: Data port ready interrupt disable 1: Data port ready interrupt enable
Bit 3	FIE	0x0	rw	Fail interrupt enable 0: Fail interrupt disable 1: Fail interrupt enable
Bit 2	NSEN	0x0	rw	Noise source enable 0: Noise source disable 1: Noise source enable TRNG_HTCTRL[19:0] and TRNG_CTRL[28:5] can be configured only when NSES=1
Bit 1:0	Reserved	0x0	resd	Kept at default value.

29.6.2 TRNG status register (TRNG_STS)

Bit	Name	Reset value	Type	Description
Bit 31:7	Reserved	0x0000000	resd	Kept at default value.
Bit 6	ESFIS	0x0	rw1c	Entropy source fail interrupt status 0: Entropy source under normal operation 1: Entropy source error interrupt generated This bit is cleared by writing 1.
Bit 5	CKFIS	0x0	rw1c	Clock fail interrupt status 0: Noise source clock under normal operation 1: Noise source clock error interrupt generated This bit is cleared by writing 1.
Bit 4:3	Reserved	0x0	resd	Kept at default value.
Bit 2	ESFES	0x0	ro	Entropy source fail event status 0: Entropy source under normal operation 1: Entropy source error detected
Bit 1	CKFES	0x0	ro	Clock fail event status 0: Noise source clock under normal operation 1: Noise source clock error detected
Bit 0	DTRDY	0x0	ro	Data port ready 0: Random data generation is not completed 1: Random data generation is completed

29.6.3 TRNG data register (TRNG_DT)

Bit	Name	Reset value	Type	Description
Bit 31:0	DT	0x00000000	ro	Data Port When DTRDY = 1, it is possible to obtain a 32-bit random data by reading this register. When DTRDY = 0, reading this register returns all zeros.

29.6.4 TRNG health test control register (TRNG_HTCTRL)

Bit	Name	Reset value	Type	Description
Bit 31:20	Reserved	0x00000	resd	Kept at default value.
Bit 19:0	HT1	0x42ce8	ro	Health test parameter 1

30 Qud-SPI interface (QSPI)

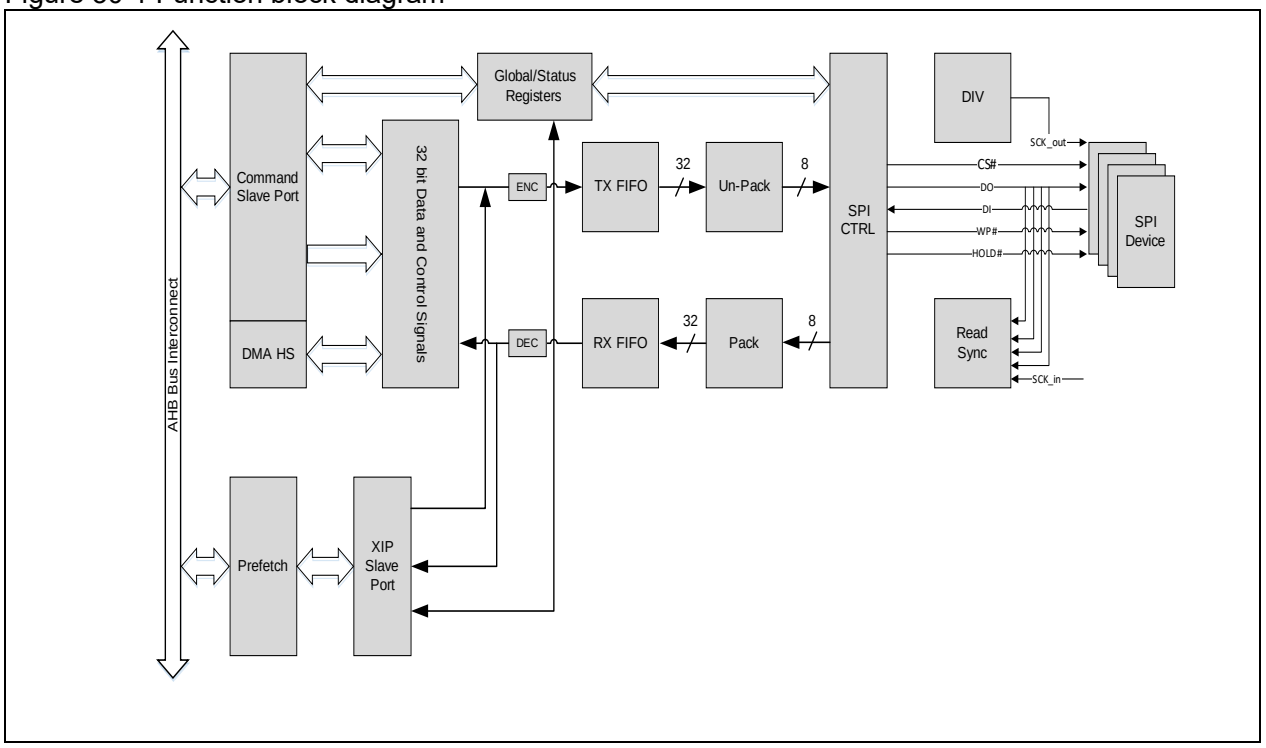
30.1 Introduction

The QSPI interface consists of a command-based slave port, an XIP port (direct address mapping access) and QSPI interface controller used for SPI Flash command execution. The command-based slave port is used to access registers and data ports, and the XIP slave port reads data from direct address mapping. Additionally, the QSPI allows AHB data port to access data in either PIO or DMA mode.

30.2 QSPI main features

- DMA handshake and CPU PIO modes
- SPI mode, dual/quad output mode, dual/quad I/O mode and DPI/QPI mode
- XIP port (direct address mapping read/write)
- XIP port prefetch function (2-channel read cache)
- 128-byte TxFIFO/RxFIFO depth
- Programmable divider
- Data encryption

Figure 30-1 Function block diagram



30.3 QSPI command slave port

30.3.1 QSPI command slave port

The QSPI has a command slave interface that contains register and data ports. The users can access registers or data ports using this interface. The command word register must be written sequentially (CMD_W3 is the last to be written) and is accessible in words, while any other registers (including data port register) can be accessed by bytes, half-words and words.

30.3.2 CPU PIO mode

The data port can be accessed in either PIO or DMA mode. The RxFIFO / TxFIFO registers (0x18) must be polled in PIO mode. When the RxFIFO is ready, the user can read or store the entire RxFIFO data. When the TxFIFO is ready, it can be written with the entire data. The polling state must be guaranteed in PIO mode.

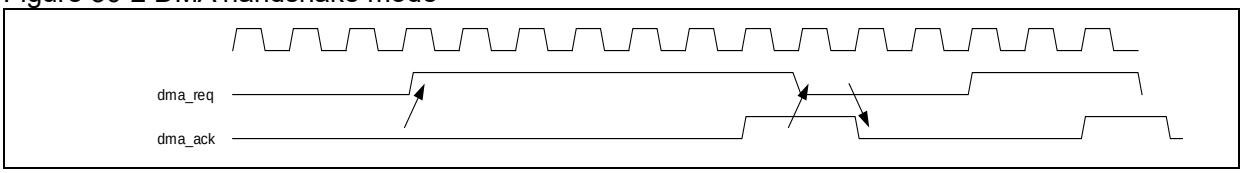
30.3.3 DMA handshake mode

The DMA mode can also be used to access data ports. The DMA controller register must be programmed as DMA handshake mode. In this mode, the host controller sends a DMA request when the receive/transmit FIFO threshold is reached. The threshold values are in terms of words. The DMA request is still sent when the last data transfer is less than the threshold.

When DMA is used in P2M mode (peripheral to memory), that is, when QSPI reads data, the minimum transmit bit is in WORD. Thus DMA length needs to be in the format of WORD unit, and so does the data size.

When 2 DMA are used in M2P mode (memory to peripheral), that is, when QSPI writes data, the maximum length is 128 bytes for a single transfer. If more than 128 bytes, it is necessary to perform multiple DMA transfers, in the format of Byte unit.

