

ARM®-based 32-bit Cortex®-M4F MCU + FPU, with 256 Kbyte ~ 1024 Kbyte Internal Flash Memory, USB, CAN, 18 Timers, 3 ADCs, 16 Communication Interfaces

Feature

- **Core: ARM®32-bit Cortex®-M4F CPU with FPU**
 - 200 MHz maximum frequency, with Memory Protection Unit (MPU),
 - Single-cycle multiplication and hardware division
 - Floating Point Unit (FPU)
 - DSP instructions
- **Memory**
 - 256 to 1024 Kbyte Flash instruction/data memory
 - SPI-M interface: Extra interfacing up to 16Mbytes of external SPI Flash (as instruction/data memory)
 - Up to 96 + 128 Kbyte SRAM
 - With 4 chip-select external controllers (XMC). Supports CF Card, SRAM, PSRAM, NOR, and NAND memory
 - Parallel LCD interface, 8080/6800 modes
- **Clock, Reset, and Power Management**
 - 2.6 V ~ 3.6 V and I/O pins
 - POR, PDR, and programmable Voltage Detector (PVD)
 - 4 to 25 MHz crystal oscillator
 - Internal 8 MHz factory-trimmed RC (25 °C 1% precision, overall temperature 2.5% precision)
 - Internal 40 kHz RC with calibration
 - 32 kHz oscillator RTC with calibration
- **Low Power Consumption**
 - Sleep, Stop, and Standby mode
- **3 12-bit A/D converters, 0.5 µs converters (up to 21 channels)**
 - Converting range: 0 V ~ 3.6 V
 - 3 sample and hold capability
 - Temperature sensor
- **2 12-bit D/A converters**
- **DMA: 12-channel DMA controller**
 - Peripherals supported: Timer, ADC, DAC, SDIO, I²S, SPI, I²C, and USART
- **Debug Mode**
 - Serial Wire Debug (SWD) and JTAG interface
 - Cortex®-M4F Embedded Trace Macrocell (ETM™)
- **Up to 112 Fast I/O Interfaces**
 - 36/51/80/112 multifunctional and bidirectional I/O ports, all mappable to 16 external interrupt vectors and almost all 5 V-tolerant.
- **Up to 18 Timers**
 - Up to 8 16-bit timers + 2 32-bit timers; each with 4 IC/OC/PWM or pulse counter and incremental encoder input.
 - Up to 3 x 16-bit motor control PWM advanced timers with dead-time generator and emergency stop
 - 2 Watchdog timers (independent and window)
 - System timer: 24-bit decremental counter
 - 2 16-bit basic timers driving DAC
- **Up to 16 Communication Interfaces**
 - Up to 3 x I²C interfaces (SMBus/PMBus)
 - Up to 5 x USARTs (ISO7816 interface, LIN, IrDA capability, and modem control)
 - Up to 4 x SPIs (50 Mbit/s), all with I²S interface multiplexed.
 - CAN interface (2.0B active)
 - USB2.0 full-speed interface
 - Up to 2 SDIO interfaces
- **CRC Calculation Unit, 96-bit Chip-only Code**
- **Packaging**
 - LQFP144 20x20 mm
 - LQFP100 14x14 mm
 - LQFP64 10x10 mm
 - LQFP48 7x7 mm
 - QFN48 6 x6 mm
- **List of Models**

Internal Flash	Model
256 Kbytes	AT32F403CCT6, AT32F403CCU6, AT32F403RCT6, AT32F403VCT6, AT32F403ZCT6
512 Kbytes	AT32F403CET6, AT32F403CEU6, AT32F403RET6, AT32F403VET6, AT32F403ZET6
1024 Kbytes	AT32F403CGT6, AT32F403CGU6, AT32F403RGT6, AT32F403VGT6, AT32F403ZGT6

Contents

1 System Architecture	23
1.1 System Introduction	23
1.1.1 Bus Architecture	25
1.1.2 ARM Cortex®-M4F Processor	25
1.2 Address Map	26
1.2.1 Register Map	27
1.2.2 Bit Banding.....	29
1.2.3 On-chip SRAM.....	29
1.2.4 On-chip Flash	29
1.3 Boot Configuration	33
1.4 Device Characteristics Information	35
1.4.1 Description of Register Abbreviations	35
1.4.2 Flash Memory Size Register	35
1.4.3 Device Electronic Signature	35
2 Power Control (PWR)	37
2.1 Introduction.....	37
2.2 Main Features	37
2.3 Function Overview.....	37
2.3.1 Power Supply.....	37
2.3.1.1 VDD/VDDA Power Domain	38
2.3.1.2 Core Power Domain	40
2.3.2 Low-power Mode	40
2.3.2.1 Sleep Mode	41
2.3.2.2 Stop Mode	42
2.3.2.3 Standby Mode	43
2.3.2.4 Debug Mode.....	44
2.3.3 Auto-wakeup (AWU).....	44
2.4 PWR Registers	44
2.4.1 Power Control Register (PWR_CTRL).....	45
2.4.2 Power Control/Status Register (PWR_CTRLST).....	45
3 Reset and Clock Control (RCC).....	47
3.1 Reset	47
3.1.1 System Reset.....	47
3.1.2 Power Reset	47
3.1.3 Backup Domain Reset	48
3.2 Clocks	48

3.2.1	HSE Clock	50
3.2.2	HIS Clock.....	50
3.2.3	PLL.....	51
3.2.4	LSE Clock.....	51
3.2.5	LSI Clock.....	51
3.2.6	System Clock (SYSCLK) Selection	52
3.2.7	Clock Failure Detection (CFD)	52
3.2.8	RTC Clock	52
3.2.9	Watchdog Clock	53
3.2.10	Clock-out Capability	53
3.3	RCC Registers Description	53
3.3.1	Clock Control Register (RCC_CTRL)	54
3.3.2	Clock Configuration Register (RCC_CFG)	56
3.3.3	Clock Interrupt Register (RCC_CLKINT)	58
3.3.4	APB2 Peripheral Reset Register (RCC_APB2RST)	59
3.3.5	APB1 Peripheral Reset Register (RCC_APB1RST)	61
3.3.6	AHB Peripheral Clock Enable Register (RCC_AHBEN)	63
3.3.7	APB2 Peripheral Clock Enable Register (RCC_APB2EN)	64
3.3.8	APB1 Peripheral Clock Enable Register (RCC_APB1EN)	66
3.3.9	Backup Domain Control Register (RCC_BDC).....	68
3.3.10	Control/Status register (RCC_CTRLSTS).....	69
3.3.11	Additional Register (RCC_MISC)	70
4	Backup Registers (BKPR).....	72
4.1	BKPR Introduction	72
4.2	BKPR Main Features	72
4.3	BKPR Function Overview	72
4.3.1	Tamper Detection	72
4.3.2	RTC Calibration.....	72
4.4	BKPR Registers	73
4.4.1	Backup Data Register x (BKPR_DRx) (x = 1, ..., 42)	75
4.4.2	RTC Clock Calibration Register (BKPR_RTCCAL).....	75
4.4.3	Backup Control Register (BKPR_CTRL).....	75
4.4.4	Backup Control/Status Register (BKPR_CTRLSTS)	76
5	Flash Memory Controller (FMC)	77
5.1	FMC Introduction	77
5.2	Main Features	77

5.2.1	Flash Memory Architecture.....	77
5.2.2	External Flash Memory Organization	80
5.3	Function Overview.....	80
5.3.1	Read Operation	80
5.3.1.1	Instruction Fetch	81
5.3.1.2	D-Code Interface	81
5.3.1.3	Flash Access Controller	81
5.3.2	Flash Program/Erase Controller (FPEC)	81
5.3.2.1	Key Value	81
5.3.2.2	Unlock the Flash Memory	82
5.3.2.3	Main Flash Programming	82
5.3.2.4	Flash Erase	83
5.3.2.5	Option Byte Programming	84
5.3.3	Protection	85
5.3.3.1	Write Protection.....	85
5.3.3.2	Read Protection.....	85
5.3.3.3	Option Byte Block Write Protection.....	86
5.3.4	Option Byte Description	86
5.4	FMC Registers	88
5.4.1	Flash Access Control Register (FLASH_ACR)	89
5.4.2	FPEC Key Register (FLASH_FCKEY)	90
5.4.3	Flash OPTKEY Register (FLASH_OPTKEYR)	90
5.4.4	Flash Status Register (FLASH_STS)	90
5.4.5	Flash Control Register (FLASH_CTRL)	91
5.4.6	Flash Address Register (FLASH_ADDR)	92
5.4.7	Option Byte Register (FLASH_UOB).....	92
5.4.8	Write Protection Register (FLASH_WRPRT).....	93
5.4.9	FPEC Key Register 2 (FLASH_FCKEY2)	93
5.4.10	Flash Status Register 2 (FLASH_STS2)	94
5.4.11	Flash Control Register 2 (FLASH_CTRL2).....	94
5.4.12	Flash Address Register 2 (FLASH_ADDR2).....	95
5.4.13	FPEC Key Register 3 (FLASH_FCKEY3)	95
5.4.14	Flash Option Byte Register (FLASH_SELECT)	96
5.4.15	Flash Status Register 3 (FLASH_STS3)	96
5.4.16	Flash Control Register 3 (FLASH_CTRL3).....	97
5.4.17	Flash Address Register 3 (FLASH_ADDR3).....	98
5.4.18	Flash Decode Address Register (FLASH_DA)	98

6 CRC Calculation Unit (CRC)	99
6.1 CRC Introduction	99
6.2 CRC Main Features.....	99
6.3 CRC Function Overview	99
6.4 CRC Registers	100
6.4.1 Data Register (CRC_DR)	100
6.4.2 Independent Data Register (CRC_IDR)	100
6.4.3 Control Register (CRC_CTRL)	101
7 General-purpose and Alternate-function I/Os (GPIOs and AFIOs)	102
7.1 Introduction.....	102
7.2 Main Features	102
7.3 Function Overview.....	102
7.3.1 GPIO Pin Configuration.....	102
7.3.2 External Interrupt/Wakeup Lines	104
7.3.3 Input Configuration.....	104
7.3.4 Analog Input Configuration.....	105
7.3.5 Output Configuration	106
7.3.6 GPIO Locking Mechanism.....	107
7.3.7 Alternate Function (AF)	107
7.4 IO Mapping Function Configuration	111
7.4.1 OSC32_IN/OSC32_OUT as GPIO Interface PC14/PC15	111
7.4.2 OSC_IN/OSC_OUT Pin as GPIO Interface PD0/PD1.....	111
7.4.3 CAN Alternate Function Remapping.....	111
7.4.4 JTAG/SWD Alternate Function Remapping.....	112
7.4.5 ADC Alternate Function Remapping	112
7.4.6 Timer Alternate Function Remapping	113
7.4.7 USART Alternate Function Remapping	115
7.4.8 I ² C Alternate Function Remapping.....	116
7.4.9 SPI1/I2S1 Alternate Function Remapping	116
7.4.10 SPI4/I2S4 Alternate Function Remapping	116
7.4.11 SDIO2 Alternate Function Remapping	116
7.4.12 External SPIF Alternate Function Remapping	117
7.5 GPIO and AFIO Registers	117
7.5.1 Port Configuration Register Low (GPIOx_CTRLLO) (x = A...E).....	118
7.5.2 Port Configuration Register High (GPIOx_CTRLH) (A...E).....	119
7.5.3 Port Input Data Register (GPIOx_IPTDT) (x = A...E)	120

7.5.4	Port Output Data Register (GPIO _x _OPTDT) (x = A...E)	120
7.5.5	Port Bit Set/Reset Register (GPIO _x _BSRE) (x = A...E)	120
7.5.6	Port Bit Reset Register (IO _x _BRE) (x = A...E)	121
7.5.7	Port Configuration Lock Register (GPIO _x _LOCK) (x = A...E)	121
7.5.8	Alternate Event Control Register (AFIO_EVCTRL)	122
7.5.9	AF Remap and Debug I/O Configuration Register (AFIO_MAP)	122
7.5.10	Alternate External Interrupt Configuration Register 1 (AFIO_EXTIC1)	125
7.5.11	Alternate External Interrupt Configuration Register 2 (AFIO_EXTIC2)	125
7.5.12	Alternate External Interrupt Configuration Register 3 (AFIO_EXTIC3)	127
7.5.13	Alternate External Interrupt Configuration Register 4 (AFIO_EXTIC4)	127
7.5.14	AF Remap and Debug I/O Configuration Register 2 (AFIO_MAP2)	127
8	Interrupts and Events	129
8.1	Nested Vectored Interrupt Controller	129
8.1.1	System Tick (SysTick) Calibration Value Register	129
8.1.2	Interrupt and Exception Vectors	129
8.2	External Interrupt/Event Controller (EXTI).....	131
8.2.1	Main Features	132
8.2.2	Block Diagram	132
8.2.3	Wakeup Event Management	132
8.2.4	Function Overview	132
8.2.5	External Interrupt/Event Line Mapping.....	133
8.3	EXTI Registers Description.....	134
8.3.1	Interrupt Mask Register (EXTI_INTEN)	135
8.3.2	Event Mask Register (EXTI_EVTEN)	135
8.3.3	Rising Edge Trigger Selection Register (EXTI_RTRSEL)	136
8.3.4	Falling Edge Trigger Selection Register (EXTI_FTRSEL).....	136
8.3.5	Software Interrupt Event Register (EXTI_SWIE)	137
8.3.6	Pending Register (EXTI_PR).....	137
9	DMA Controller (DMA)	138
9.1	DMA Introduction	138
9.2	DMA Main Features	138
9.3	Function Overview.....	139
9.3.1	DMA Transaction	139
9.3.2	Arbiter	140
9.3.3	DMA Channels	140
9.3.4	Programmable Data Transfer Width, Alignment, and Endian	141

9.3.5	Error Management	142
9.3.6	Interrupts.....	142
9.3.7	DMA Request Mapping	143
9.4	DMA Registers.....	146
9.4.1	DMA Interrupt Status Register (DMA_ISTS)	148
9.4.2	DMA Interrupt Flag Clear Register (DMA_ICLR)	148
9.4.3	DMA Channel x Configuration Register (DMA_CHCTRLx) (x = 1 ... 7)	149
9.4.4	DMA Channel x Number of Data Register (DMA_TCNTx) (x = 1 ... 7).....	150
9.4.5	DMA Channel x Peripheral Address Register (DMA_CPBAx) (x = 1 ... 7)	151
9.4.6	DMA Channel x Memory Address Register (DMA_CMBAx) (x = 1 ... 7)	151
10	Timer	152
10.1	Basic Timer (TMR6 and TMR7)	153
10.1.1	TMR6 and TMR7 Introduction	153
10.1.2	TMR6 and TMR7 Main Features.....	153
10.1.3	TMR6 and TMR7 Function Overview	153
10.1.3.1	Time-base Unit	153
10.1.3.2	Prescaler.....	154
10.1.3.3	Counting Mode	155
10.1.3.4	Clock Source.....	158
10.1.3.5	Debug Mode.....	158
10.1.4	TMR6 and TMR7 Registers	159
10.1.4.1	TMR6 and TMR7 Control Register 1 (TMRx_CTRL1)	159
10.1.4.2	TMR6 and TMR7 Control Register 2 (TMRx_CTRL2)	160
10.1.4.3	TMR6 and TMR7 DMA/Interrupt Enable Register (TMRx_DIE)	161
10.1.4.4	TMR6 and TMR7 Status Register (TMRx_STS).....	161
10.1.4.5	TMR6 and TMR7 Event Generation Register (TMRx_EVEG).....	161
10.1.4.6	TMR6 and TMR7 Counter (TMRx_CNT)	162
10.1.4.7	TMR6 and TMR7 Prescaler (TMRx_DIV)	162
10.1.4.8	TMR6 and TMR7 Auto-reload Register (TMRx_AR)	162
10.2	General-purpose Timer (TMR2 to TMR5)	163
10.2.1	TMRx Introduction	163
10.2.2	TMRx Main Function	163
10.2.3	TMRx Function Overview	164
10.2.3.1	Time-base Unit	164
10.2.3.2	Counter Mode	165
10.2.3.3	Clock Selection	173
10.2.3.4	Capture/Compare Channel.....	175