Figure 30-2 DMA handshake mode



30.3.4 XIP port (direct address mapping read/write)

The QSPI offers a XIP slave port (direct address mapping read/write) for users to perform a direct access to system addresses. In this case, only 0x10 register can be used to select a command slave port or a XIP port, or read other registers than data register (0x100). It should be noted that when reading Flash memory using this XIP port, the Abort function (bit [8] in the 0x10 register) cannot be used (meaning that the bit [20] of the 0x10 register is set to 1), and the XIP port cannot be aborted with this bit, which means that port switch must be performed in sequence. The user's system must have a main Flash memory, in which the user can load the port switch code and perform switching operation.

30.3.5 XIP port prefetch

The XPI port supports read prefetch with 2-channel read cache.

30.3.6 SPI device operation

Command phase

The QSPI operations are conducted through commands, which are used to indicate the intent of the operation and SPI operating mode (multiple-line communication).

Note: the command phase can be set to 1 or 2 bytes. However, in 2-byte mode, it sends two repeated 1-byte commands. Only 1-byte register is used to store command words.

Address phase

The address phase is required to perform read, write, and erase operations on the memory.

In the address phase, the address phase can be from 0 to 4byte in length; In XIP mode, the length of the address phase can be optional 3byte or 4byte.

Dummy-cycles phase

It is used to allow the to-be-read device the time to prepare for the subsequent to-be-read data. XIP and command mode must be configured through DUM2 and XIPR_DUM2, respectively. Besides, the write timing of XIP mode also reserves programmable dummy cycles, which is usually configured to 0 through the XIPW_DUM2 register.

Data phase

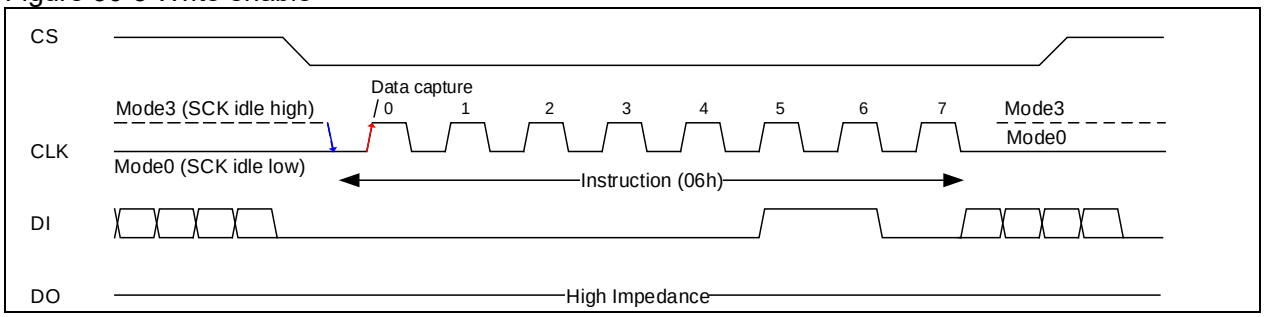
It is used as input when reading memory device data/status registers, and as an output when writing the memory device.

Serial (1-1-1) mode

The host controller supports a SPI protocol (mode 0 and mode 3). The command queue register must be programmed according to SPI device specification. Refer to Figure 30-3 and Figure 30-10 for more information on how to program a command queue.

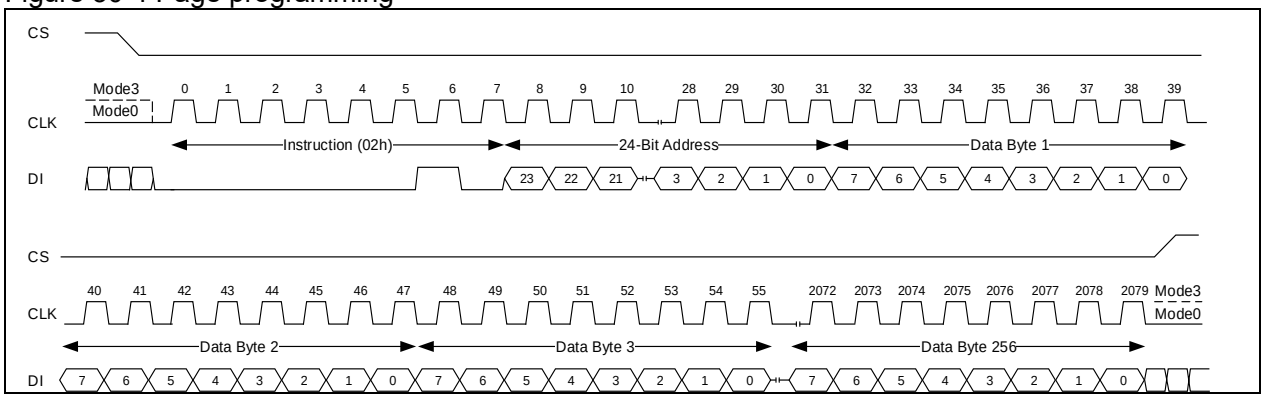
To execute a write enable command, set instruction code to 06h, write enable, and set instruction length to 1. Refer to Figure 30-3 for details.

Figure 30-3 Write enable



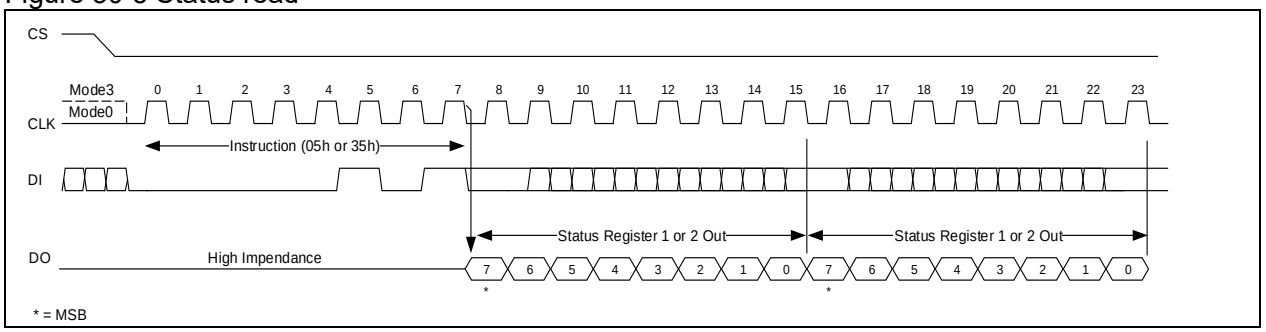
To execute page programming command, set instruction code to 02h, address register, address size, write enable and set instruction length to 1.

Figure 30-4 Page programming



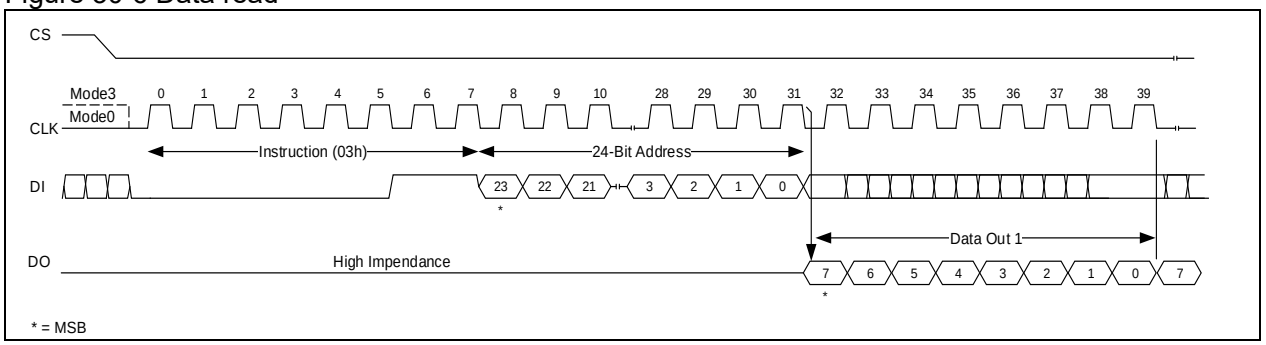
To execute a read status command, set instruction code to 05h/35h, enable read status, select read status through hardware or software, and set instruction length to 1. Refer to Figure 30-5 for more information.

Figure 30-5 Status read



To execute a read data command, set instruction code and length to 1 byte, address/address size to 3 bytes, and set write instruction to 0. Refer to Figure 30-6 for more information.

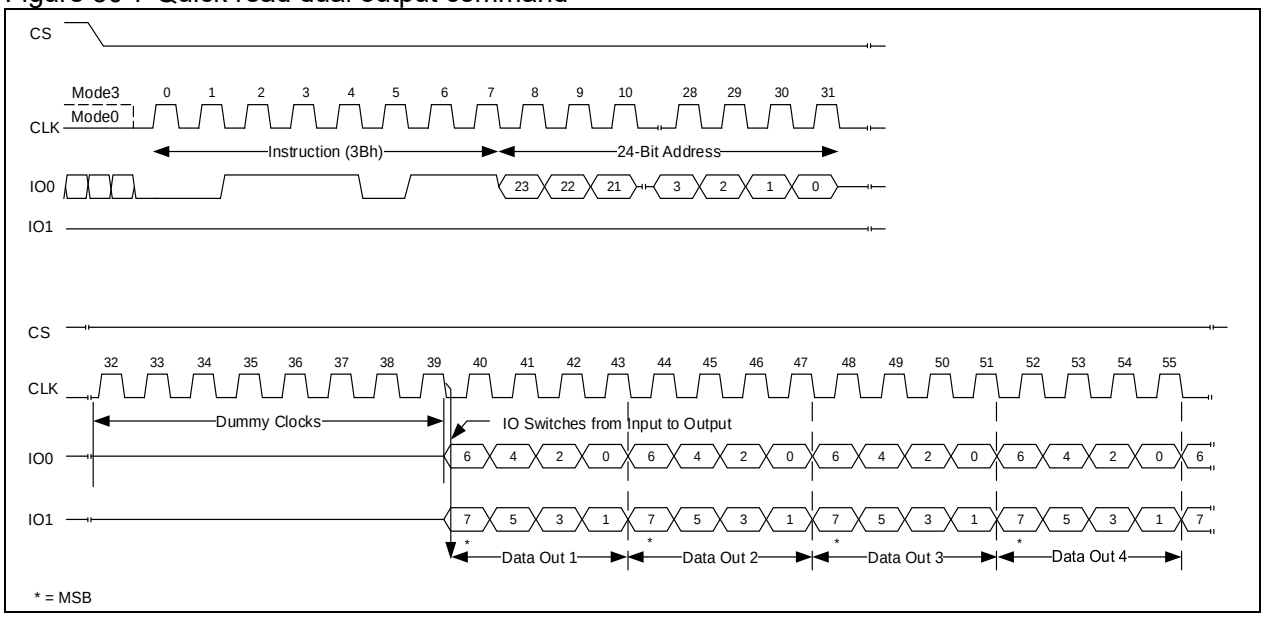
Figure 30-6 Data read



Dual (1-1-2) mode

To execute a quick read dual output command, set instruction code/length to 1, address/address size to 3 bytes, set the second dummy cycle to 8, and enable dual mode. Refer to Figure 30-7 for details.

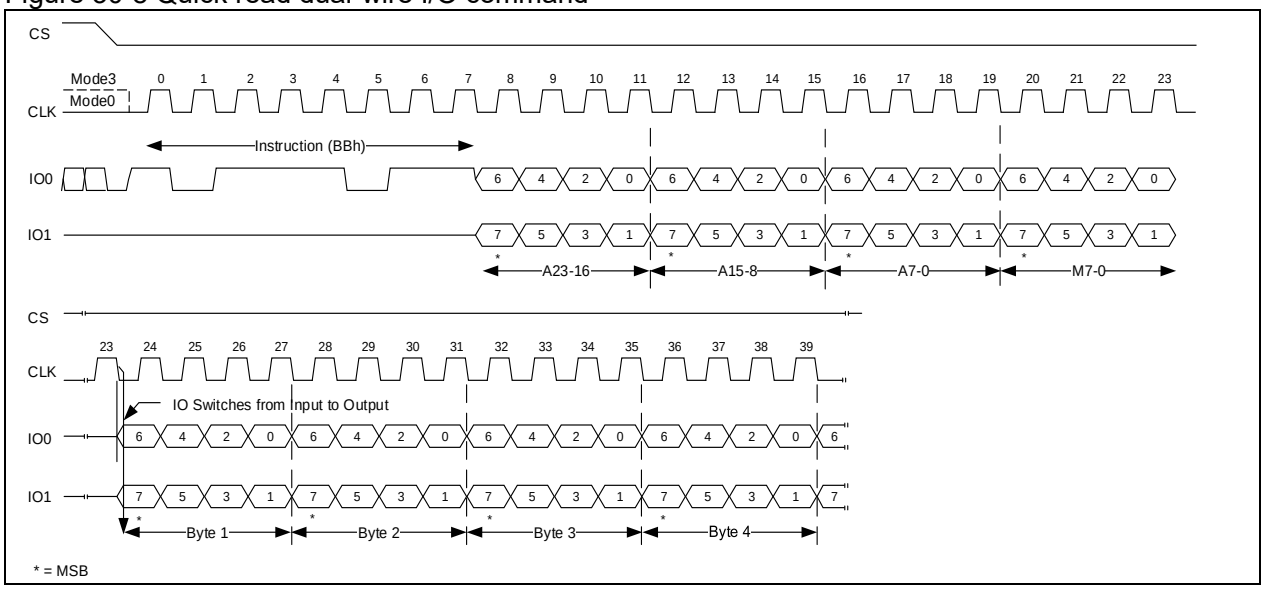
Figure 30-7 Quick read dual output command



Dual I/O (1-2-2) mode

To execute a quick read dual I/O command, set code/ length, address/address size, enable enhanced performance mode/code and dual I/O operation commands. Refer to Figure 30-8 for details.

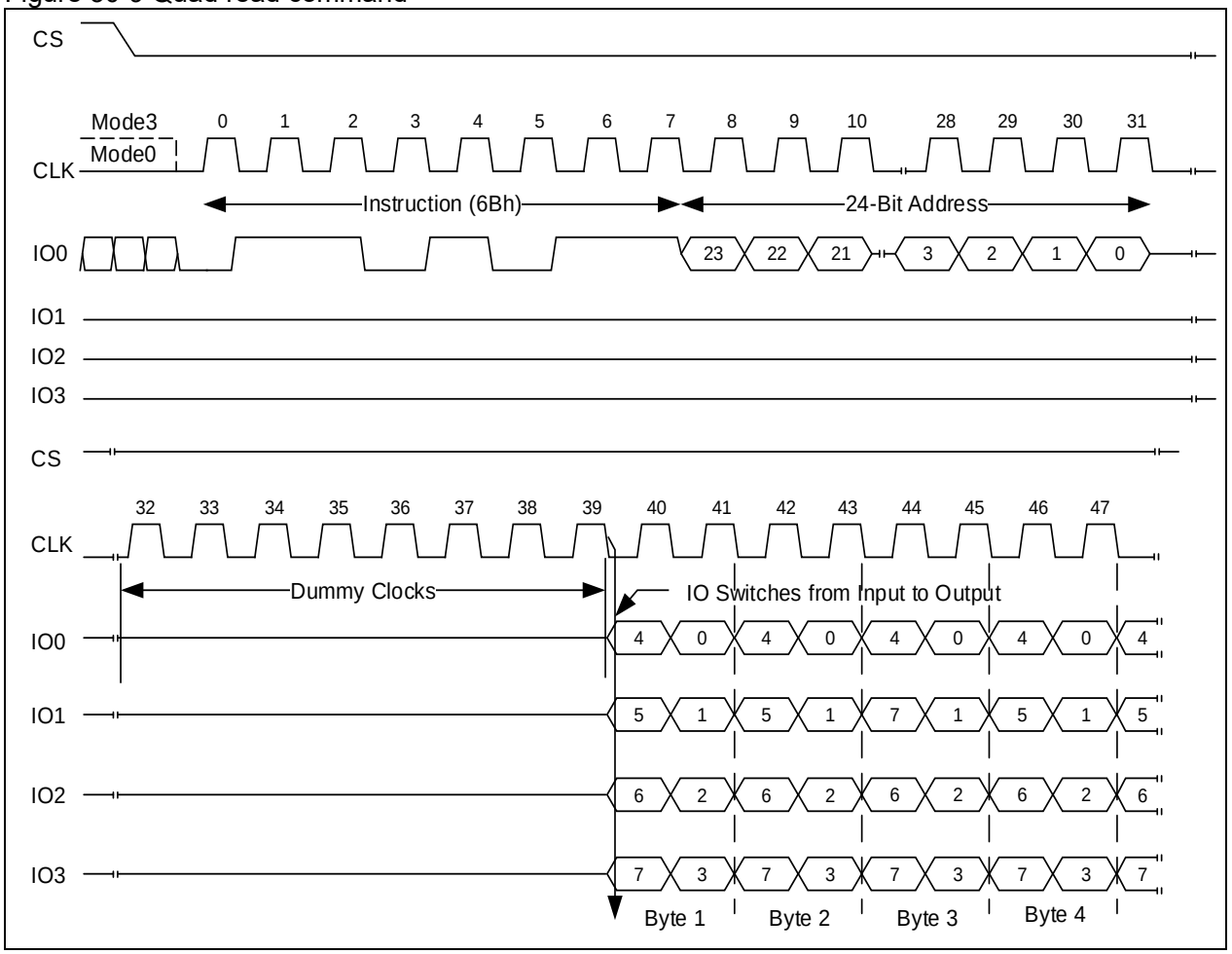
Figure 30-8 Quick read dual-wire I/O command