10.2.3.5	Input Capture Mode.....	177
10.2.3.6	PWM Input Mode.....	177
10.2.3.7	Forced Output Mode	178
10.2.3.8	Output Compare Mode.....	179
10.2.3.9	PWM Mode	179
10.2.3.10	One-pulse Mode	182
10.2.3.11	Clearing OCxREF Signal on an External Event.....	183
10.2.3.12	Encoder Interface Mode	183
10.2.3.13	Timer Input XOR Function.....	185
10.2.3.14	Timer and External Trigger Synchronization	185
10.2.3.15	Timer Synchronization.....	188
10.2.3.16	Debug Mode.....	192
10.2.4	TMRx Registers Description	193
10.2.4.1	Control Register 1 (TMRx_CTRL1)	194
10.2.4.2	Control Register 2 (TMRx_CTRL2)	196
10.2.4.3	Slave Mode Control Register (TMRx_SMC)	196
10.2.4.4	DMA/Interrupt Enable Register (TMRx_DIE)	198
10.2.4.5	Status Register (TMRx_STS)	199
10.2.4.6	Event Generation Register (TMRx_EVEG)	200
10.2.4.7	Capture/Compare Mode Register 1 (TMRx_CCM1)	201
10.2.4.8	Capture/Compare Mode Register 2 (TMRx_CCM2)	203
10.2.4.9	Capture/Compare Enable Register (TMRx_CCE)	205
10.2.4.10	Counter (TMRx_CNT).....	206
10.2.4.11	Prescaler (TMRx_DIV).....	206
10.2.4.12	Auto-reload Register (TMRx_AR)	206
10.2.4.13	Capture/Compare Register 1 (TMRx_CC1)	208
10.2.4.14	Capture/Compare Register 2 (TMRx_CC2)	208
10.2.4.15	Capture/Compare Register 3 (TMRx_CC3)	209
10.2.4.16	Capture/Compare Register 4 (TMRx_CC4)	209
10.2.4.17	DMA Control Register (TMRx_DMAC).....	210
10.2.4.18	DMA Address in Burst Mode (TMRx_DMABA)	210
10.3	General-purpose Timer (TMR9 to TMR14).....	211
10.3.1	TMRx Introduction	211
10.3.2	TMRx Main Function	211
10.3.2.1	TMR9 and TMR12 Main Function	211
10.3.2.2	TMR10, TMR11, TMR13, and TMR14 Main Function	212
10.3.3	TMRx Function Overview	213
10.3.3.1	Time-base Unit	213

10.3.3.2	Counter Mode	214
10.3.3.3	Clock Selection	217
10.3.3.4	Capture/Compare Channel.....	219
10.3.3.5	Input Capture Mode.....	220
10.3.3.6	PWM Input Mode (Only TMR9 and TMR12)	221
10.3.3.7	Forced Output Mode	222
10.3.3.8	Output Compare Mode.....	222
10.3.3.9	PWM Mode	223
10.3.3.10	One-pulse Mode	224
10.3.3.11	Timer and External Trigger Synchronization (TMR9 and TMR12 Only)	225
10.3.3.12	Timer Synchronization (TMR9 and TMR12 Only)	227
10.3.3.13	Debug Mode.....	227
10.3.4	TMR9 and TMR12 Register Description	227
10.3.4.1	Control Register 1 (TMRx_CTRL1)	229
10.3.4.2	Slave Mode Control Register (TMRx_SMC)	229
10.3.4.3	DMA/Interrupt Enable Register (TMRx_DIE)	230
10.3.4.4	Status Register (TMRx_STS)	231
10.3.4.5	Event Generation Register (TMRx_EVEG)	232
10.3.4.6	Capture/Compare Mode Register 1 (TMRx_CCM1)	232
10.3.4.7	Capture/Compare Enable Register (TMRx_CCE)	235
10.3.4.8	Counter (TMRx_CNT)	236
10.3.4.9	Prescaler (TMRx_DIV).....	236
10.3.4.10	Auto-reload Register (TMRx_AR)	236
10.3.4.11	Capture/Compare Register 1 (TMRx_CC1)	236
10.3.4.12	Capture/Compare Register 2 (TMRx_CC2)	237
10.3.5	TMR10, TMR11, TMR13, and TMR14 Registers Description	237
10.3.5.1	Control Register 1 (TMRx_CTRL1)	238
10.3.5.2	DMA/Interrupt Enable Register (TMRx_DIE)	239
10.3.5.3	Status Register (TMRx_STS)	239
10.3.5.4	Event Generation Register (TMRx_EVEG)	240
10.3.5.5	Capture/Compare Mode Register 1 (TMRx_CCM1)	240
10.3.5.6	Capture/Compare Enable Register (TMRx_CCE)	242
10.3.5.7	Counter (TMRx_CNT)	242
10.3.5.8	Prescaler (TMRx_DIV).....	243
10.3.5.9	Auto-reload Register (TMRx_AR)	243
10.3.5.10	Capture/Compare Register 1 (TMRx_CC1)	243
10.4	Advanced-control Timer (TMR1, TMR8, and TMR15).....	244
10.4.1	TMR1, TMR8, and TMR15 Introduction	244

10.4.2	TMR1, TMR8, and TMR15 Main Features	244
10.4.3	TMR1, TMR8, and TMR15 Function Overview	245
10.4.3.1	Time-base Unit	245
10.4.3.2	Counter Mode	246
10.4.3.3	Repetition Counter	254
10.4.3.4	Clock Selection	255
10.4.3.5	Capture/Compare Channel.....	258
10.4.3.6	Input Capture Mode.....	260
10.4.3.7	PWM Input Mode.....	260
10.4.3.8	Forced Output Mode	261
10.4.3.9	Output Compare Mode.....	261
10.4.3.10	PWM Mode	262
10.4.3.11	Complementary Output and Dead-time Insertion.....	264
10.4.3.12	Using the Break Function.....	266
10.4.3.13	Clearing OCxREF Signal on an External Event.....	268
10.4.3.14	6-step PWM Output Generation	268
10.4.3.15	One-pulse Mode	269
10.4.3.16	Encoder Interface Mode	271
10.4.3.17	Timer Input XOR Function.....	272
10.4.3.18	Interfacing with Hall Sensors	272
10.4.3.19	TMRx Timer and External Trigger Synchronization	274
10.4.3.20	Timer Synchronization.....	277
10.4.3.21	Debug Mode.....	277
10.4.4	TMR1, TMR8, and TMR15 Registers Description	278
10.4.4.1	TMR1, TMR8, and TMR15 Control Register 1 (TMRx_CTRL1)	279
10.4.4.2	TMR1, TMR8, and TMR15 Control Register 2 (TMRx_CTRL2)	280
10.4.4.3	TMR1, TMR8, and TMR15 Slave Mode Control Register (TMRx_SMC)	282
10.4.4.4	TMR1, TMR8, and TMR15 DMA/Interrupt Enable Register (TMRx_DIE).....	283
10.4.4.5	TMR1, TMR8, and TMR15 Status Register (TMRx_STS).....	284
10.4.4.6	TMR1, TMR8, and TMR15 Event Generation Register (TMRx_EVEG).....	286
10.4.4.7	TMR1, TMR8, and TMR15 Capture/Compare Mode Register 1 (TMRx_CCM1)..	287
10.4.4.8	TMR1, TMR8, and TMR15 Capture/Compare Mode Register 2 (TMRx_CCM2)..	289
10.4.4.9	TMR1, TMR8, and TMR15 Capture/Compare Enable Register (TMRx_CCE) ..	291
10.4.4.10	TMR1, TMR8, and TMR15 Counter (TMRx_CNT)	293
10.4.4.11	TMR1, TMR8, and TMR15 Prescaler (TMRx_DIV)	293
10.4.4.12	TMR1, TMR8, and TMR15 Auto-reload Register (TMRx_AR)	293
10.4.4.13	TMR1, TMR8, and TMR15 Repetition Counter Register (TMRx_RC)	294
10.4.4.14	TMR1, TMR8, and TMR15 Capture/Compare Register 1 (TMRx_CC1).....	294

10.4.4.15	TMR1, TMR8, and TMR15 Capture/Compare Register 2 (TMRx_CC2)	295
10.4.4.16	TMR1, TMR8, and TMR15 Capture/Compare Register 3 (TMRx_CC3).....	295
10.4.4.17	TMR1, TMR8, and TMR15 Capture/Compare Register 4 (TMRx_CC4)	295
10.4.4.18	TMR1, TMR8, and TMR15 Break and Dead-time Register (TMRx_BRKDT)....	296
10.4.4.19	TMR1, TMR8, and TMR15 DMA Control Register (TMRx_DMAC)	297
10.4.4.20	TMR1, TMR8, and TMR15 DMA Address in Burst Mode (TMRx_DMABA)	298
11	Watchdog	299
11.1	Window Watchdog (WWDG)	299
11.1.1	WWDG Introduction	299
11.1.2	WWDG Main Features	299
11.1.3	WWDG Function Overview	299
11.1.4	How to Program Watchdog Timeout	300
11.1.5	Debug Mode	301
11.1.6	Register Description.....	302
11.1.6.1	Control Register (WWDG_CTRL)	302
11.1.6.2	Configuration Register (WWDG_CFG)	303
11.1.6.3	Status Register (WWDG_STS)	303
11.2	Independent Watchdog (IWDG)	304
11.2.1	Introduction	304
11.2.2	IWDG Main Features	304
11.2.3	IWDG Function Overview	304
11.2.3.1	Hardware Watchdog	304
11.2.3.2	Register Access Protection	304
11.2.3.3	Debug Mode.....	304
11.2.4	IWDG Register Description	306
11.2.4.1	Key Register (IWDG_KEY).....	306
11.2.4.2	Prescaler Register (IWDG_PR)	307
11.2.4.3	Reload Register (IWDG_RLD).....	307
11.2.4.4	Status Register (IWDG_STS)	308
12	Real-time Clock (RTC)	309
12.1	RTC Introduction	309
12.2	Main Features	309
12.3	Function Overview.....	309
12.3.1	Introduction	309
12.3.2	Reset Process	310
12.3.3	Reading RTC Registers	310
12.3.4	RTC Register Configuration	311

12.3.5 RTC Flag Assertion	311
12.4 RTC Register Description	312
12.4.1 RTC Control Register High (RTC_CTRLH)	313
12.4.2 RTC Control Register Low (RTC_CTRLL)	313
12.4.3 RTC Prescaler Load Register (RTC_DIVH/RTC_DIVL)	314
12.4.4 RTC Prescaler Divider Register (RTC_DIVCNTH/RTC_DIVCNTL)	315
12.4.5 RTC Counter Register (RTC_CNTH/RTC_CNTL)	316
12.4.6 RTC Alarm Register (RTC_ALAH/RTC_ALAL)	316
13 Analog-to-Digital Converter (ADC)	318
13.1 ADC Introduction	318
13.2 ADC Main Features	318
13.3 ADC Function Overview	319
13.3.1 ADC Switch	320
13.3.2 ADC Clock	320
13.3.3 Channel Selection	320
13.3.4 Single Conversion Mode	320
13.3.5 Continuous Conversion Mode	321
13.3.6 Timing Diagram	321
13.3.7 Analog Watchdog	321
13.3.8 Scan Mode	322
13.3.9 Injected Channel Management	322
13.3.10 Discontinuous Mode	323
13.3.11 Calibration	324
13.3.12 Data Alignment	325
13.3.13 Programmable Channel Sample Time	325
13.3.14 Conversion on External Trigger	325
13.3.15 DMA Request	328
13.3.16 Dual ADC Mode	328
13.3.16.1 Injected Simultaneous Mode	329
13.3.16.2 Regular Simultaneous Mode	330
13.3.16.3 Fast Interleaved Mode	330
13.3.16.4 Slow Interleaved Mode	331
13.3.16.5 Alternate Trigger Mode	332
13.3.16.6 Independent mode	332
13.3.16.7 Combined Regular/Injected Simultaneous Mode	333
13.3.16.8 Combined Regular Simultaneous + Alternate Trigger Mode	333
13.3.16.9 Combined Injected Simultaneous + Interleaved mode	334

13.3.17 Temperature Sensor.....	334
13.3.18 ADC Interrupts.....	335
13.4 ADC Registers.....	335
13.4.1 ADC Status Register (ADC_STS)	337
13.4.2 ADC Control Register 1 (ADC_CTRL1)	337
13.4.3 ADC Control Register 2 (ADC_CTRL2)	339
13.4.4 ADC Sample Time Register 1 (ADC_SMPT1)	342
13.4.5 ADC Sample Time Register 2 (ADC_SMPT2)	342
13.4.6 ADC Injected Channel Data Offset Register x (ADC_JOFSx) (x = 1...4)	343
13.4.7 ADC Watchdog High Threshold Register (ADC_WHTR)	343
13.4.8 ADC Watchdog Low Threshold Register (ADC_WLTR)	343
13.4.9 ADC Regular Sequence Register 1 (ADC_RSQ1)	344
13.4.10 ADC Regular Sequence Register 2 (ADC_RSQ2)	344
13.4.11 ADC Regular Sequence Register 3 (ADC_RSQ3)	345
13.4.12 ADC Injected Sequence Register (ADC_JSQ)	345
13.4.13 ADC Injected Data Register x (ADC_JDORx) (x = 1...4)	346
13.4.14 ADC Regular Data Register (ADC_RDOR)	346
14 Digital-to-Analog Converter (DAC).....	347
14.1 DAC Introduction.....	347
14.2 DAC Main Features	347
14.3 DAC Function Overview	348
14.3.1 DAC Channel Enable	348
14.3.2 DAC Output Buffer Enable	348
14.3.3 DAC Data Format	348
14.3.4 DAC Conversion	349
14.3.5 DAC Output Voltage	350
14.3.6 DAC Trigger Selection.....	350
14.3.7 DMA Request.....	351
14.3.8 Noise Generation	351
14.3.9 Triangle Wave Generation	352
14.4 Dual DAC Channel Conversion.....	353
14.4.1 Independent Trigger without Wave Generation	353
14.4.2 Independent Trigger with Same LFSR Generation	353
14.4.3 Independent Trigger with Different LFSR Generation	354
14.4.4 Independent Trigger with Same Triangle Generation	354
14.4.5 Independent Trigger with Different Triangle Generation	354