Quad (1-1-4) mode

To execute a quick read quad command, set instruction code/length, address/address size, enable second dummy period and enable quad mode. Refer to Figure 30-9 for more information.

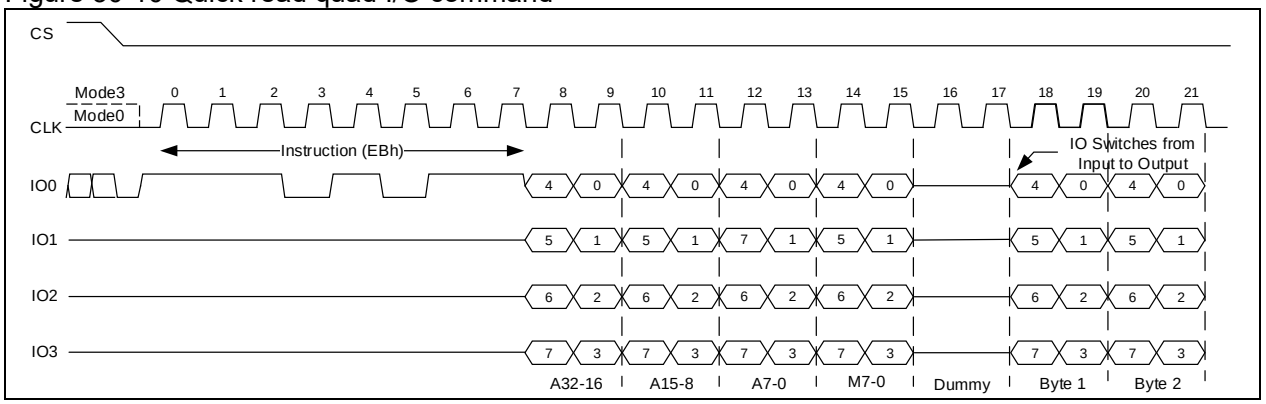
Figure 30-9 Quad read command



Quad I/O (1-4-4) mode

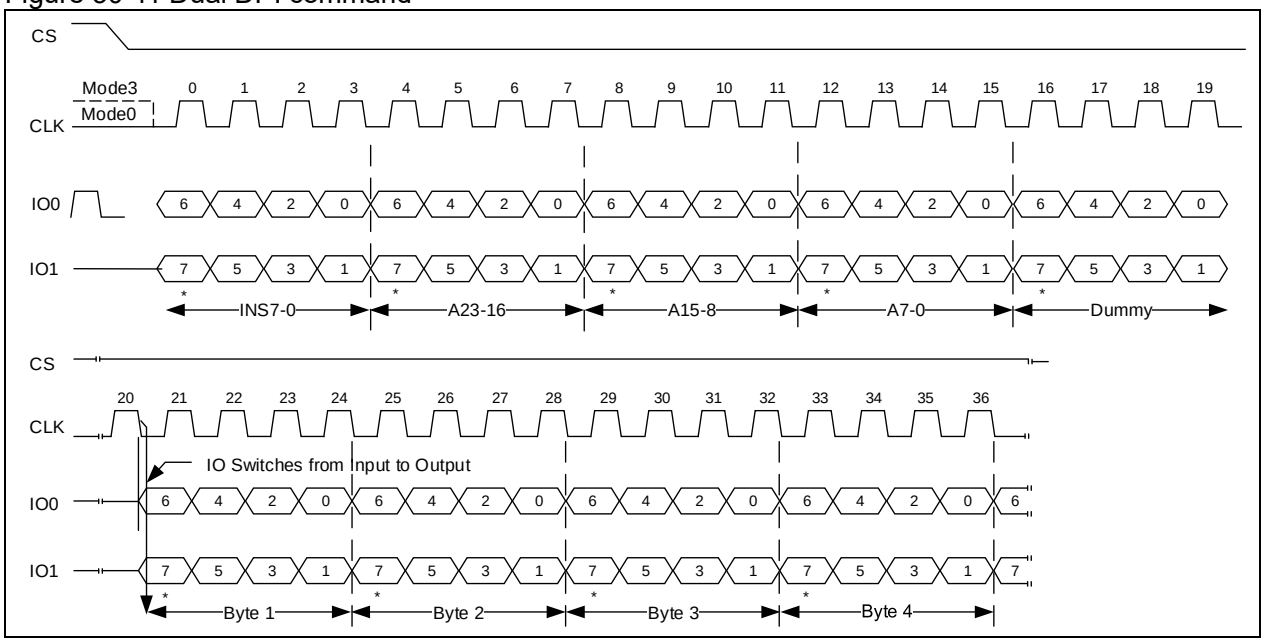
To execute a quick read quad I/I command, set instruction code/length, address/address size, the second dummy period, enable performance enhancement mode and enable quad I/O mode. Refer to Figure 30-10 for more information.

Figure 30-10 Quick read quad I/O command



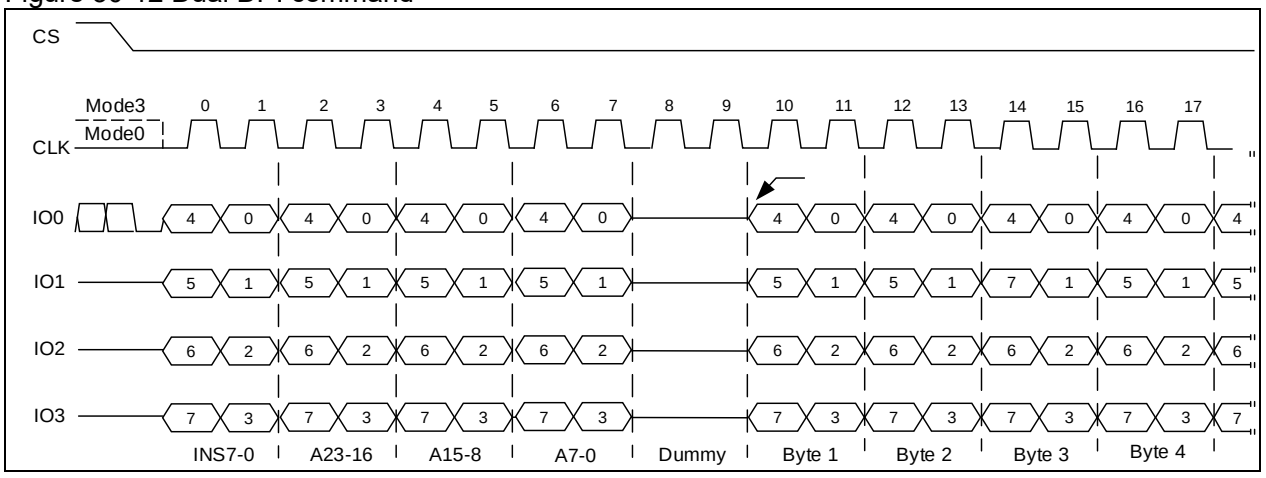
If dual DPI (2-2-2) mode is to be used, it is necessary to set code/length, address/address length, the second dummy cycle and operation command of dual DPI mode. For details, refer to the figure below.

Figure 30-11 Dual DPI command



If quad DPI (4-4-4) mode is to be used, it is necessary to set code/length, address/address length, the second dummy cycle and operation command of quad DPI mode. For details, refer to the figure below.

Figure 30-12 Dual DPI command



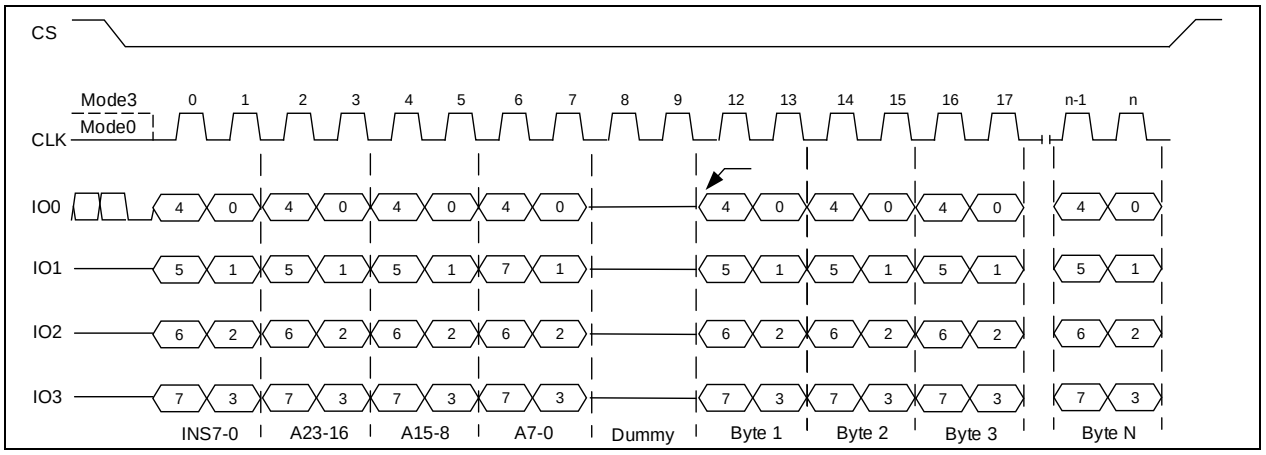
XIP read/write D mode

If XIP read/write D mode is to be used, it is necessary to set code/length, address/address length, the second dummy cycle, select serial/two-line/two-line IO/four-line/four-line IO or DPI/QPI mode, and to set read/write D mode and D mode upper limit counter.

When XIP port is reading or writing back-to-back addresses, it wil keep reading or writing until the higher threshold is reached or the address becomes discontinuous.

When reading, N(byte) = XIPR_DCNT*8; when writing, N(byte) = XIPW_DCNT*8.

For more details, refer to the figure below.



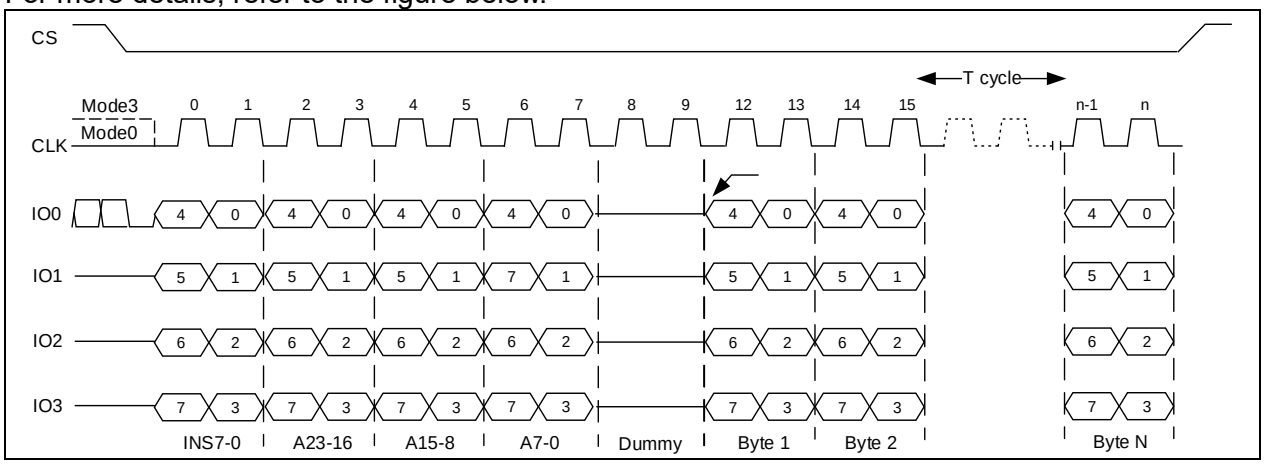
XIP read/write T mode

If XIP read/write T mode is to be used, it is necessary to set code/length, address/address length, the second dummy cycle, select serial/two-line/two-line IO/four-line/four-line IO or DPI/QPI mode, and to set read/write T mode and T mode upper limit counter.

When XIP port is reading or writing back-to-back addresses at intervals, it wil disable CLK wait aftr reading or writing data. If the wait duration is longer than the threshold or when the next address is discontinuous, read/write operations stops.

When reading, T(cycle) = XIPR_TCNT; when writing, T(cycle) = XIPW_TCNT.

For more details, refer to the figure below.



30.4 QSPI registers

These registers must be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

Table 30-1 QSPI register map and reset values

Register	Offset	Reset value
CMD_W0	0x0	0x0000 0000
CMD_W1	0x4	0x0100 0003
CMD_W2	0x8	0x0000 0000
CMD_W3	0xC	0x0000 0000
CTRL	0x10	0x0010 0083
FIFOSTS	0x18	0x0000 0001
CTRL2	0x20	0x0000 0000
CMDSTS	0x24	0x0000 0000
RSTS	0x28	0x0000 0000
FSIZE	0x2C	0xF0000 0000
XIP CMD_W0	0x30	0x0000 3000
XIP CMD_W1	0x34	0x0000 2000
XIP CMD_W2	0x38	0x0F01 0F01
XIP CMD_W3	0x3C	0x0000 0000
CTRL3	0x40	0x0000 0000
REV	0x50	0x0001 0500
DT	0x100	0x0000 0000

30.4.1 Command word 0 (CMD_W0)

No-wait states, assessable by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 0	SPIADR	0x0	rw	SPI Flash address This register defines the values of SPI Flash addresses, and sends them to SPI Flash. The address byte is based on the bit [2: 0] of CMD_W1 register.

30.4.2 Command word 1 (CMD_W1)

No-wait states, assessable by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 29	Reserved	0x0	resd	Kept at its default value.
Bit 28	PEMEN	0x0	rw	Performance enhanced mode enable Locates between the address and the second dummy state. In this mode, the command status after the second read command can be removed. Do not set this bit when CMD_W2=0. 0: Performance enhanced mode disabled 1: 1-byte Performance enhanced mode enabled
Bit 27: 26	Reserved	0x0	resd	Kept at its default value.

Bit 25: 24	INSLEN	0x1	rw	Instruction code length Instruction code is required for SPI Flash command execution. The instruction code length varies from SPI Flash supplier to SPI Flash supplier. Thus this register can be used to program the desired instruction code length. Typically, the instruction code is one-byte length. However, if the user sets two-byte instruction code, the host controller sends this instruction code twice. 00: No instruction code. It cannot be used until the continuous read mode command is completed. 01: 1-byte instruction code 10: 2-byte instruction code (repeated instruction code) 11: Reserved
Bit 23: 16	DUM2	0x0	rw	Second dummy state cycle The second dummy cycle is located between the address and data state, excluding performance enhanced mode status. The user can check there is a dummy state between the address and data status in SPI Flash specification. The host controller sends logic 1 in a dummy cycle. 0: No second dummy state 1~32: 1 dummy second period~32 dummy second period
Bit 15: 3	Reserved	0x0	resd	Kept at its default value.
Bit 2: 0	ADRLEN	0x3	rw	SPI address length This field defines the number of bytes of the SPI Flash address, ranging from one to four bytes. 000: No address state 001: 1-byte address 010: 2-byte address 011: 3-byte address 100: 4-byte address Others: Reserved

30.4.3 Command word 2 (CMD_W2)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 0	DCNT	0x0	rw	Read/Write data counter This bit must be set to 0 when executing read status command. 0: No read/write data 1~FFFFFFFF: 1~FFFFFFFF byte data Note: This register must not be padded with 0 for data read or write. However, for “read status” or “write enable” instruction, this register must be set to 0.