14.4.6	Simultaneous Software Start.....	354
14.4.7	Simultaneous Trigger without Wave Generation.....	355
14.4.8	Simultaneous Trigger with Same LFSR Generation	355
14.4.9	Simultaneous Trigger with Different LFSR Generation	355
14.4.10	Simultaneous Trigger with Same Triangle Generation	355
14.4.11	Simultaneous Trigger with Different Triangle Generation.....	356
14.5	DAC Registers.....	356
14.5.1	DAC Control Register (DAC_CTRL).....	357
14.5.2	DAC Software Trigger Register (DAC_SWTRG).....	359
14.5.3	DAC Channel 1 12-bit Right-aligned Data Holding Register (DAC_HDR12R1)	359
14.5.4	DAC Channel 1 12-bit Left-aligned Data Holding Register (DAC_HDR12L 1).....	360
14.5.5	DAC Channel 1 8-bit Right-aligned Data Holding Register (DAC_HDR8R1).....	360
14.5.6	DAC Channel 2 12-bit Right-aligned Data Holding Register (DAC_HDR12 R2)	360
14.5.7	DAC Channel 2 12-bit Left-aligned Data Holding Register (DAC_HDR12L 2)	361
14.5.8	DAC Channel 2 8-bit Right-aligned Data Holding Register (DAC_HDR8R2)	361
14.5.9	Dual DAC 12-bit Right-aligned Data Holding Register (DAC_HDR12RD).....	361
14.5.10	Dual DAC 12-bit Left-aligned Data Holding Register (DAC_HDR12LD)	362
14.5.11	Dual DAC 8-bit Right-aligned Data Holding Register (DAC_HDR8RD).....	362
14.5.12	DAC Channel 1 Data Output Register (DAC_ODT1)	363
14.5.13	DAC Channel 2 Data Output Register (DAC_ODT2)	363
15	I²C Interface	364
15.1	I ² C Introduction.....	364
15.2	I ² C Main Features	364
15.3	I ² C Function Overview.....	365
15.3.1	Mode Selection.....	365
15.3.2	I ² C Slave Mode.....	366
15.3.3	I ² C Master Mode	368
15.3.4	Error Condition.....	373
15.3.5	SDA/SCL Line Control	373
15.3.6	SMBus.....	374
15.3.7	DMA Request.....	376
15.3.8	Packet Error Checking (PEC).....	377
15.3.9	I ² C Interrupt Request.....	377
15.3.10	I ² C Debug Mode	378
15.4	I ² C Registers.....	379
15.4.1	Control Register 1 (I ² C_CTRL1).....	380

15.4.2	Control Register 2 (I^2C_CTRL2)	381
15.4.3	Own Address Register 1 (I^2C_OADDR1)	382
15.4.4	Own Address Register 2 (I^2C_OADDR2)	383
15.4.5	Data Register (I^2C_DT)	383
15.4.6	Status Register 1 (I^2C_STS1)	384
15.4.7	Status Register 2 (I^2C_STS2)	386
15.4.8	Clock Control Register ($I^2C_CLKCTRL$)	387
15.4.9	TMRISE Register (I^2C_TMRISE)	388
16	Universal Synchronous/Asynchronous Receiver/Transmitter (USART).....	389
16.1	USART Introduction	389
16.2	USART Main Features	389
16.3	USART Function Overview	390
16.3.1	USART Feature Description	391
16.3.2	Transmitter	392
16.3.2.1	Character Transmission	392
16.3.2.2	Configurable Stop Bit	393
16.3.2.3	Single Byte Communication	394
16.3.2.4	Break Frame	394
16.3.2.5	Idle Character	394
16.3.3	Receiver	395
16.3.3.1	Start Bit Detection	395
16.3.3.2	Character Reception	395
16.3.3.3	Break Frame	396
16.3.3.4	Idle Frame	396
16.3.3.5	Overrun Error	396
16.3.3.6	Framing Error	397
16.3.3.7	Configurable Stop Bits during Reception	398
16.3.4	Fractional Baud Rate Generation	398
16.3.4.1	How to Derive USARTDIV from USART_BAUDR Register Values	398
16.3.5	USART Receiver's Tolerance to Clock Deviation	399
16.3.6	Multiprocessor Communication	400
16.3.6.1	Idle Line Detection (WUMODE = 0)	400
16.3.6.2	Address Mark Detection (WUMODE = 1)	401
16.3.7	Parity Control	401
16.3.8	LIN (Local Interconnection Network) Mode	402
16.3.8.1	LIN Transmission	402
16.3.8.2	LIN Reception	402

16.3.9 USART Synchronous Mode	404
16.3.10 Single-wire Half-duplex Communication	406
16.3.11 Smartcard	407
16.3.12 IrDA SIR ENDEC Block	408
16.3.13 Continuous Communication Using DMA	410
16.3.13.1 Transmission Using DMA	410
16.3.13.2 Reception Using DMA.....	411
16.3.13.3 Error Flag and Interrupt Generation in Multi-buffer Communication	412
16.3.14 Hardware Flow Control.....	412
16.3.14.1 RTS Flow Control	412
16.3.14.2 CTS Flow Control	412
16.4 USART Interrupt Request.....	413
16.5 USART Mode Configuration	414
16.6 USART Registers	414
16.6.1 USART Register Map.....	414
16.6.2 Status Register (USART_STS).....	415
16.6.3 Data Register (USART_DT)	417
16.6.4 Baud Rate Register (USART_BAUDR)	417
16.6.5 Control Register 1 (USART_CTRL1).....	417
16.6.6 Control Register 2 (USART_CTRL2).....	419
16.6.7 Control Register 3 (USART_CTRL3).....	420
16.6.8 Guard Time and Prescaler Register (GTP)	423
17 Serial Peripheral Interface (SPI)	424
17.1 SPI Introduction	424
17.2 Main Features	424
17.2.1 SPI Main Features	424
17.2.2 I ² S Function Overview	424
17.3 Function Overview.....	425
17.3.1 SPI Function Overview.....	425
17.3.1.1 Introduction.....	425
17.3.1.2 Configure SPI in Slave Mode	428
17.3.1.3 Configure SPI in Master Mode.....	429
17.3.1.4 Configure SPI for Half-duplex Communication	430
17.3.1.5 Data Transmission and Reception.....	430
17.3.1.6 CRC Calculation	436
17.3.1.7 Status Flag	437
17.3.1.8 Disabling SPI	437

17.3.1.9	SPI Communication Using DMA	438
17.3.1.10	Error Flag	440
17.3.1.11	SPI Interrupt	440
17.3.2	I ² S Function Overview	441
17.3.2.1	I ² S Function Overview	441
17.3.2.2	Supported Audio Protocol	442
17.3.2.3	Clock Generator.....	448
17.3.2.4	I ² S Master Mode.....	450
17.3.2.5	I ² S Slave Mode.....	451
17.3.2.6	Status Flag	452
17.3.2.7	Error Flag	453
17.3.2.8	I ² S Interrupt	453
17.3.2.9	DMA Function.....	453
17.4	SPI Registers	454
17.4.1	SPI Control Register 1 (SPI_CTRL1) (Not Used in I ² S Mode)	455
17.4.2	SPI Control Register 2 (SPI_CTRL2)	456
17.4.3	SPI Status Register (SPI_STS)	457
17.4.4	SPI Data Register (SPI_DT)	458
17.4.5	SPICRC Polynomial Register (SPI_CPOLY) (Not Used in I ² S Mode).....	458
17.4.6	SPIRxCRC Register (SPI_RCRC) (Not Used in I ² S Mode).....	458
17.4.7	SPITxCRC Register (SPI_TCRC)	459
17.4.8	SPI_I2S Configuration Register (SPI_I2SCTRL)	459
17.4.9	SPI_I2S Prescaler Register (SPI_I2SCLKP)	461
18	CAN Bus Controller.....	462
18.1	Introduction.....	462
18.2	Main Features	462
18.3	Function Overview.....	462
18.3.1	CAN Overall Function Description	462
18.3.2	Operating Mode.....	464
18.3.2.1	Initialization Mode	464
18.3.2.2	Normal Mode	464
18.3.2.3	Sleep Mode (Low Power)	465
18.3.3	Test Mode	465
18.3.3.1	Silent Mode.....	465
18.3.3.2	Loopback Mode	466
18.3.3.3	Loopback and Silent Mode	467
18.3.4	AT32F403 in Debug Mode	467

18.3.5	Transmission Handling	467
18.3.6	Time-triggered Communication Mode.....	468
18.3.7	Reception Handling.....	468
18.3.8	Identifier Filtering	470
18.3.9	Message Storage.....	473
18.3.10	Error Management.....	474
18.3.11	Bit Timing.....	475
18.3.12	bxCAN Interrupt.....	478
18.4	CAN Registers.....	479
18.4.1	Register Access Protection	481
18.4.2	CAN Control and Status Register	481
18.4.2.1	CAN Main Control Register (CAN_MCTRL)	481
18.4.2.2	CAN Main Status Register (CAN_MSTS)	483
18.4.2.3	CAN Tx Status Register (CAN_TSTS)	484
18.4.2.4	CAN Receive FIFO 0 Register (CAN_RF0).....	485
18.4.2.5	CAN Receive FIFO 1 Register (CAN_RF1).....	486
18.4.2.6	CAN Interrupts Enable Register (CAN_INTEN)	487
18.4.2.7	CAN Error Status Register (CAN_ESTS)	488
18.4.2.8	CAN Bit Timing Register (CAN_BTMG)	489
18.4.3	CAN Mailbox Register.....	489
18.4.3.1	Tx Mailbox Identifier Register (CAN_TMI x) ($x = 0...2$)	490
18.4.3.2	Mailbox Data Length and Time Stamp Register (CAN_TDT x) ($x = 0...2$).....	491
18.4.3.3	Tx Mailbox Data Low Register (CAN_TDL x) ($x = 0...2$)	491
18.4.3.4	Tx Mailbox Data High Register (CAN_TD Hx) ($x = 0...2$)	492
18.4.3.5	Rx FIFO Mailbox Identifier Register (CAN_RFI x) ($x = 0...1$)	492
18.4.3.6	Rx FIFO Mailbox Data Length and Time Stamp Register (CAN_RDT x) ($x = 0...1$)	493
18.4.3.7	Rx FIFO Mailbox Data High Register (CAN_RDL x) ($x = 0...1$)	493
18.4.3.8	Rx FIFO Mailbox Data High Register (CAN_RDH x) ($x = 0...1$)	494
18.4.4	CAN Filter Register.....	494
18.4.4.1	CAN Filter Main Control Register (CAN_FM)	494
18.4.4.2	CAN Filter Mode Register (CAN_FM1)	494
18.4.4.3	CAN Filter Scale Register (CAN_FS1)	495
18.4.4.4	CAN Filter FIFO Assignment Register (CAN_FFA1)	496
18.4.4.5	CAN Filter Activation Register (CAN_FA1)	497
18.4.4.6	CAN Filter Bank i Register x (CAN_FBiRx) (where i = 0...13; x = 1...2)	497
19	External Memory Controller (XMC)	498
19.1	Introduction.....	498

19.2 Main Features	498
19.2.1 Block Diagram	498
19.2.2 AHB Interface	499
19.2.3 Supported Memories and Transactions	499
19.3 Function Overview.....	500
19.3.1 Address Mapping	500
19.3.1.1 NOR and PSRAM Address Mapping	500
19.3.1.2 NAND and PC Card Address Mapping	501
19.3.2 NOR Flash/PSRAM Controller	502
19.3.2.1 External Memory Interface Signal	503
19.3.2.2 Supported Memories and Transactions	504
19.3.2.3 Timing Rules.....	504
19.3.2.4 NOR Flash and PSRAM Controller Timing Diagram.....	505
19.3.2.5 Synchronous Transactions	517
19.3.3 NAND Flash/PC Card Controller	521
19.3.3.1 External Memory Interface Signal	521
19.3.3.2 NAND Flash/PC Card Supported Memories and Transaction	523
19.3.3.3 NAND Flash, ATA, and PC Card Timing Diagram	523
19.3.3.4 NAND Flash Operation.....	524
19.3.3.5 NAND Flash Pwait Functionality	525
19.3.3.6 NAND Flash ECC Calculation.....	526
19.4 XMC Registers	526
19.4.1 NOR Flash and PSRAM Controller Register	527
19.4.1.1 SRAM/NOR Flash Chip-select Control Register 1...4 (XMC_BK1CTRL1...4)	527
19.4.1.2 SRAM/NOR Flash Chip-select Timing Register 1...4 (XMC_BK1TMG1...4)	529
19.4.1.3 SRAM/NOR Flash Write Timing Register 1...4 (XMC_BK1TMGWR1...4).....	531
19.4.1.4 SRAM/NOR Additional Timing Register 1...4 (XMC_EXT1...4)	532
19.4.2 NAND Flash and PC Card Controller Register	532
19.4.2.1 PC Card/NAND Flash Control Register 2...4 (XMC_BK2...4CTRL).....	532
19.4.2.2 FIFO Status and Interrupt Register 2...4 (XMC_BK2...4STS)	534
19.4.2.3 Common Memory Space Timing Register 2...4 (XMC_BK2...4TMGMEM)	535
19.4.2.4 Attribute Memory Space Timing Register 2...4 (XMC_BK2...4TMGATT)	535
19.4.2.5 I/O Space Timing Register 4 (XMC_BK4TMGIO)	536
19.4.2.6 ECC Result Register 2/3 (XMC_BK2/3ECC)	538
20 SDIO Interface	539
20.1 Introduction.....	539
20.2 Main Features	539

20.3 Function Overview.....	541
20.3.1 SDIO Function Overview	541
20.3.1.1 SDIO Adapter.....	543
20.3.1.2 SDIO AHB Interface	550
20.3.2 Card Function Overview	551
20.3.2.1 Card Identification Mode.....	551
20.3.2.2 Card Reset	551
20.3.2.3 Operating Voltage Range Validation	551
20.3.2.4 Card Identification Process	551
20.3.2.5 Block Write	552
20.3.2.6 Block Read	553
20.3.2.7 Stream Access, Stream Write, and Stream Read (MultiMediaCard Only)	553
20.3.2.8 Erase: Group Erase and Sector Erase	554
20.3.2.9 Wide Bus Selection or Deselection	554
20.3.2.10 Protection Management	555
20.3.2.11 Card Status Register.....	557
20.3.2.12 SD Status Register.....	559
20.3.2.13 SD I/O Mode.....	562
20.3.2.14 Command and Response	563
20.3.3 Response Format	566
20.3.3.1 R1 (Normal Response Command).....	566
20.3.3.2 R1b.....	566
20.3.3.3 R2 (The CID and CSD Registers)	566
20.3.3.4 R3 (The OCR Register)	566
20.3.3.5 R4 (Fast I/O).....	567
20.3.3.6 R4b.....	567
20.3.3.7 R5 (Interrupt Request)	568
20.3.3.8 R6 (Interrupt Request)	568
20.3.4 SDIO I/O Card-specific Operation	568
20.3.4.1 SDIO I/O Read Wait Operation by SDIO_D2 Signaling	569
20.3.4.2 SDIO Read Wait Operation by Stopping SDIO_CK	569
20.3.4.3 SDIO Suspend/Resume Operation.....	569
20.3.4.4 SDIO Interrupts	569
20.3.5 CE-ATA Specific Operation	569
20.3.5.1 Command Completion Signal Disable.....	569
20.3.5.2 Command Completion Signal Enable	569
20.3.5.3 CE-ATA Interrupt	570

20.3.5.4	Aborting CMD61	570
20.3.6	Hardware Flow Control.....	570
20.4	SDIO Registers.....	570
20.4.1	SDIO Power Control Register (SDIO_POWER).....	572
20.4.2	SDIO Clock Control Register (SDIO_CLKCTRL).....	572
20.4.3	SDIO Argument Register (SDIO_ARG)	573
20.4.4	SDIO Command Register (SDIO_CMD)	573
20.4.5	SDIO Command Response Register (SDIO_RSPCMD)	574
20.4.6	SDIO Response 1...4 Register (SDIO_RSPx)	575
20.4.7	SDIO Data Timer Register (SDIO_DTTMR)	575
20.4.8	SDIO Data Length Register (SDIO_DTLEN).....	576
20.4.9	SDIO Data Control Register (SDIO_DTCTRL)	576
20.4.10	SDIO Data Counter Register (SDIO_DTCNTR)	577
20.4.11	SDIO Status Register (SDIO_STS)	578
20.4.12	SDIO Clear Interrupt Register (SDIO_INTCLR)	579
20.4.13	SDIO Interrupt Mask Register (SDIO_INTEN).....	580
20.4.14	SDIO BUF Counter Register (SDIO_BUFCNTR).....	582
20.4.15	SDIO Data BUF Register (SDIO_BUF)	582
21	Universal Serial Bus Full-speed Device Interface (USBDEV).....	584
21.1	Introduction.....	584
21.2	USB Main Features	584
21.3	Function Overview.....	585
21.3.1	USB Blocks Function Description	586
21.3.2	Generic USB Device Programming	587
21.3.2.1	System Reset and Power-on Reset.....	587
21.3.2.2	Double-buffered Endpoints.....	591
21.3.2.3	Isochronous Transfers	593
21.3.2.4	Suspend/Resume Events	593
21.4	USB Registers	595
21.4.1	Common Registers	597
21.4.1.1	USB Control Register (USB_CTRL)	597
21.4.1.2	USB Interrupt Status Register (USB_INTSTS).....	598
21.4.1.3	USB Frame Number Register (USB_FRNUM).....	600
21.4.1.4	USB Device Address Register (USB_DEVADR)	601
21.4.1.5	USB Packet Buffer Description Table Address Register (USB_BUFTBL)	601
21.4.2	Endpoint Registers	601
21.4.2.1	USB Endpoint n Register (USB_EPTn), n = [0...7]	601