30.4.4 Command word 3 (CMD_W3)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 24	INSC	0x00	rw	Instruction code This code is set to enable SPI Flash command.
Bit 23: 16	PEMOPC	0x00	rw	Performance enhanced mode operation code This field works with the PEMEN bit. This code can be padded to execute performance enhanced mode. Follow the corresponding Flash specification document to write the corresponding value.

Bit 15: 10	Reserved	0x0	resd	Kept at its default value.
Bit 7: 5	OPMODE	0x0	rw	SPI Operation mode 000: Serial mode (1-1-1) 001: Dual mode (1-1-2) 010: Quad mode (1-1-4) 011: Dual I/O mode (1-2-2) 100: Quad I/O mode (1-4-4) 101: DPI mode (2-2-2) 110: QPI mode (4-4-4) Others: Reserved
Bit 4	Reserved	0x0	resd	Kept at its default value.
Bit 3	RSTSC	0x0	rw	Read SPI status configuration This bit is valid only when read state and write is enabled. The user must send a SPI read state command. 0: Hardware read. The controller keeps polling until the state is ready (not busy) and feedbacks to the status register. 1: Software read. Read status ones and feedback to the status register until the user is able to read it.
Bit 2	RSTSEN	0x0	rw	Read SPI status enable This bit is valid when WEN =“0”, and the user must send SPI read status command. 0: Read SPI status disabled 1: Read SPI status enabled
Bit 1	WEN	0x0	rw	Write data enable This bit is used to enable SPI write data, excluding read data or read status (read data return path); the user must set write enable bit=1 for other SPI commands. Note: Write enable must be set to 1 in data write or Flash erase command. The write enable must be set to 0 only in read data or read status command. 0: Disabled 1: Enabled
Bit 0	Reserved	0x0	resd	Kept at its default value.

30.4.5 Control register (CTRL)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 22	Reserved	0x000	resd	Kept at its default value.
Bit 21	KEYEN	0x0	rw	SPI data encryption key enable 0: SPI data encryption key disabled 1: SPI data encryption key enabled When this bit is enabled, raw data is converted into ciphertext and written into the QSPI peripheral through QSPIKEY. While read, data is decrypted into plaintext and sends to the CPU.
Bit 20	XIPSEL	0x1	rw	XIP port selection Read SPI Flash data from the following ports: 0: Command slave port 1: XIP port When this bit is switched, the QSPI sends automatically an Abort signal. The user can send a command only after the completion of Abort function.
Bit 19	XIPRCMDF	0x0	rw	XIP read command flush

Bit 18: 16	BUSY	0x0	rw	Busy bit of SPI status The host polls this busy bit and remains in hardware read state. 000~111: bit 0~bit7
Bit 15: 9	Reserved	0x00	resd	Kept at its default value.
Bit 8	ABORT	0x0	rw	Refresh all commands/FIFOs and reset state machine When an Abort event occurs, this bit must be written (This bit is automatically cleared to 0). 0: No effect 1: Enabled
Bit 7	XIPIDLE	0x1	ro	XIP port idle status 0: XIP port is busy 1: XIP port is idle
Bit 6: 5	Reserved	0x0	resd	Kept at its default value.
Bit 4	SCKMODE	0x0	rw	Sckout mode 0: For mode 0, sck_out is low in idle state, with data capture on the first edge 1: For mode 3, sck_out is high in idle state, with data capture on the second edge
Bit 3: 0	CLKDIV	0x3	rrw	Clk divider This field is used to divide the spi_clk. 0000: Divided by 2 0001: Divided by 4 0010: Divided by 6 0011: Divided by 8 0100: Divided by 3 0101: Divided by 5 0110: Divided by 10 0111: Divided by 12 1xxx: Divided by 1

30.4.6 FIFO status register (FIFOSTS)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 2	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 1	RXFIFORDY	0x0	ro	RxFIFO ready status When this bit is set, it indicates the following: 1: RxFIFO full 2: The remaining data in the RxFIFO is less than the depth of RxFIFO, but it is the last data.
Bit 0	TXFIFORDY	0x1	ro	TxFIFO ready status When the TxFIFO is set, it indicates that the TxFIFO will get empty so that data can be transmitted into it until it becomes full.

30.4.7 Control register 2 (CTRL2)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 14	Reserved	0x0000 0	resd	Kept at its default value.

Bit 13: 12	RXFIFO THOD	0x0	rw	This field is used to program the level value to trigger RxFIFO threshold interrupt for DMA handshake mode. The value is in terms of word. The trigger value is the data in the RxFIFO. 00: 8 WORD 01: 16 WORD 10: 24 WORD 11: Reserved
Bit 11: 10	Reserved	0x0	resd	Kept at its default value.
Bit 9: 8	TXFIFO THOD	0x0	rw	This field is used to program the level value to trigger TxFIFO threshold interrupt for DMA handshake mode. The value is in terms of word. The trigger value is the data in the TxFIFO. 00: 8 WORD 01: 16 WORD 10: 24 WORD 11: Reserved
Bit 7: 2	Reserved	0x00	resd	Kept at its default value.
Bit 1	CMDIE	0x0	rw	Command complete Interrupt enable 0: Command complete Interrupt disabled 1: Command complete Interrupt enabled
Bit 0	DMAEN	0x0	rw	DMA enable Note: This bit must be disabled before moving from command-based slave port to XIP port.

30.4.8 Command status register (CMDSTS)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	CMDSTS	0x0	rw1c	Command complete status Set at the end of a command.

30.4.9 Read status register (RSTS)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 00	resd	Kept at its default value.
Bit 7: 0	SPISTS	0x00	ro	SPI Read status The host sends a read SPI Flash status command and stores the returned data in this register. By reading it, the user can check the status of SPI Flash.

30.4.10 Flash size register (FSIZE)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 0	SPIFSIZE	0xF000 0000	rw	SPI Flash Size In direct address map mode, system address is always greater than that of SPI Flash. The user must mask the upper bits of the system address to match SPI Flash size.

30.4.11 XIP command word 0 (XIP CMD_W0)

No-wait states, accessible by words.

Bit	Name	Reset value	Type	Description
Bit 31: 20	Reserved	0x000	resd	Kept at its default value.
Bit 19: 12	XIPR_INSC	0x03	rw	XIP read instruction code This field is set to execute SPI Flash command of XPI read.
Bit 11	XIPR_ADRLEN	0x0	rw	XIP read address length This bit defines the number of bytes of SPI Flash address. The user can uses this bit to program a 3-byte or 4-byte XIP read address. 0: 3-byte address 1: 4-byte address
Bit 10: 8	XIPR_OPMODE	0x0	rw	XIP read Operation mode 000: Serial mode (1-1-1) 001: Dual mode (1-1-2) 010: Quad mode (1-1-4) 011: Dual IO mode (1-2-2) 100: Quad IO mode (1-4-4) 101: DPI mode (2-2-2) 110: QPI mode (4-4-4) 111: Reserved
Bit 7: 0	XIPR_DUM2	0x00	rw	XIP Read second dummy cycle The second dummy state is located between the address and data status, excluding continuous read mode status. The user can check if there is a dummy state between the address and data status in SPI Flash specification. The host controller issues logic 1 in a dummy cycle. 0: No second dummy state 1~32: 1 dummy second period~32 dummy second periods

30.4.12 XIP command word 1 (XIP CMD_W1)

No-wait states, accessible by words.