21.4.3	Buffer Description Table Registers	604
21.4.3.1	Transmission Buffer Address Register n (USB_ADRn_TX)	604
21.4.3.2	Transmission Byte Count Register n (USB_CNTn_TX)	604
21.4.3.3	Reception Buffer Address Register n (USB_ADRn_RX)	605
21.4.3.4	Reception Byte Count Register n (USB_CNTn_RX)	605
22	MCU Debug (MCUDBG).....	607
22.1	Introduction	607
22.2	Function Overview	608
22.2.1	Debug Support for Low-power Mode	608
22.2.2	Debug Support for Timer, Watchdog, bxCAN, and I ² C.....	608
22.2.3	ID Code	608
22.2.4	SWJ Debug Port Pins	608
22.2.5	Trace Pin Assignment	608
22.3	MCUDBG Registers.....	609
22.3.1	MCUDBG Control Register (MCUDBG_CTRL)	610
23	Revision History	612

1 System Architecture

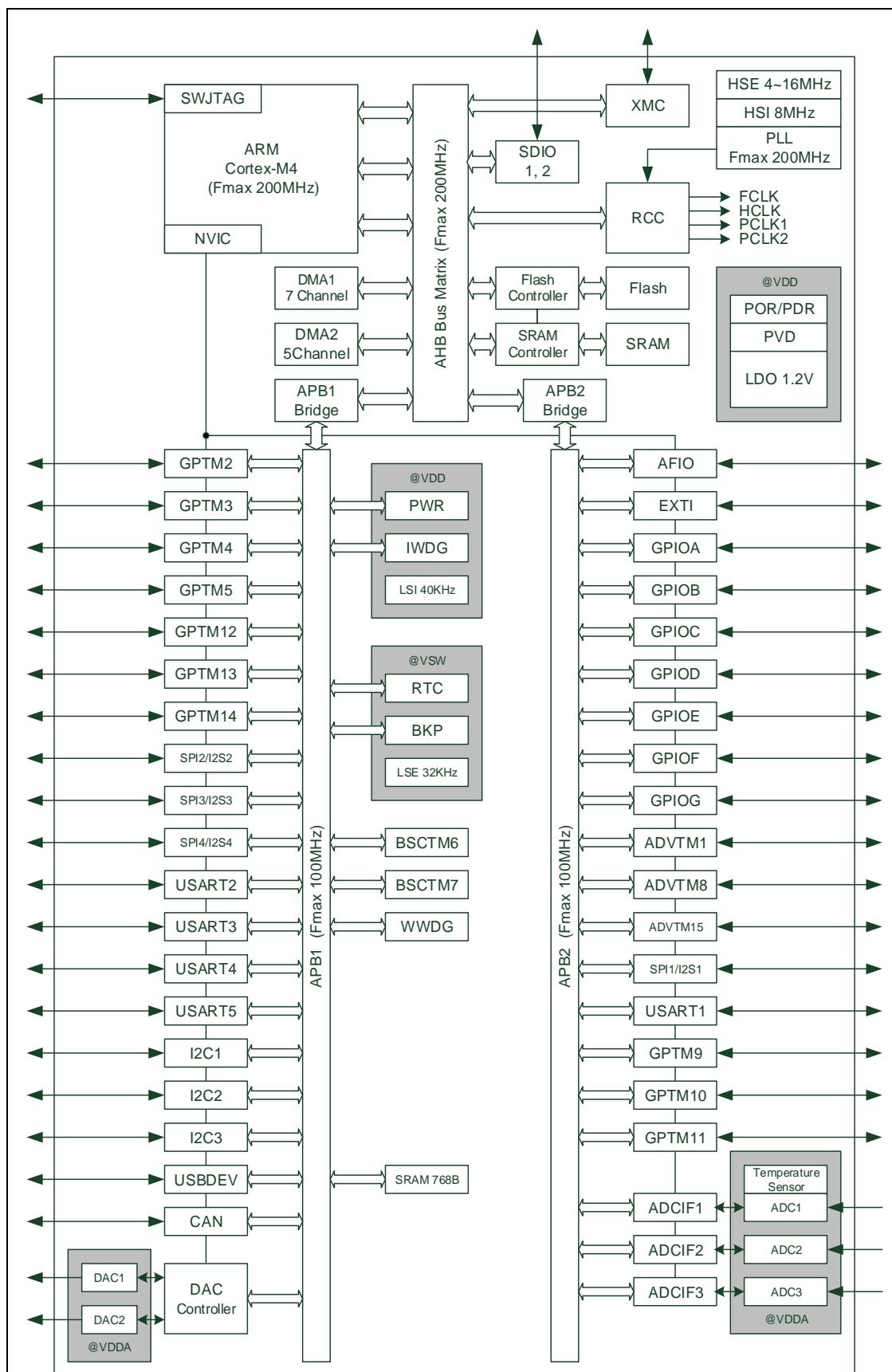
AT32F403 series microcontrollers consist of ARM® Cortex®-M4F processor core, bus architecture, peripherals, and memory. Cortex®-M4F process is a stage-of-the-art core, featuring many advanced functions. Compared with Cortex®-M3, Cortex®-M4F processor supports strengthened high-performance DSP instruction set, including extended single cycle 16-bit/32-bit multiply accumulate (MAC), dual 16-bit MAC instruction, optimized 8-bit/16-bit SIMD operation and saturation operation instruction, and single-precision (IEEE-754) float point unit (FPU). When Cortex®-M4F is designed with DSP function, it can save power and is faster than software solution. Cortex®-M4F will be applicable for those products and markets that require high performance and low power consumption.

1.1 System Introduction

AT32F403 main system is based on AHB bus matrix integration. AHB bus matrix is based on AMBA3.0 AHB-LITE multilayer AHB, enabling parallel access paths between masters and slaves in the system, ensuring the effectiveness of bus bandwidth. AHB bus matrix includes five masters: Cortex®-M4F core I-Code bus, D-Code bus, system bus, and two DMA controllers. Bus includes six slaves: Flash Memory Controller (FMC), on-chip SRAM, External memory controller (XMC), SDIO controller, and two APB bus bridges. System bus is used for loading/storing data and debug access of the system memory areas. System memory areas include on-chip SRAM area, external memory area, and peripheral map area.

System AHB bus is connected to all AHB peripherals; in addition, two AHB-APB bridges are included for full synchronous connection between system AHB bus and two APB buses. Two APB buses are connected to all APB peripherals. APB1/APB2 buses are limited to the maximum value of 100 MHz. The above-mentioned devices are interconnected through multilayer AHB bus architecture, as shown in [Figure 1-1](#):

Figure 1-1 AT32F403 Series Microcontrollers System Architecture



1.1.1 Bus Architecture

I-Code bus

- This bus connects the Instruction bus of the Cortex®-M4F core to the Flash memory instruction interface. Prefetching is performed on this bus.

D-Code bus

- This bus connects the D-Code bus (literal load and debug access) of the Cortex®-M4F core to the Flash memory data interface.

System bus

- This bus connects the system bus of the Cortex®-M4F (peripherals bus) to a BusMatrix which manages the arbitration between the core and the DMA.

DMA bus

- This bus connects the AHB master interface of the DMA to the BusMatrix which manages the access of CPU D-Code and DMA to SRAM, Flash memory, and peripherals.

Bus Matrix

- The BusMatrix manages the access arbitration between the core system bus and the DMA master bus. The arbitration uses a Round Robin algorithm. AHB peripherals are connected on system bus through a BusMatrix to allow DMA access.

AHB/APB Bridge (APB)

The two AHB/APB bridges provide full synchronous connections between the AHB and the 2 APB buses. APB1 and APB2 are limited to 100 MHz. Please refer to [Section 1.2](#) for the address mapping of the peripherals connected to each bridge. After each device reset, all peripheral clocks, except for the SRAM and FMC, are disabled. Before using a peripheral, users have to enable its clock in the RCC_AHBENR register.

Note: When a 16-bit or 8-bit access is performed on an APB register, the access will be transformed into a 32-bit access, and the bridge duplicates the 8-bit or 32-bit data to feed the 32-bit vector.

1.1.2 ARM Cortex®-M4F Processor

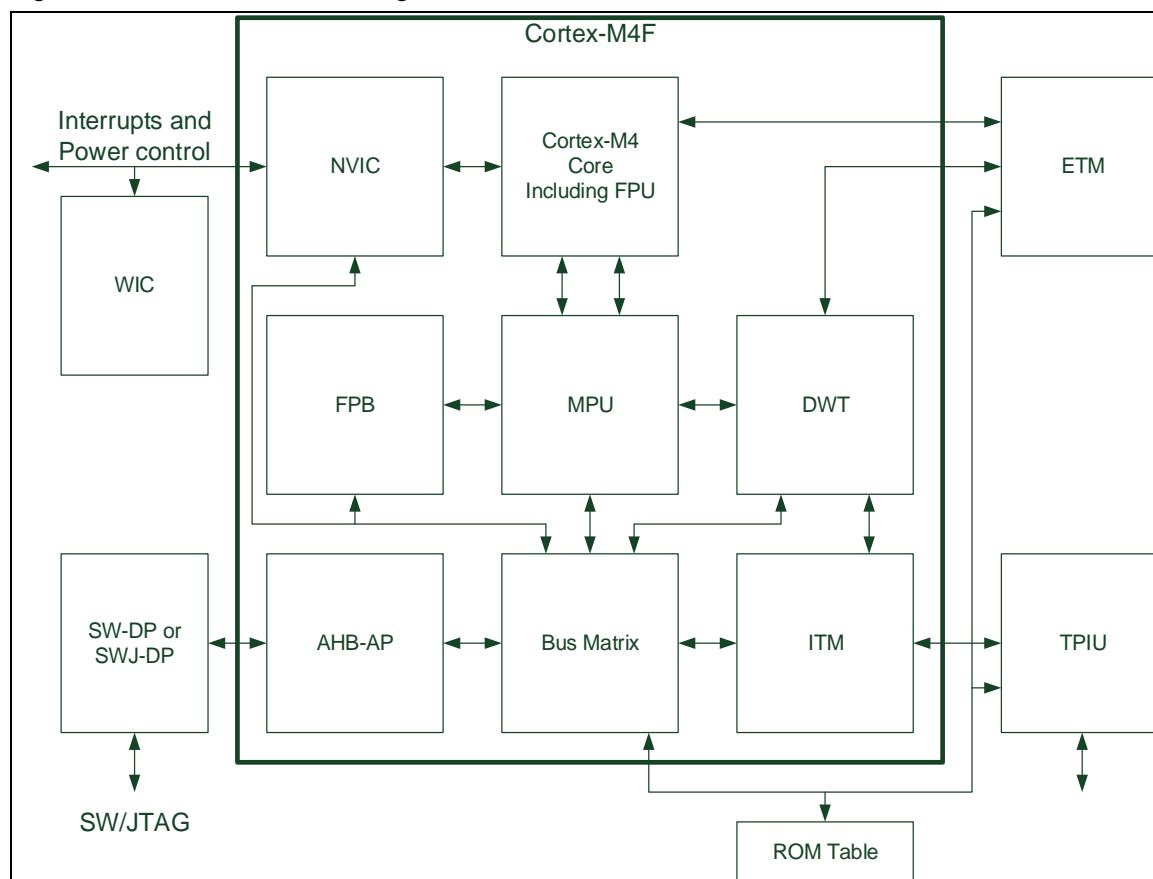
Cortex®-M4F processor is a low power consumption processor featuring low gate count, low interrupt latency, and low-cost debug. It supports DSP instruction set and FPU, and is applicable for deeply-embedded applications that require quick response to interruption. Cortex®-M4F processor is based on ARMv7-M architecture, and it supports both Thumb instruction set and DSP instruction set. Cortex®-M4F processor also provides the following systems and peripherals:

Internal bus matrix, which is used for the interconnect between I-Code bus, D-Code bus, System bus, Private Peripheral Bus (PPB), and debug access (AHB-AP):

- Nested Vectored Interrupt Controller (NVIC)
- Flash Patch and Breakpoint (FPB)
- Memory Protection Unit (MPU)
- Data Watchpoint and Trace (DWT)
- Instrument Trace Macrocell (ITM)
- Serial Wire JTAG Debug Port (SWJ-DP)
- Embedded Trace Macrocell (ETM)
- Trace Port Interface Unit (TPIU)

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

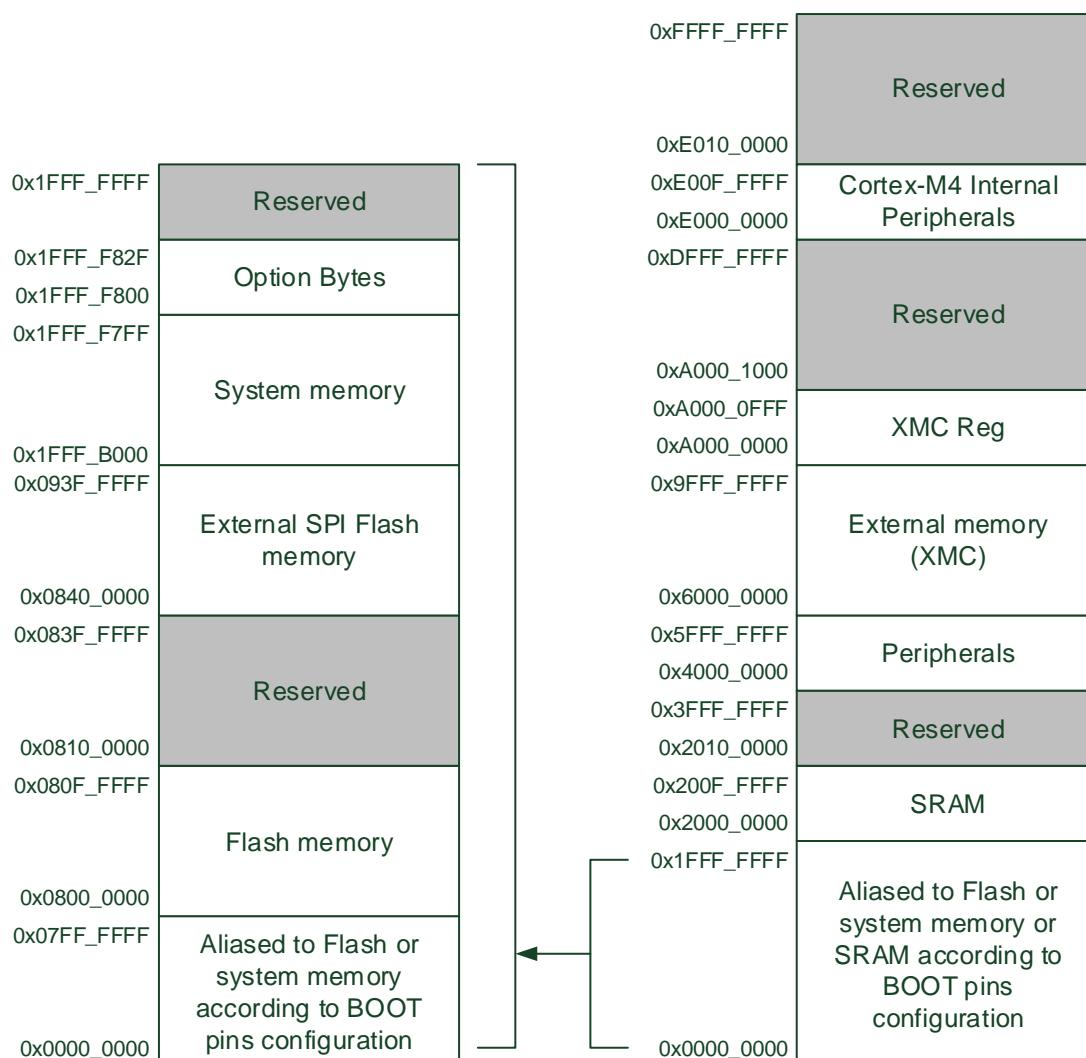


1.2 Address Map

Program memory, data memory, registers, and I/O ports are organized within the same linear 4-GB address space. The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the least significant byte of the word, and the highest numbered byte will be the most significant.