Bit	Name	Reset value	Type	Description
Bit 31: 18	Reserved	0x000	resd	Kept at its default value.
Bit 19: 12	XIPW_INSC	0x02	rw	XIP write instruction code The user can set this field to enable a SPI Flash command of XIP write.
Bit 11	XIPW_ADRLEN	0x0	rw	XIP write address length This bit defines the number of bytes of SPI Flash address. The user can uses this bit to program a 3-byte or 4-byte XIP write address. 0: 3-byte address 1: 4-byte address
Bit 10: 8	XIPW_OPMODE	0x0	rw	XIP write operation mode 000: Serial mode (1-1-1) 001: Dual mode (1-1-2) 010: Quad mode (1-1-4) 011: Dual IO mode (1-2-2) 100: Quad IO mode (1-4-4) 101: DPI mode (2-2-2) 110: QPI mode (4-4-4) 111: Reserved

Bit 7: 0	XIPW_DUM2	0x00	rw	XIP Write second dummy cycle The second dummy state is located between the address and data status, excluding continuous read mode status. The user can check if there is a dummy state between the address and data status in SPI Flash specification. The host controller issues logic 1 in a dummy cycle. 0: No second dummy state 1~32: 1 dummy second period~32 dummy second periods

30.4.13 XIP command word 2 (XIP CMD_W2)

No-wait states, accessible by bytes, half words and words.

Bit	Name	Reset value	Type	Description
Bit 31	XIPW_SEL	0x0	rw	XIP write mode select 0: Mode D 1: Mode T The XIP slave port can be used for the improvement of read/write process and performance. Mode D: When data are written to the consecutive addresses, put a limit on the maximum data count (DCNT) in a single write operation Mode T: When two consecutive data are written and the addresses are also continuous, and the interval is less than a specified time (TCNT), then they are combined into one command.
Bit 30: 24	XIPW_TCNT	0x0F	rw	This indicates the time counter that is used to judge time interval in mode T. Value is in terms of sck_out period. This counter is valid when mode T is selected.
Bit 23: 22	Reserved	0x0	resd	Kept at its default value.
Bit 21: 16	XIPW_DCNT	0x01	rw	This indicates the time counter that is used to judge the maximum data count in mode D. Value is in terms of word, and must not be 0. This counter is valid when mode D is selected.
Bit 15	XIPR_SEL	0x0	rw	XIP read mode select 0: Mode D 1: Mode T The XIP slave port can be used for the improvement of read/write process and performance. Mode D: When data are read from the consecutive addresses, put a limit on the maximum data count (DCNT) in a single write operation Mode T: When two consecutive data are read and the addresses are also continuous, and the interval is less than a specified time (TCNT), then they are combined into one command.
Bit 14: 8	XIPR_TCNT	0x0F	rw	This indicates the time counter that is used to judge time interval in mode T. Value is in terms of sck_out period. This counter is valid when mode T is selected.
Bit 7: 6	Reserved	0x0	resd	Kept at its default value.
Bit 5: 0	XIPR_DCNT	0x01	rw	This indicates the time counter that is used to judge the maximum data count in mode D. Value is in terms of word, and must not be 0. This counter is valid when mode D is selected.

30.4.14 XIP command word 3 (XIP CMD_W3)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 4	Reserved	0x0000 000	resd	Kept at its default value.
Bit 3	CSTS	0x0	r	Cache Status 0: Cache verified 1: Cache failed
Bit 2: 1	Reserved	0x0	resd	Kept at its default value.
Bit 0	BYPASSC	0x0	rw	Bypass Cache Function When this bit is set, the high-speed cache feature is deactivated, and all read transfers do not check high-speed cache. Cache function is only applicable to XIP Read applications only (extended Flash). For XIP read/write applications (extended PSRAM), this bit must be set to 1.

30.4.15 Control register (CTRL3)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 9	Reserved	0x0000 000	resd	Kept at its default value.
Bit 8	ISPC	0x0	rw	Input sampling phase correction enable This bit is used to correct sampling phase according to input sampling phase delay and SPI Flash output timing. 0: Fixed sampling phase 1: Input sampling phase correction is enabled
Bit 7: 6	Reserved	0x0	resd	Kept at its default value.
Bit 0	ISPD	0x00	rw	Input sampling phase delay When ISPC is set, this bit setting is taken into account.

30.4.16 Revision register (REV)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 0	REV	0x0001 0500	ro	Indicates IP version.

30.4.17 Data port register (DT)

No-wait states, accessible by bytes, half-words and words.

Bit	Name	Reset value	Type	Description
Bit 31: 0	DT	0x0000 0000	rw	Data port register This port is used for data read or write.

31 Debug (DEBUG)

31.1 Debug introduction

Cortex®-M4F core provides powerful debugging features including halt and single step support, as well as trace function that is used for checking the details of the program execution. The debug features are implemented withserial wire debug (SWD).

ARM Cortex®-M4F reference documentation:

- Cortex®-M4 Technical Reference Manual (TRM)
- ARM Debug Interface V5
- ARM CoreSight Design Kit revision r1p0 Technical Reference Manual

31.2 Debug and Trace

It is possible to support debugging for different peripherals, and configure the status of peripherals during debugging. For timers and watchdogs, the user can select whether or not to stop or continue counting during debugging; For CAN, the user can select whether or not to stop or continue updating receive registers during debugging; For I^2C, the user can select whether or not to stop or continue SMBUS timeout counting.

In addition, code debugging is supported in Low-power mode. In Sleep mode, the clock programmed by code remains active for HCLK and FCLK to continue to work. In DeepSleep mode, HICK oscillator is enabled to feed FCLK and HCLK.

There are several ID codes inside the MCU, which is accessible by the debugger using the DEBUG_IDCODE at address 0xE0042000. It is part of the DEBUG and is mapped on the external PPB bus. These codes are accessible using the JTAG debug port or the SWD debug port or by the user software. They are even accessible while the MCU is under system reset.

31.3 I/O pin control

SWJ-DP debug is supported in different packages of AT32F455x series. It uses 5 general-purpose I/O ports. After reset, the SWJ-DP can be immediately used by the debugger as a default function.

GPIO and IOMUX registers can be configured to allow users to switch between debug ports or disable debug feature.

31.4 DEGUB registers

Table 31-1 shows DEBUG register map and reset values.

These peripheral registers must be accessed by words (32 bits).

Table 31-1 DEBUG register address and reset value

Register	Offset	Reset value
DEBUG_IDCODE	0xE004 2000	0xXXXX XXXX
DEBUG_CTRL	0xE004 2004	0x0000 0000
DEBUG_APB1_PAUSE	0xE004 2008	0x0000 0000
DEBUG_APB2_PAUSE	0xE004 200C	0x0000 0000
DEBUG_APB3_PAUSE	0xE004 2010	0x0000 0000
DEBUG_SER_ID	0xE004 2020	0x0000 XX0X

31.4.1 DEBUG device ID (DEBUG_IDCODE)

MCU integrates an ID code that is used to identify MCU's revision code. The DEBUG_IDCODE register is mapped on the external PPB bus at address 0xE0042000. This code is accessible by the JTAG debug port or SW debug port or by the user code.

Bit	Name	Reset value	Type	Description
Bit 31: 0	PID	0xXXXX XXXX	ro	PID information

PID [31: 0]	AT32 part number	FLASH size	Packages
0x7006_32C0	AT32F455ZET7	512KB	144LQFP
0x7005_3241	AT32F455ZCT7	256KB	144LQFP
0x7006_32C2	AT32F455VET7	512KB	100LQFP
0x7005_3243	AT32F455VCT7	256KB	100LQFP
0x7006_32C4	AT32F455RET7	512KB	64LQFP
0x7005_3245	AT32F455RCT7	256KB	64LQFP
0x7006_32C6	AT32F455CET7	512KB	48LQFP
0x7005_3247	AT32F455CCT7	256KB	48LQFP
0x7006_32C8	AT32F455CEU7	512KB	48QFN
0x7005_3249	AT32F455CCU7	256KB	48QFN
0x7006_32CA	AT32F456ZET7	512KB	144LQFP
0x7005_324B	AT32F456ZCT7	256KB	144LQFP
0x7006_32CC	AT32F456VET7	512KB	100LQFP
0x7005_324D	AT32F456VCT7	256KB	100LQFP
0x7006_32CE	AT32F456RET7	512KB	64LQFP
0x7005_324F	AT32F456RCT7	256KB	64LQFP
0x7006_32D0	AT32F456CET7	512KB	48LQFP
0x7005_3251	AT32F456CCT7	256KB	48LQFP
0x7006_32D2	AT32F456CEU7	512KB	48QFN
0x7005_3253	AT32F456CCU7	256KB	48QFN
0x7006_32D4	AT32F457ZET7	512KB	144LQFP
0x7005_3255	AT32F457ZCT7	256KB	144LQFP
0x7006_32D6	AT32F457VET7	512KB	100LQFP
0x7005_3257	AT32F457VCT7	256KB	100LQFP
0x7006_32D8	AT32F457RET7	512KB	64LQFP
0x7005_3259	AT32F457RCT7	256KB	64LQFP

31.4.2 DEBUG control register (DEBUG_CTRL)

This register is asynchronously reset by POR Reset (not reset by system reset). It can be written by the debugger under reset.