For the detailed mapping information of peripheral registers, please refer to the related sections. The addressable memory space is divided into 8 main blocks, each of which is 512 MB. All the memory areas that are not allocated to on-chip memories and peripherals are reserved address spaces. Please refer to the diagram of memory map listed in the corresponding data sheet of the user device.

Figure 1-3 AT32F403 Address Configuration



1.2.1 Register Map

Please refer to the diagram of memory map listed in the corresponding data sheet of the user device. [Table 1-1](#) lists all the peripheral boundary addresses in AT32F403.

Table 1-1 Register Boundary Address

Boundary Address	Peripherals	Bus	Register Map
0A000 1000 - 0xFFFF FFFF	Reserved	AHB	
0A000 0000 - 0xA000 0FFF	XMC_REG		Refer to Section 19.4
0x6000 0000 - 0x9FFF FFFF	XMC_MEM		Refer to Section 19.4
0x4002 8000 - 0x5FFF FFFF	Reserved		
0x4002 3400 - 0x4002 7FFF	SDIO2		Refer to Section 20.4
0x4002 3000 - 0x4002 33FF	CRC		Refer to Section 6.4
0x4002 2000 - 0x4002 23FF	Flash memory interface (FMC)		Refer to Section 5.4
0x4002 1400 - 0x4002 1FFF	Reserved		
0x4002 1000 - 0x4002 13FF	Reset and clock control (RCC)		Refer to Section 3.3
0x4002 0800 - 0x4002 0FFF	Reserved		
0x4002 0400 - 0x4002 07FF	DMA2		Refer to Section 9.4

0x4002 0000 - 0x4002 03FF	DMA1	APB2	Refer to Section 9.4
0x4001 8400 - 0x4001 7FFF	Reserved		
0x4001 8000 - 0x4001 83FF	SDIO		Refer to Section 20.4
0x4001 4400 - 0x4001 7FFF	Reserved		
0x4001 4000 - 0x4001 43FF	TMR15 Timer		Refer to Section 10.4.4
0x4001 3C00 - 0x4001 3FFF	ADC3		Refer to Section 13.4
0x4001 3800 - 0x4001 3BFF	USART1		Refer to Section 16.6
0x4001 3400 - 0x4001 37FF	TMR8 Timer		Refer to Section 10.4.4
0x4001 3000 - 0x4001 33FF	SPI1		Refer to Section 17.4
0x4001 2C00 - 0x4001 2FFF	TMR1 Timer		Refer to Section 10.4.4
0x4001 2800 - 0x4001 2BFF	ADC2		Refer to Section 13.4
0x4001 2400 - 0x4001 27FF	ADC1		Refer to Section 13.4
0x4001 2000 - 0x4001 23FF	GPIO Port G		Refer to Section 7.5
0x4001 1C00 - 0x4001 1FFF	GPIO Port F		Refer to Section 7.5
0x4001 1800 - 0x4001 1BFF	GPIO Port E		Refer to Section 7.5
0x4001 1400 - 0x4001 17FF	GPIO Port D		Refer to Section 7.5
0x4001 1000 - 0x4001 13FF	GPIO Port C		Refer to Section 7.5
0X4001 0C00 - 0x4001 0FFF	GPIO Port B		Refer to Section 7.5
0x4001 0800 - 0x4001 0BFF	GPIO Port A		Refer to Section 7.5
0x4001 0400 - 0x4001 07FF	EXTI		Refer to Section 8.3
0x4001 0000 - 0x4001 03FF	AFIO		Refer to Section 7.5
0x4000 7800 - 0x4000 FFFF	Reserved	APB1	
0x4000 7400 - 0x4000 77FF	DAC		Refer to Section 14.5
0x4000 7000 - 0x4000 73FF	Power control (PWR)		Refer to Section 2.4
0x4000 6C00 - 0x4000 6FFF	Backup registers (BKPR)		Refer to Section 4.4
0x4000 6800 - 0x4000 6BFF	I2C3		Refer to Section 15.4
0x4000 6400 - 0x4000 67FF	bxCAN1		Refer to Section 18.4
0x4000 6000 - 0x4000 63FF	Shared USB/CAN SRAM 512 bytes <small>(Note)</small>		Refer to Section 18.2 Refer to Section 21.2
0x4000 5C00 - 0x4000 5FFF	USB device FS registers		Refer to Section 21.4
0x4000 5800 - 0x4000 5BFF	I2C2		Refer to Section 15.4
0x4000 5400 - 0x4000 57FF	I2C1		Refer to Section 15.4
0x4000 5000 - 0x4000 53FF	UART5		Refer to Section 16.6
0x4000 4C00 - 0x4000 4FFF	UART4		Refer to Section 16.6
0x4000 4800 - 0x4000 4BFF	USART3		Refer to Section 16.6
0x4000 4400 - 0x4000 47FF	USART2		Refer to Section 16.6
0x4000 4000 - 0x4000 43FF	SPI4/I2S4		Refer to Section 17.4
0x4000 3C00 - 0x4000 3FFF	SPI3/I2S3		Refer to Section 17.4
0x4000 3800 - 0x4000 3BFF	SPI2/I2S2		Refer to Section 17.4
0x4000 3400 - 0x4000 37FF	Reserved		
0x4000 3000 - 0x4000 33FF	Independent watchdog (IWDG)		Refer to Section 11.2.4
0x4000 2C00 - 0x4000 2FFF	Window watchdog (WWDG)		Refer to Section 11.1.6

0x4000 2800 - 0x4000 2BFF	RTC	Refer to Section 12.4
0x4000 2400 - 0x4000 27FF	Reserved	
0x4000 2000 - 0x4000 23FF	TMR14 Timer	Refer to Section 10.3.4
0x4000 1C00 - 0x4000 1FFF	TMR13 Timer	Refer to Section 10.3.4
0x4000 1800 - 0x4000 1BFF	TMR12 Timer	Refer to Section 10.3.4
0x4000 1400 - 0x4000 17FF	TMR7 Timer	Refer to Section 10.1.4
0x4000 1000 - 0x4000 13FF	TMR6 Timer	Refer to Section 10.1.4
0x4000 0C00 - 0x4000 0FFF	TMR5 Timer	Refer to Section 10.2.4
0x4000 0800 - 0x4000 0BFF	TMR4 Timer	Refer to Section 10.2.4
0x4000 0400 - 0x4000 07FF	TMR3 Timer	Refer to Section 10.2.4
0x4000 0000 - 0x4000 03FF	TMR2 Timer	Refer to Section 10.2.4

Note: When USB SRAM configuration is 768 bytes, its address is 0x4000 7800~0x4000 7FFF.

1.2.2 Bit Banding

The Cortex®-M4F memory map includes two bit-band regions. These regions map each word in an alias region of memory to a bit in a bit-band region of memory. Writing to a word in the alias region has the same effect as a read-modify-write operation on the targeted bit in the bit-band region.

In the AT32F403 series, both peripheral registers and SRAM are mapped in a bit-band region.

This allows single bit-band write and read operations to be performed.

The following mapping formula shows how to reference each word in the alias region to a corresponding bit in the bit-band region. The mapping formula is:

$$\text{bit_word_addr} = \text{bit_band_base} + (\text{byte_offset} \times 32) + (\text{bit_number} \times 4)$$

where:

`bit_word_addr`: The address of the word in the alias memory region, and it maps to certain targeted bit.

`bit_band_base`: The starting address of the alias region

`byte_offset`: The number of the byte in the bit-band region that contains the targeted bit

`bit_number`: The bit position (0-31) of the targeted bit

Example:

The following example shows how to map bit 2 of the byte located at SRAM address 0x2000_0200 in the alias region:

$$\text{bit_word_addr} = 0x2200_0000 + (0x200 \times 32) + (2 \times 4) = 0x2200_4008$$

Writing to address 0x2200_4008 has the same effect as a read-modify-write operation on bit 2 of the byte at SRAM address 0x2000_0200. Reading address 0x2200_4008 returns the value (0x0000_0001 or 0x0000_0000) of bit 2 of the byte at SRAM address 0x2000_0200. For more information on bit-banding, please refer to “ARM® Cortex-M4 Technical Reference Manual.”

1.2.3 On-chip SRAM

The AT32F403 series microcontrollers contain up to 96 KB of on-chip SRAM which starts at the address of 0x2000_0000. It supports byte, half-word (16 bits), and word (32 bits) accesses. In addition, AT32F403 series microcontrollers provide a special mode which can extend on-chip SRAM to 224 KB. Users can use this extension mode through option byte EOPB0 0 bit of extension setting (Please refer to [Section 5.3.4.](#)). In 224 KB extension mode, Flash memory limit of zero wait state is 128 KB.

1.2.4 On-chip Flash

AT32F403 series provide up to 1024 KB of on-chip Flash memory, supporting zero wait state single cycle 32-bit read operation. The Flash memory is divided into main memory block and information block. The main memory block is used for storing application code; it can be accessed by bytes (8-bit aligned), half-words (16-bit aligned), or full words (32-bit aligned). Flash memory can be programmed as 16 bits (half-word) or 32 bits (full words) at a time. The main block consists of pages, each of them is

2 KB for 256K and above, and 1KB for 128K and below. The Flash memory erase operation can be performed at page level or on the whole Flash area (mass-erase) (The mass-erase does not affect the information blocks.). Information block contains system memory and option byte. System memory is used for storing boot loader, which is programmed in the factory, and users cannot modify it. Option bytes store the options that users can modify. For detailed information, please refer to Table 1-2.

Flash memory is controlled by Flash memory controller. The controller provides prefetch buffer, option byte loader, Flash program/erase operation, and Read/Write protection. Please refer to [Chapter 5](#) for more details about Flash memory controller and register configuration.

Table 1-2 Flash Memory Module Organization (1024K)

Block	Name	Address Range	Size (bytes)
Main Memory Block	Page0	0x0800 0000 – 0x0800 07FF	2 K
	Page1	0x0800 0800 – 0x0800 0FFF	2 K
	Page2	0x0800 1000 – 0x0800 17FF	2 K
	Page3	0x0800 1800 – 0x0800 1FFF	2 K
	Page4	0x0800 2000 – 0x0800 27FF	2 K
	.	.	.
	Page255	0x0807 F800 – 0x0807 FFFF	2 K
	Bank 2 512 KB	Page256	0x0808 0000 – 0x0808 07FF
		Page257	0x0808 0800 – 0x0808 0FFF
		Page258	0x0808 1000 – 0x0808 17FF
		Page259	0x0808 1800 – 0x0808 1FFF
		Page260	0x0808 2000 – 0x0808 27FF
		.	.
		Page511	0x080F F800 – 0x080F FFFF
		.	.
External Memory	Bank 3	0x0840 0000 – 0x093F FFFF	16 M
Information Block	Boot Loader	0x1FFF B000 – 0x1FFF F7FF	18 K
	User Option bytes	0x1FFF F800 – 0x1FFF F80F	16
Flash Memory Interface Register	FLASH_ACR	0x4002 2000 – 0x4002 2003	4
	FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
	FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
	FLASH_STS	0x4002 200C – 0x4002 200F	4
	FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
	FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
	Reserved	0x4002 2018 – 0x4002 201B	4
	FLASH_UOB	0x4002 201C – 0x4002 201F	4
	FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
	Reserved	0x4002 2024 - 0x4002 2043	32
	FLASH_FCKEY2	0x4002 2044 - 0x4002 2047	4
	Reserved	0x4002 2048 - 0x4002 204B	4
	FLASH_STS2	0x4002 204C - 0x4002 204F	4
	FLASH_CTRL2	0x4002 2050 - 0x4002 2053	4
	FLASH_ADDR2	0x4002 2054 - 0x4002 2057	4
	Reserved	0x4002 2058 - 0x4002 2083	44
	FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
	Reserved	0x4002 2088 - 0x4002 208B	4
	FLASH_STS3	0x4002 208C - 0x4002 208F	4
	FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
	FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4

Flash Memory Module Organization (512K)

Block		Name	Address Range	Size (bytes)	
Main Memory Block	Bank1 512KB	Page 0	0x0800 0000 – 0x0800 07FF	2K	
		Page 1	0x0800 0800 – 0x0800 0FFF	2K	
		Page 2	0x0800 1000 – 0x0800 17FF	2K	
		Page 3	0x0800 1800 – 0x0800 1FFF	2K	
		Page 4	0x0800 2000 – 0x0800 27FF	2K	
		.	.	.	
		页 255	0x0807 F800 – 0x0807 FFFF	2K	
External Memory	Bank 3		0x0840 0000 – 0x093F FFFF	16M	
Information Block		Boot Loader	0x1FFF B000 – 0x1FFF F7FF	18K	
		User Option bytes	0x1FFF F800 – 0x1FFF F82F	48	
Flash Memory Interface Register		FLASH_ACR	0x4002 2000 – 0x4002 2003	4	
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4	
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4	
		FLASH_STS	0x4002 200C – 0x4002 200F	4	
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4	
		FLASH_ADDR	0x4002 2014 – 0x4002 2017	4	
		Reserved	0x4002 2018 – 0x4002 201B	4	
		FLASH_UOB	0x4002 201C – 0x4002 201F	4	
		FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4	
		Reserved	0x4002 2024 - 0x4002 2083	96	
		FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4	
		FLASH_SELECT	0x4002 2088 - 0x4002 208B	4	
		FLASH_STS3	0x4002 208C - 0x4002 208F	4	
		FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4	
		FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4	
		FLASH_DA	0x4002 2098 – 0x4002 209B	4	

Flash Memory Module Organization (256K)

Block		Name	Address Range	Size (bytes)	
Main Memory Block	Bank1 256KB	Page 0	0x0800 0000 – 0x0800 07FF	2K	
		Page 1	0x0800 0800 – 0x0800 0FFF	2K	
		Page 2	0x0800 1000 – 0x0800 17FF	2K	
		Page 3	0x0800 1800 – 0x0800 1FFF	2K	
		Page 4	0x0800 2000 – 0x0800 27FF	2K	
		.	.	.	
		Page 127	0x0803 F800 – 0x0803 FFFF	2K	
External Memory	Bank 3		0x0840 0000 – 0x093F FFFF	16M	
Information Block		Boot Loader	0x1FFF B000 – 0x1FFF F7FF	18K	
		User Option bytes	0x1FFF F800 – 0x1FFF F82F	48	
Flash Memory Interface Register		FLASH_ACR	0x4002 2000 – 0x4002 2003	4	
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4	
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4	
		FLASH_STS	0x4002 200C – 0x4002 200F	4	
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4	
		FLASH_ADDR	0x4002 2014 – 0x4002 2017	4	
		Reserved	0x4002 2018 – 0x4002 201B	4	
		FLASH_UOB	0x4002 201C – 0x4002 201F	4	
		FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4	
		Reserved	0x4002 2024 - 0x4002 2083	96	
		FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4	
		FLASH_SELECT	0x4002 2088 - 0x4002 208B	4	
		FLASH_STS3	0x4002 208C - 0x4002 208F	4	
		FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4	
		FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4	
		FLASH_DA	0x4002 2098 – 0x4002 209B	4	