Bit	Name	Reset value	Type	Description
Bit 31:3	Reserved	0x0000 0000	resd	Always 0.
Bit 2	STANDBY_DEBUG	0x0	rw	Debug Standby mode control bit 0: The whole 1.2V digital circuit is unpowered in Standby mode 1: The whole 1.2V digital circuit is not unpowered in Standby mode, and the system clock is provided by the internal RC oscillator (HICK)
Bit 1	DEEPSLEEP_DEBUG	0x0	rw	Debug Deepsleep mode control bit 0: In Deepsleep mode, all clocks in the 1.2V domain are disabled. When exiting from Deepsleep mode, the internal RC oscillator (HICK) is enabled, and HICK is used as the system clock source, and the software must reprogram the system clock according to application requirements. 1: In Deepsleep mode, system clock is provided by the internal RC oscillator (HICK). When exiting from Deepsleep mode, HICK is used as the system clock source, and the software must reprogram the system clock. According to application requirements.
Bit 0	SLEEP_DEBUG	0x0	rw	Debug Sleep mode control bit 0: When entering Sleep mode, CPU HCLK clock is disabled, but other clocks remain active. When exiting from Sleep mode, it is not necessary to reprogram the clock system. 1: When entering Sleep mode, all clocks keep running.

31.4.3 DEBUG APB1 pause register (DEBUG_ APB1_PAUSE)

This register is asynchronously reset by POR Reset (not reset by system reset). It can be written by the debugger under reset.

Bit	Name	Reset value	Type	Description
Bit 31: 29	Reserved	0x0	resd	Kept at its default value.
Bit 28	I2C3_SMBUS_TIMEOUT	0x0	rw	I^2C3 pause control bit 0: I^2C3 SMBUS timeout control works normally 1: I^2C3 SMBUS timeout control stops running
Bit 27	I2C2_SMBUS_TIMEOUT	0x0	rw	I^2C2 pause control bit 0: I^2C2 SMBUS timeout control works normally 1: I^2C2 SMBUS timeout control stops running
Bit 26:25	Reserved	0x0	resd	Kept at its default value.
Bit 24	I2C1_SMBUS_TIMEOUT	0x0	rw	I^2C1 pause control bit 0: I^2C1 SMBUS timeout control works normally 1: I^2C1 SMBUS timeout control stops running
Bit 23: 13	Reserved	0x0	resd	Kept at its default value.

Bit 12	WDT_PAUSE	0x0	rw	WDT pause control bit 0: WDT works normally 1: WDT stops running
Bit 11	WWDT_PAUSE	0x0	rw	WWDT pause control bit 0: WWDT works normally 1: WWDT stops running
Bit 10:9	Reserved	0x0	resd	Kept at its default value.
Bit 8	TMR14_PAUSE	0x0	rw	TMR14 pause control bit 0: TMR14 works normally 1: TMR14 stops running
Bit 7	TMR13_PAUSE	0x0	rw	TMR13 pause control bit 0: TMR13 works normally 1: TMR13 stops running
Bit 6	TMR12_PAUSE	0x0	rw	TMR12 pause control bit 0: TMR12 works normally 1: TMR12 stops running
Bit 5	TMR7_PAUSE	0x0	rw	TMR7 pause control bit 0: TMR7 works normally 1: TMR7 stops running
Bit 4	TMR6_PAUSE	0x0	rw	TMR6 pause control bit 0: TMR6 works normally 1: TMR6 stops running
Bit 3	TMR5_PAUSE	0x0	rw	TMR5 pause control bit 0: TMR5 works normally 1: TMR5 stops running
Bit 2	TMR4_PAUSE	0x0	rw	TMR4 pause control bit 0: TMR4 works normally 1: TMR4 stops running
Bit 1	TMR3_PAUSE	0x0	rw	TMR3 pause control bit 0: TMR3 works normally 1: TMR3 stops running
Bit 0	TMR2_PAUSE	0x0	rw	TMR2 pause control bit 0: TMR2 works normally 1: TMR2 stops running

31.4.4 DEBUG APB2 pause register (DEBUG_ APB2_PAUSE)

This register is asynchronously reset by POR Reset (not reset by system reset). It can be written by the debugger under reset.

Bit	Name	Reset value	Type	Description
Bit 31: 19	Reserved	0x0000	resd	Kept at its default value.
Bit 18	TMR11_PAUSE	0x0	rw	TMR11 pause control bit 0: TMR11 works normally 1: TMR11 stops running
Bit 17	TMR10_PAUSE	0x0	rw	TMR10 pause control bit 0: TMR10 works normally 1: TMR10 stops running
Bit 16	TMR9_PAUSE	0x0	rw	TMR9 pause control bit 0: TMR9 works normally 1: TMR9 stops running
Bit 15: 2	Reserved	0x0000	resd	Kept at its default value.
Bit 1	TMR8_PAUSE	0x0	rw	TMR8 pause control bit 0: TMR8 works normally 1: TMR8 stops running
Bit 0	TMR1_PAUSE	0x0	rw	TMR1 pause control bit 0: TMR2 works normally 1: TMR2 stops running

31.4.5 DEBUG APB3 pause register (DEBUG_ APB3_PAUSE)

This register is asynchronously reset by POR Reset (not reset by system reset). It can be written by the debugger under reset.

Bit	Name	Reset value	Type	Description
Bit 31: 1	Reserved	0x0000 0000	resd	Kept at its default value.
Bit 0	ERTC_PAUSE	0x0	rw	ERTC pause control bit 0: ERTC works normally 1: ERTC stops running

31.4.6 DEBUG SERIES ID register (DEBUG_ SER_ID)

DEBUG_SIE_ID register is used to identify MCU part number and its revision code. The DEBUG_IDCODE register is mapped on the external PPB bus. This register is asynchronously reset by POR Reset (not reset by system reset). This code is accessible by the JTAG debug port or SW debug port or by the user code.

Bit	Name	Reset value	Type	Description
Bit 31: 16	Reserved	0x0000	resd	Kept at its default value.
Bit 15: 8	SER_ID	0xXX	ro	MCU part number ID AT32F455: 0x15 AT32F456: 0x16 AT32F457: 0x17
Bit 7: 3	Reserved	0x0X	resd	Kept at its default value.
Bit 2: 0	REV_ID	0xX	ro	Revision code 0x0: Revision A

32 Revision history

Document Revision History

Date	Version	Revision Note
2024.12.16	2.00	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services, and ARTERY assumes no liability whatsoever relating to the choice, selection or use of the ARTERY products and services described herein

No license, express or implied, to any intellectual property rights is granted under this document. If any part of this document deals with any third party products or services, it shall not be deemed a license granted by ARTERY for the use of such third party products or services, or any intellectual property contained therein, or considered as a warranty regarding the use in any manner of such third party products or services or any intellectual property contained therein.

Unless otherwise specified in ARTERY's terms and conditions of sale, ARTERY provides no warranties, express or implied, regarding the use and/or sale of ARTERY products, including but not limited to any implied warranties of merchantability, fitness for a particular purpose (and their equivalents under the laws of any jurisdiction), or infringement on any patent, copyright or other intellectual property right.

Purchasers hereby agree that ARTERY's products are not designed or authorized for use in: (A) any application with special requirements of safety such as life support and active implantable device, or system with functional safety requirements; (B) any aircraft application; (C) any aerospace application or environment; (D) any weapon application, and/or (E) or other uses where the failure of the device or product could result in personal injury, death, property damage. Purchasers' unauthorized use of them in the aforementioned applications, even if with a written notice, is solely at purchasers' risk, and Purchasers are solely responsible for meeting all legal and regulatory requirements in such use.

Resale of ARTERY products with provisions different from the statements and/or technical characteristics stated in this document shall immediately void any warranty grant by ARTERY for ARTERY's products or services described herein and shall not create or expand any liability of ARTERY in any manner whatsoever.