Flash Memory Module Organization (128K)

Block		Name	Address Range	Size (bytes)
Main Memory Block	Bank1 128KB	Page 0	0x0800 0000 – 0x0800 03FF	1K
		Page 1	0x0800 0400 – 0x0800 07FF	1K
		Page 2	0x0800 0800 – 0x0800 0BFF	1K
		.	.	.
		Page 127	0x0801 FC00 – 0x0801 FFFF	1K
External Memory	Bank 3		0x0840 0000 – 0x093F FFFF	16M
Information Block		Boot Loader	0x1FFF B000 – 0x1FFF F7FF	18K
Flash Memory Interface Register		User Option bytes	0x1FFF F800 – 0x1FFF F82F	48
FLASH_ACR		0x4002 2000 – 0x4002 2003	4	
FLASH_FCKEY		0x4002 2004 – 0x4002 2007	4	
FLASH_OPTKEYR		0x4002 2008 – 0x4002 200B	4	
FLASH_STS		0x4002 200C – 0x4002 200F	4	
FLASH_CTRL		0x4002 2010 – 0x4002 2013	4	
FLASH_ADDR		0x4002 2014 – 0x4002 2017	4	
Reserved		0x4002 2018 – 0x4002 201B	4	
FLASH_UOB		0x4002 201C – 0x4002 201F	4	
FLASH_WRPRT		0x4002 2020 – 0x4002 2023	4	
Reserved		0x4002 2024 - 0x4002 2083	96	
FLASH_FCKEY3		0x4002 2084 - 0x4002 2087	4	
FLASH_SELECT		0x4002 2088 - 0x4002 208B	4	
FLASH_STS3		0x4002 208C - 0x4002 208F	4	
FLASH_CTRL3		0x4002 2090 - 0x4002 2093	4	
FLASH_ADDR3		0x4002 2094 - 0x4002 2097	4	
FLASH_DA		0x4002 2098 – 0x4002 209B	4	

1.3 Boot Configuration

AT32F403 provides three kinds of boot sources which can be selected through the BOOT[1:0] pins. The details are shown in Table 1-3.

Table 1-3 Boot Mode

Boot Mode	Description	Boot Mode Selection Pins	
		BOOT1	BOOT0
Main Flash memory	Main Flash memory is selected as boot space.	X	0
System memory	System memory is selected as boot space.	0	1
On-chip SRAM	Embedded SRAM is selected as boot space.	1	1

The values on the BOOT pins are latched after reset. Users can set the BOOT1 and BOOT0 pins after reset, to select the required boot mode. The BOOT pins are re-sampled when exiting from the Standby mode. Therefore, they must be kept in the required boot mode configuration in the Standby mode.

Depending on the selected boot mode, main Flash memory, system memory, or SRAM are accessible as follows:

- Boot from main Flash memory: The main Flash memory is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x800 0000). In other words, the Flash memory contents can be accessed starting from address 0x0000 0000 or 0x800 0000.

- Boot from system memory: The system memory is aliased in the boot memory space (0x0000 0000), but it is still accessible from its original memory space (0x1FFF F000 in other devices).
- Boot from the embedded SRAM: SRAM is accessible only at address 0x2000 0000.

After startup delay, CPU fetches the top-of-stack value from address 0x0000 0000, and it starts executing code from the designated address 0x0000_0004 of boot memory. Due to its fixed memory map, the code area starts from address 0x0000 0000 (accessed through the I-Code/D-Code buses) while the data area (SRAM) starts from address 0x2000 0000 (accessed through the system bus). Cortex®-M4F CPU always fetches the reset vector on the I-Code bus, which implies that boot space is available only in the code area (typically, Flash memory).

AT32F403 series microcontrollers implement a special mechanism which enables booting from on-chip SRAM. When booting from on-chip SRAM, in the application initialization code, NVIC exception table and offset register must be used to relocate the vector table in SRAM.

System memory contains the embedded boot loader, which can be used for programming the Flash memory. Boot loader reprograms the Flash memory through USART1, USART2 or USB interface.

1.4 Device Characteristics Information

1.4.1 Description of Register Abbreviations

The following abbreviations are used in the register descriptions of this manual:

Table 1-4 List of Register Abbreviations

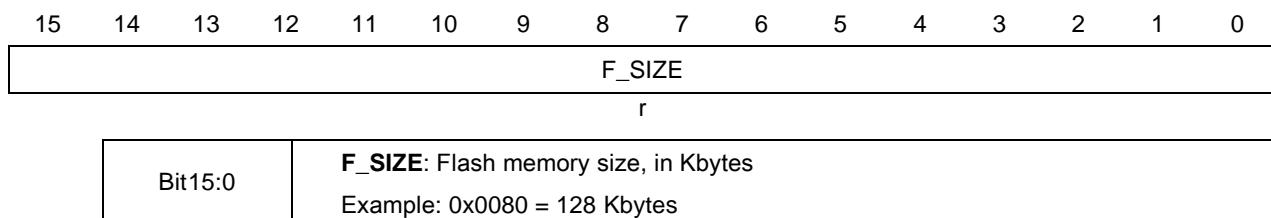
read/write (rw)	Software can read and write to these bits.
read-only (r)	Software can only read these bits.
write-only (w)	Software can only write to this bit. Reading the bit returns the reset value.
read/clear (rc_w0)	Software can read as well as clear this bit by writing 0. Writing '1' has no effect on the bit value.
read/set (rs)	Software can read as well as set this bit by writing 1. Writing '0' has no effect on the bit value.
Reserved (res)	Reserved bit; it must be kept at reset value.

1.4.2 Flash Memory Size Register

Flash memory size register can retrieve the information about chip Flash memory size, and users can get the Flash memory size through this register.

Base address: 0xFFFF F7E0

Reset value: 0xFFFF (Factory-programmed)



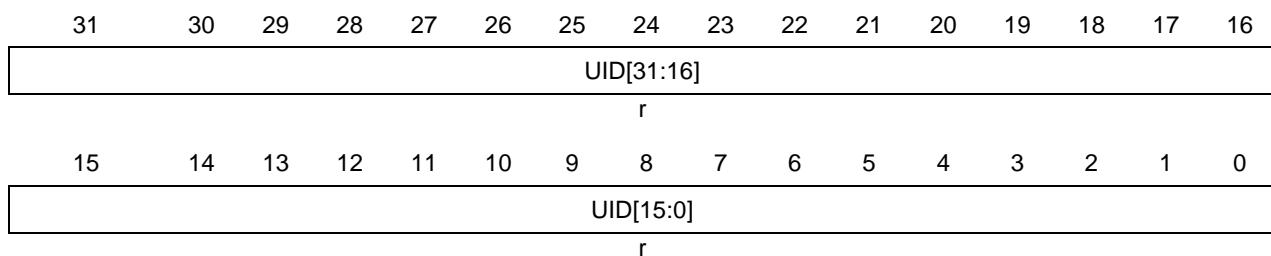
1.4.3 Device Electronic Signature

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

- Serial numbers; such as USB string serial numbers
- Part of security keys

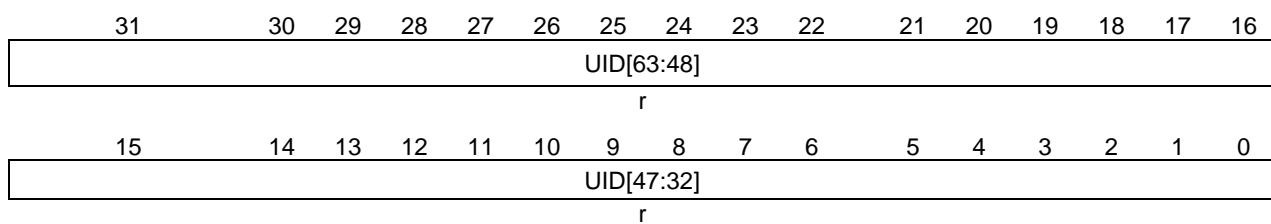
Base address: 0xFFFF F7E8

Reset value: 0xFFFF XXXX (Factory-programmed)



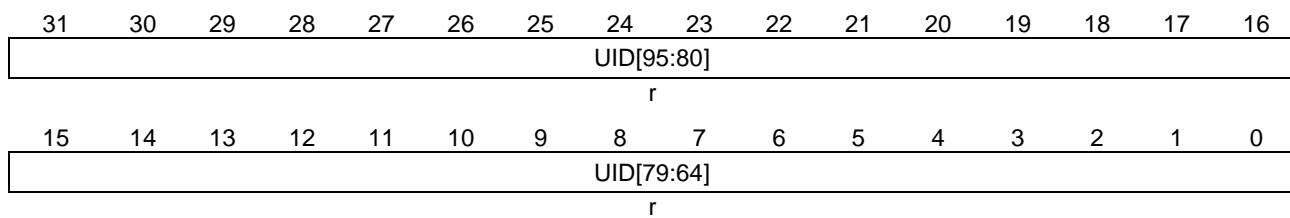
Base address: 0x1FFF F7EC

Reset value: 0xXXXX XXXX (Factory-programmed)



Base address: 0x1FFF F7F0

Reset value: 0xXXXX XXXX (Factory-programmed)



2 Power Control (PWR)

2.1 Introduction

Power consumption is one of the most important issue in AT32F403 series devices. The operating voltage supply is 2.6 V ~ 3.6 V, and it can function normally within -40°C ~ +85°C. To reduce power consumption and to achieve the best tradeoff among the conflicting demands of CPU operating time, speed, and power consumption, the Power Management Unit (PMU) provides three types of power saving modes, including Sleep, Deep-sleep, and Standby mode. For AT32F403 series devices, there are three power domains, including VDD/VDDA domain, 1.2 V domain, and backup domain. The power of VDD/VDDA domain is supplied directly by external power; however, VDDA and VSSA provide separated analog module power to reduce noise on external power. An embedded LDO in the VDD/VDDA domain is used to supply the 1.2 V domain power.

2.2 Main Features

Three power domains: VDD/VDDA domain, core domain, and backup domain.

Three types of power saving modes: Sleep mode, Deep-sleep mode, and Standby mode.

Internal voltage regulator supplies 1.2 V voltage source for core domain.

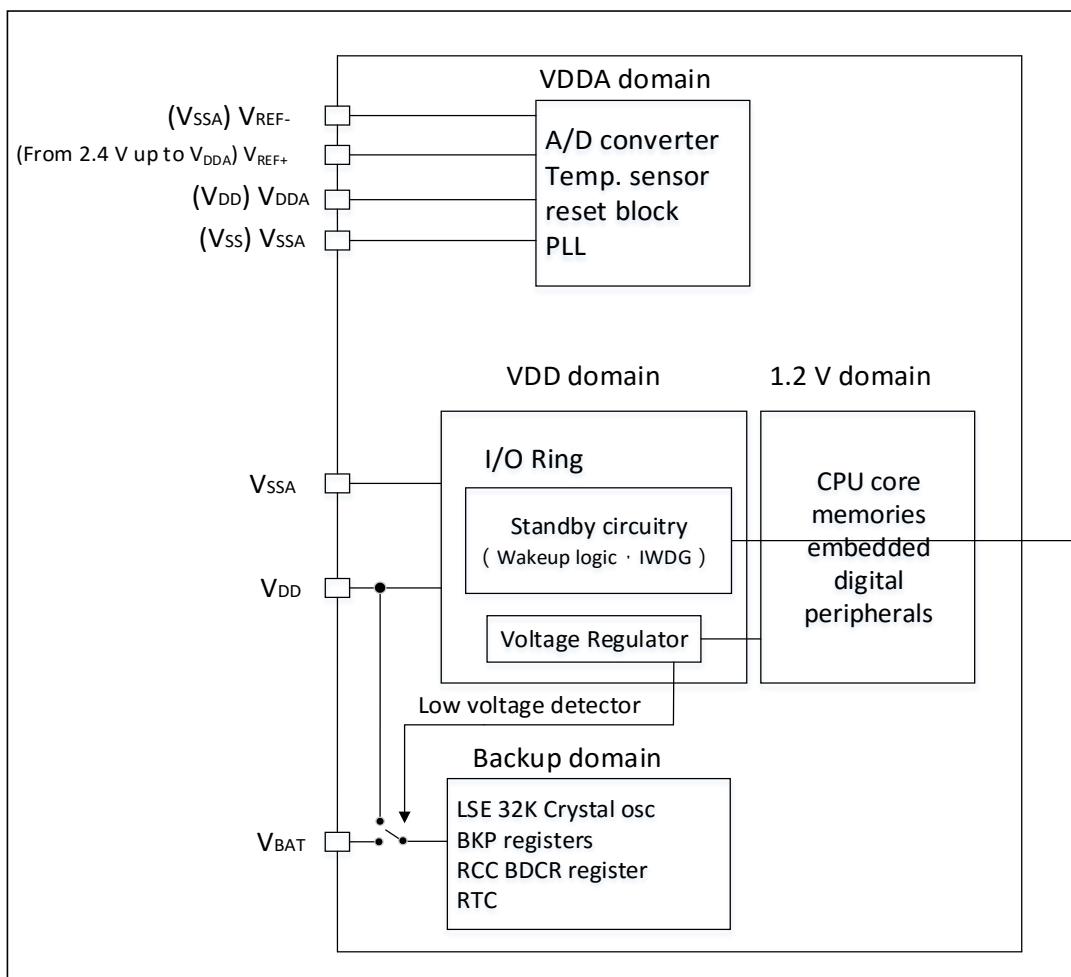
Low voltage detector is provided to issue an interrupt when the supply voltage is lower than a programmed threshold.

2.3 Function Overview

2.3.1 Power Supply

The operating voltage (VDD) of AT32F403 is 2.6 V ~ 3.6 V. An embedded regulator is used to supply the internal 1.2 V power.

Figure 2-1 Block Diagram of Each Power Supply



2.3.1.1 VDD/VDDA Power Domain

The operating voltage (VDD) of AT32F403 is 2.6 V ~ 3.6 V. An embedded regulator is used to supply the internal 1.2 V power.

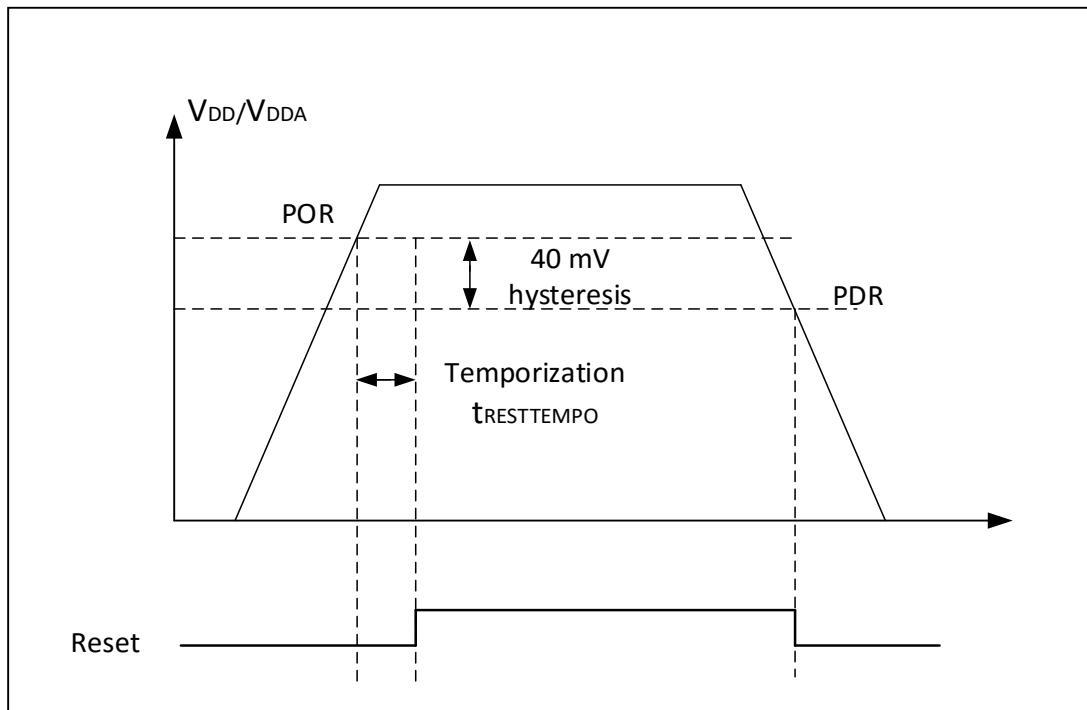
To increase the conversion accuracy, the ADC has an independent power supply which can filter and shield from noises on the PCB. The ADC voltage supply input is available on a separate VDDA pin. An isolated supply ground connection is provided on the VSSA pin.

When available (according to package), VREF- must be connected to VSSA.

On 100-pin and 144-pin packages, user can connect a separate external reference voltage ADC to VREF+ pin and VREF- pin, to ensure a better accuracy on low-voltage inputs. The voltage on VREF+ can range from 2.4 V to VDDA. On 64-pin packages and packages with less pins, VREF+ and VREF- pins are not available; they are internally connected to the ADC voltage supply (VDDA) and ground (VSSA).

AT32F403 has an integrated internal POR/PDR circuitry that allows proper operation when the supply voltage reaches 2.6. When VDD/VDDA is below a specified threshold VPOR/VPDR, the device remains in Reset mode, without the need for an external reset circuit. For more details concerning the power-on/power-down reset, please refer to the electrical characteristics section of the data sheet.

Figure 2-2 Power-on Reset/Power-down Reset Waveform

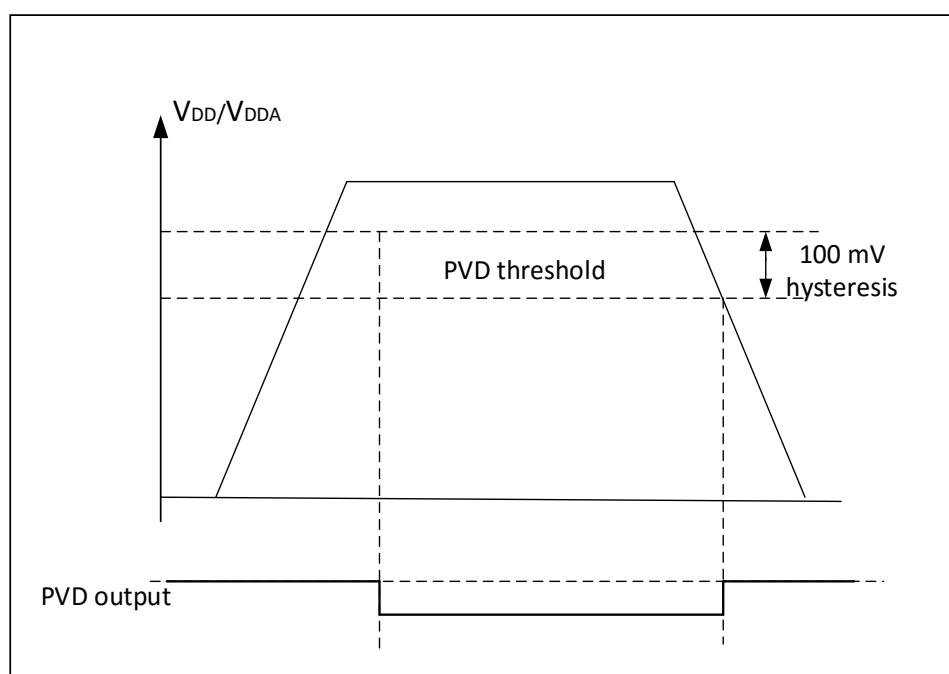


AT32F403 provides a programmable voltage detector (PWD). Users can use this PVD to monitor VDD power supply by comparing it to the PVDS[2:0] bits in the power control register (PWR_CTRL), selecting the threshold for voltage monitor.

The PVD is enabled by setting the PVDEN bit. The PVD bit in the power control/status register (PWR_CTRLSTS) indicates whether VDD is higher or lower than the PVD threshold.

This event is internally connected to the EXTI line 16 and can generate an interrupt if it is enabled through the EXTI registers. When VDD drops below the PVD threshold and/or when VDD rises above the PVD threshold, the PVD output interrupt will be generated depending on the rising/falling edge of EXTI line16 configuration. This characteristic can be used to perform emergency power-down tasks.

Figure 2-3 PVD Threshold and Outputs



2.3.1.2 Core Power Domain

Core power domain includes CPU core, memory, and embedded digital peripherals. This power domain is supplied by a voltage regulator. The voltage regulator is always enabled after reset. It works in three different modes depending on the application modes.

Run mode: The regulator supplies full power to the 1.2 V domain (core, memories, and digital peripherals).

Stop mode: The regulator supplies low power to the 1.2 V domain, preserving contents of registers and SRAM.

Standby mode: The regulator is powered off. The contents of the registers and SRAM are lost except for the backup circuitry and the backup domain.

Battery backup domain

The VBAT pin also supplies power to the RTC unit, the LSE oscillator, and the PC13 to PC15 IOs.

Note: At VDD startup phase ($t_{RSTTEMPO}$) or after PVD is detected, the power switch between VBAT and VDD remains connected to VBAT. At the startup phase, if VDD is established in less than $t_{RSTTEMPO}$ (Please refer to the data sheet for the value of $t_{RSTTEMPO}$) and $VDD > VBAT + 0.6$ V, a current may be injected into VBAT through an internal diode connected between VDD and the power switch (VBAT). If the power supply/battery connected to the VBAT pin cannot support this current injection, connecting an external low-drop diode between this power supply and the VBAT pin is strongly recommended.

Connecting VBAT externally to VDD with a 100 nF ceramic decoupling capacitor is recommended. When the backup domain is supplied by VDD (analog switch connected to VDD), the following functions are available:

- PC14 and PC15 can be used as either GPIO or LSE pins.
- PC13 can be used as GPIO, TAMPER pin, RTC calibration clock, RTC alarm or second output (Please refer to [Chapter 4](#)).

Note: Due to the fact that the analog switch only sinks a limited amount of current (3 mA), the use of GPIOs PC13 to PC15 in the output mode is restricted: the speed has to be limited to 2 MHz with a maximum load of 30 pF, and these IOs must not be used as a current source (e.g. To drive a LED).

The backup domain is supplied by VDD/VBAT for the following functions:

- PC14 and PC15 can only be used as LSE pins.
- PC13 can be used as TAMPER pin, RTC alarm, or second output (Please refer to [Section 4.4.2](#)).

2.3.2 Low-power Mode

After a system or a power reset, the microcontroller is in Run mode by default. When the CPU does not need to be kept running, there are several low-power modes available to save power. For example, when waiting for an external event, users can select the mode that gives the best compromise according to the low-power consumption, short startup time, and available wakeup sources.

AT32F403 provides three low-power modes:

- Sleep mode (Cortex®-M4F CPU clock off, all peripherals including Cortex®-M4F core peripherals like NVIC, SysTick, etc. are kept running)
- Stop mode (All clocks are stopped except for systick.)
- Standby mode (1.2 V domain powered-off)

Mode	Entry	Wakeup	Effect on 1.2 V Domain Clocks	Effect on VDD Domain Clocks	Voltage Regulator
Sleep (SLP-NOW or SLP-ON-EXIT)	WFI	Any interrupt	CPU clock OFF, no effect on other clocks or ADC clock sources	None	ON
	WFE	Wakeup event			

Stop	PDDS bit + SLEEPDEEP bit + WFI or WFE	Any EXTI line (configured in the EXTI registers)	All 1.2 V domain clocks are OFF.	HSI and HSE oscillators are OFF.	ON or in low-power mode (depends on the power control register (PWR_CTRL))
Standby	PDDS bit + SLEEPDEEP bit + WFI or WFE	WKUP pin rising edge, RTC alarm, external reset in NRST pin, IWDG reset			OFF

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

- Slowing down the system clocks. In Run mode, the speed of system clock (SYSCLK, HCLK, PCLK1, and PCLK2) can be reduced by programming the prescaler registers. Before entering the Sleep mode, the prescalers can also be used to slow down peripheral clocks. Please refer to [Section 3.3.2](#) for more details.
- Gating the clocks of the APB and AHB peripherals when they are unused. In Run mode, peripherals and memories clocks (HCLK and PCLKx) can be stopped at any time to reduce power consumption. To further reduce power consumption in Sleep mode, all the peripheral clocks can be disabled before executing the WFI or WFE instructions. Peripheral clock gating is controlled by configuring AHB peripheral clock (which enables the RCC_AHBEN register), APB2 peripheral clock (which enables the RCC_APB2EN register), and APB1 peripheral clock (which enables the RCC_APB1EN register).

2.3.2.1 Sleep Mode

Entering Sleep mode:

The Sleep mode is entered by executing WFI or WFE instructions. According to the SLEEPONEXIT bit in the Cortex®-M4F system control register, there are two options to select the Sleep mode entry mechanism:

- SLP-NOW: If the SLEEPONEXIT bit is cleared, the MCU enters Sleep mode as soon as WFI or WFE instruction is executed.
- SLP-ON-EXIT: If the SLEEPONEXIT bit is set, the MCU enters Sleep mode as soon as it exits the lowest priority ISR.

In the Sleep mode, all I/O pins keep the same state as in the Run mode. Please refer to [Table 2-1](#) and [Table 2-2](#) for more details on how to enter Sleep mode.

Exiting Sleep mode

If the WFI instruction is executed to enter Sleep mode, any peripheral interrupt acknowledged by the nested vectored interrupt controller (NVIC) can wake up the device from Sleep mode.

If the WFE instruction is 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 an interrupt in the peripheral control register instead of in the NVIC, and enabling the SEVONPEND bit in the Cortex®-M4F system control register. When the MCU resumes from WFE, the peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) need to be cleared.
- Configuring an external or internal EXTI line in event mode. When the MCU resumes from WFE, since the pending bit corresponding to the event line is not set, it is not necessary to clear the peripheral interrupt pending bit or the NVIC IRQ channel pending bit.

The wakeup time required by this mode is the shortest, since there is no time wasted in interrupt

entry/exit. Please refer to [Table 2-1](#) and [Table 2-2](#) for more details on how to exit Sleep mode.

Table 2-1 SLP-NOW Mode

SLP-NOW Mode	Description
Entry	WFI (Wait for Interrupt) or WFE (Wait for Event) while: - SLEEPDEEP = 0 and - SLEEPONEXIT = 0 Refer to the Cortex®-M4F system control register.
Exit	If WFI was executed for entry: Interrupt: Refer to Interrupt and exception vectors (Table 8-1) If WFE was executed for entry: Wakeup event: Refer to Wakeup event management (Section 8.2.3)
Wakeup latency	None

Table 2-2 SLP-ON-EXIT Mode

SLP-ON_EXIT Mode	Description
Entry	Execute WFI while: - SLEEPDEEP = 0 and - SLEEPONEXIT = 1 Refer to the Cortex®-M4F system control register.
Exit	Interrupt: Refer to Interrupt and exception vectors (Table 8-1)
Wakeup latency	None

2.3.2.2 Stop Mode

The Stop mode is based on the Cortex®-M4F Deep-sleep mode and combined with peripheral clock gating. In Stop mode, the voltage regulator can operate normally either in normal or low-power mode. At this time, all clocks in the 1.2 V domain are stopped, the PLL, the HIS, and the HSE RC oscillators are disabled. SRAM and register contents are preserved.

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

Entering Stop mode

Please refer to [Table 2-3](#) for details on how to enter the Stop mode.

If Flash memory programming is ongoing, Stop mode entry is delayed until the memory access is finished. If an access to the APB domain is ongoing, the Stop mode entry is delayed until the APB access is finished. The following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except by reset. Please refer to [Section 11.2.3](#).
- Real-time clock (RTC): This is configured by the RTCEN bit in the backup domain control register (RCC_BDCR)
- Internal RC oscillator (LSI RC): This is configured by the LSION bit in the control/status register (RCC_CTRLSTS).
- External 32.768 kHz oscillator (LSE): This is configured by the LSEON bit in the backup domain control register (RCC_BDC).

In Stop mode, peripherals such as the ADC or DAC still consume power if they are not turned off before entering Stop mode. To disable them, the ADON bit in the ADC_CR2 register and the CHENx bit in the DAC_CTRL register must both be written to 0.

Exiting Stop mode

Please refer to Table 2-3 for more details on how to exit Stop mode.

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

When the voltage regulator operates in low-power mode, an additional startup delay is incurred when waking up from Stop mode. By keeping the internal regulator ON in Stop mode, the consumption

increased as the startup time is reduced.

Table 2-3 Stop Mode

Stop mode	Description
Entry	<p>Execute WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> - Set SLEEPDEEP bit in Cortex®-M4F system control register. - Clear PDDS bit in power control register (PWR_CR). <p>Note: To enter Stop mode, all EXTI Line pending bits (in the Pending register (EXTI_PR)), all peripheral interrupt pending bits, and RTC alarm flag must be reset. Otherwise, the program will skip the entry process of Stop mode to continue to execute the following procedure.</p>
Exit	<p>If WFI was executed for entry: Configure any EXTI Line as Interrupt mode (the corresponding EXTI interrupt vector must be enabled in the NVIC). Please refer to interrupt and exception vectors (Table 8-1).</p> <p>If WFE was used for entry: Configure any EXTI Line as event mode. Please refer to wakeup event management (Section 8.2.3).</p>
Wakeup latency	HSI RC wakeup time

2.3.2.3 Standby Mode

Standby mode can achieve the lowest power consumption for the device. This mode is based on the Cortex®-M4F Deep-sleep mode, with the voltage regulator disabled. The whole 1.2 V domain is power-off. The PLL, the HSI oscillator, and the HSE oscillator are also switched off. SRAM and register contents are lost. Only registers in the backup domain and standby circuitry remain supplied (Please refer to [Figure 2-1](#)).

Entering Standby mode

Please refer to Table 2-4 for more details on how to enter Standby mode. In Standby mode, the following features can be selected by programming individual control bits:

- Independent watchdog (IWDG): IWDG is started by writing to its Key register or by hardware option. Once started, it cannot be stopped except by reset. See [Section 11.2.3](#).
- Real-time clock (RTC): This is configured by the RTCEN bit in the backup domain control register (RCC_BDCR).
- Internal RC oscillator (LSI RC): This is configured by the LSIEN bit in the control/status register (RCC_CTRLSTS).
- External 32.768 kHz oscillator (LSE OSC): This is configured by the LSEEN bit in the backup domain control register (RCC_BDC).

Exiting Standby mode

The microcontroller exits the Standby mode when an external reset (NRST pin), an IWDG reset, a rising edge on the WKUP pin, or the rising edge of an RTC alarm occurs (See [Figure 12-1](#)). All registers are reset after waking up from Standby mode except for the power control/status register (PWR_CTRLSTS) (See [Section 3.3.10](#)). After waking up from Standby mode, the program executes the instruction code in the same way as after reset (boot pins sampling, vector reset is fetched, etc.). The power control/status register (PWR_CTRLSTS) (See [Section 3.3.10](#)) will instruct the MCU to exit from the Standby mode.

Table 2-4 Standby Mode

Standby Mode	Description
Entry	<p>Execute WFI (Wait for Interrupt) or WFE (Wait for Event) while:</p> <ul style="list-style-type: none"> - Set SLEEPDEEP bit in Cortex®-M4F system control register - Set PDDS bit in power control register (PWR_CR) - Clear WUF bit in power control/status register (PWR_CSR)

Exit	Rising edge of the WKUP pin, rising edge of the RTC alarm event, external reset in NRST pin, IWDG reset
Wakeup latency	$T_{RSTTEMPO}$

I/O states in Standby mode

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

- Reset pad (still available)
- The TAMPER pin if configured for tamper or calibration out
- The WKUP pin, if enabled

2.3.2.4 Debug Mode

By default, the debug connection is lost if the MCU is in Stop or Standby mode while debugging. This is because Cortex®-M4F core is no longer clocked. However, by setting some configuration bits in the DBGMCU_CTRL register, the software can be debugged even in the low-power modes. For more details, please refer to [Section 22.2.1](#).

2.3.3 Auto-wakeup (AWU)

The RTC can wake up the MCU from the low-power mode without an external interrupt (Auto-wakeup mode). The RTC provides a programmable time base for waking up from Stop or Standby mode at regular intervals. Two of the three RTC clock sources can be selected to perform this function by programming the RTCSEL[1:0] bits in the backup domain control register (RCC_BDC):

Low-power 32.768 kHz external crystal oscillator (LSE)

This clock source provides a precise time base with very low power consumption (typically less than 1 μ A).

Low-power internal RC oscillator (LSI RC)

When being used, this clock source saves the cost of the 32.768 kHz crystal. However, the power consumed by the RC Oscillator may slightly increases.

To wake up the device from Stop mode with an RTC alarm event, it is necessary to:

Configure the EXTI Line 17 as rising edge trigger

Configure the RTC to generate the RTC alarm

To wake up from Standby mode, there is no need to configure the EXTI Line 17.

2.4 PWR Registers

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

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PWR_CTRL	Reserved																				DBP	PVDS[2:0]			PVDEN	CLSBF	3					
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x04	PWR_CTRL_STS	Reserved																				WUPEN	Reserved			PVD	SBF	0	0	0			
	Reset Value	0																				0	0	0	0	0	0	0	0	0	0		

2.4.1 Power Control Register (PWR_CTRL)

Address offset: 0x000

Reset value: 0x0000 0000 (Reset by waking up from Standby mode.)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16					
Reserved																				
r																				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Reserved				DBP	PVDS[2:0]	PVDEN	CLSBF	CLWUF	PDSS	Reserved										
r				rw	rw	rw	rc_w1	rc_w1	rw	r										
Bit 31:9		Reserved. Always read as 0.																		
Bit 8		DBP: Disable backup domain write protection After reset, the RTC and backup registers are protected against unwanted write access. This bit must be set to enable write access to these registers. 0: Access to RTC and backup registers is disabled. 1: Access to RTC and backup registers is enabled. Note: If HSE/128 is used as the RTC clock, this bit must be set.																		
Bit 7:5		PVDS[2:0]: PVD level selection These bits are used to select the voltage threshold detected by the power voltage detector 000: Reserved 100: 2.6 V 001: 2.3 V 101: 2.7 V 010: 2.4 V 110: 2.8 V 011: 2.5 V 111: 2.9 V Note: Please refer to the electrical characteristics of the data sheet for more details.																		
Bit 4		PVDEN: Power voltage detector (PVD) enable 0: PVD is disabled. 1: PVD is enabled.																		
Bit 3		CLSBF: Clear standby flag This bit is always read as 0. 0: No effect 1: Clear the SBF Standby Flag (Write)																		
Bit 2		CLWUF: Clear wakeup flag This bit is always read as 0. 0: No effect 1: Clear the WUF wakeup flag after 2 system clock cycles. (Write)																		
Bit 1		PDSS: Power down Deep-sleep 0: Enter Stop mode when the CPU enters Deep-sleep. 1: Enter Standby mode when the CPU enters Deep-sleep.																		
Bit 0		Reserved																		

2.4.2 Power Control/Status Register (PWR_CTRLST)

Address offset: 0x004

Reset value: 0x0000 0000 (Not reset by waking up from Standby mode.)

Additional APB cycles are needed to read this register versus a standard APB read.

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

Bit 31:9	Reserved. Always read as 0.
Bit 8	<p>WUPEN: Enable WKUPF pin 0: WKUPF pin is used for general purpose I/O. An event on the WKUPF pin does not wake up the CPU from Standby mode. 1: WKUP pin is used for waking up the CPU from Standby mode, and is forced in input pull-down configuration. (Rising edge on the WKUPF pin wakes up the system from Standby mode.) Note: This bit is reset by a system reset.</p>
Bit 7:3	Reserved. Always read as 0.
Bit 2	<p>PVD: PVD output This bit is valid only if PVD is enabled by the PVDEN bit. 0: VDD/VDDA is higher than the PVD threshold selected by the PLS[2:0] bits. 1: VDD/VDDA is lower than the PVD threshold selected by the PLS[2:0] bits. Note: The PVD is stopped in Standby mode. Hence, this bit remains 0 after Standby or reset until the PVDEN bit is set.</p>
Bit 1	<p>SBF: Standby flag This bit is set by hardware and can only be cleared by a POR/PDR (Power-on reset/power-down reset bit) or by setting the CSBF bit in the power control register (PWR_CTRL). 0: Device is not in Standby mode. 1: Device is in Standby mode.</p>
Bit 0	<p>WUF: Wakeup flag This bit is set by hardware and can only be cleared by a POR/PDR (Power-on reset/power-down reset bit) or by setting the CLWUF bit in the power control register (PWR_CTRL). 0: No wakeup event occurred. 1: A wakeup event was received from the WKUPF pin or from the RTC alarm. Note: An additional wakeup event is detected if the WKUPF pin is enabled (by setting the WUPEN bit) when the WKUPF pin level is already high.</p>

3 Reset and Clock Control (RCC)

3.1 Reset

AT32F403 provides three types of reset, including system reset, power-on reset, and backup domain reset.

3.1.1 System Reset

Except for the reset flags in the RCC_CTRLSTS register of clock controller and the registers in the backup domain, a system reset sets all registers to their reset values (See [Figure 2-1](#)).

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

1. A low level on the NRST pin (external reset)
2. Window watchdog end of count condition (WWDG reset)
3. Independent watchdog end of count condition (IWDG reset)
4. A software reset (SW reset)
5. Low-power management reset

The reset source can be identified by checking the reset flags in the control/status register (RCC_CTRLSTS).

Software reset:

By setting the SYSRESETREQ bit in Cortex®-M4F Application Interrupt and Reset Control Register, software reset can be launched. Please refer to the Cortex®-M4 programming manual for further information.

Low-power management reset:

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

1. Reset generated when entering Standby mode: This type of reset is enabled by setting the nSTDBY_RST bit in User Option Bytes. In this case, whenever a Standby mode entry sequence is successfully executed, the device is reset instead of entering Standby mode.
2. Reset generated when entering Stop mode: This type of reset is enabled by setting the nSTP_RST bit in User Option Bytes. In this case, whenever a Stop mode entry sequence is successfully executed, the device is reset instead of entering Stop mode.

For further information on the User Option Bytes, please refer to the AT32F403 Flash memory programming manual.

3.1.2 Power Reset

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

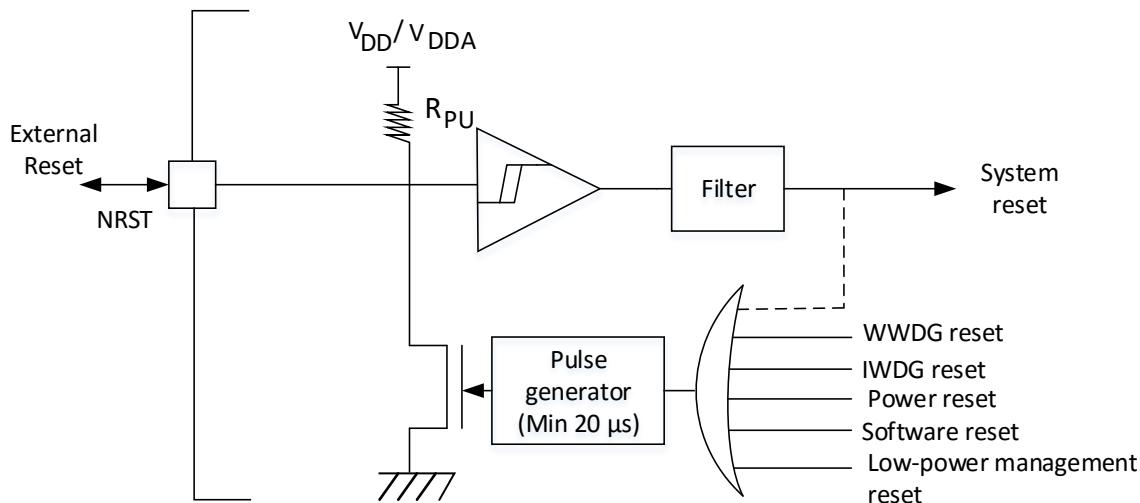
1. Power-on/Power-down reset (POR/PDR reset)
2. When exiting Standby mode

A power reset sets all registers to their reset values except the backup domain (see [Figure 2-1](#)).

The reset sources act on the RESET pin, and it is always kept low at the reset phase. The RESET service routine vector is fixed at address 0x0000_0004 on the memory map. For more details, please refer to [Table 8-1](#).

The system reset signal provided to the device is outputted on the NRST pin. The pulse generator guarantees a minimum reset pulse duration of 20 µs for each reset source (external or internal reset). In cases of an external reset, the reset pulse is generated while the NRST pin is asserted low.

Figure 3-1 Reset Circuit



3.1.3 Backup Domain Reset

The backup domain has two specific resets that affect only the backup domain (See [Figure 2-1](#)).

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

1. Software reset, triggered by setting the BDRST bit in the backup domain control register (RCC_BDC) (See [Section 3.3.9](#)).
2. VDD or VBAT power on, if both supplies have previously been powered off.

3.2 Clocks

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

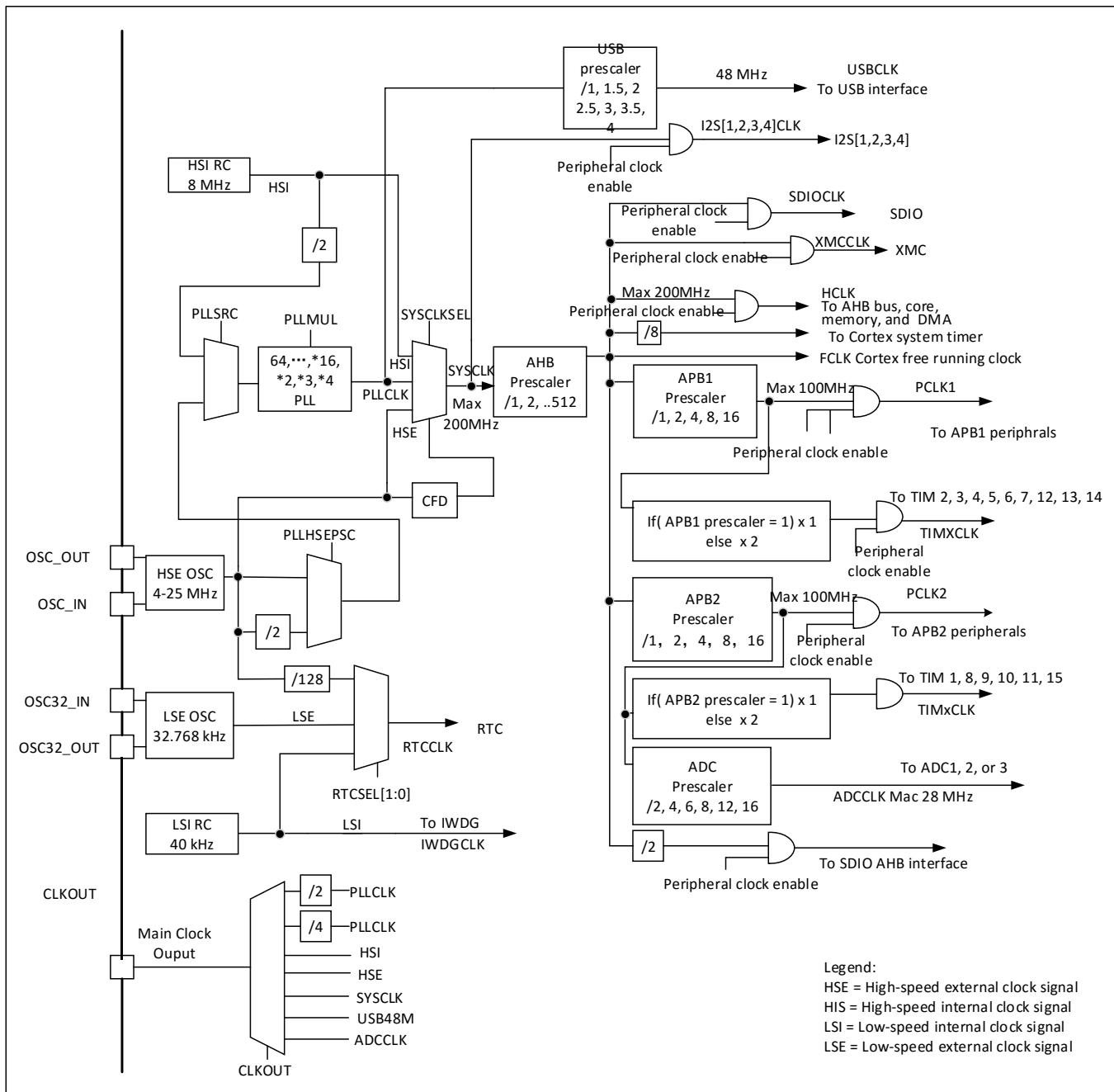
- HSI oscillator clock
- HSE oscillator clock
- PLL clock

The devices have the following two secondary clock sources:

- 40 kHz low-speed internal RC which drives the independent watchdog, and the RTC is used for auto-wakeup from Stop/Standy mode.
- 32.768 kHz low-speed external crystal (LSE crystal) which drives the Real Time Clock (RTCCLK).

Each clock source can be switched on or off independently when it is not used, to optimize power consumption.

Figure 3-2 Clock Tree



- When the HSI is used as a PLL clock input, the maximum system clock frequency that can be achieved is 200 MHz.
- For the internal and external clock source characteristics, please refer to the “Electrical characteristics” section in the corresponding device data sheet.

Several prescalers allow the configuration of the AHB frequency, the high-speed APB (APB2) and the low-speed APB (APB1) domain frequency. The maximum frequency of the AHB domain is 200 MHz. The maximum frequency of the APB1 and the APB2 domains is 100 MHz. The SDIO interface is clocked with a fixed frequency of HCLK/2.

The RCC feeds the Cortex® System Timer (SysTick) external clock with the AHB clock (HCLK) divided by 8. The SysTick can work either with this clock or with the Cortex® clock (HCLK), configurable in the SysTick Control and Status Register. The ADCs are clocked by the clock of the high-speed domain APB2 divided by 2, 4, 6, 8, 12, or 16.

The timer clock frequencies are automatically fixed by hardware depending on the two conditions:

- If the APB prescaler is 1, the timer clock frequencies are set to the same frequency as that of the APB domain to which the timers are connected.

2. Otherwise, they are set to double ($\times 2$) the frequency of the APB domain to which the timers are connected.

FCLK acts as Cortex®-M4F's free-running clock. For more details, please refer to ARM Cortex®-M4 Technical Reference Manual.

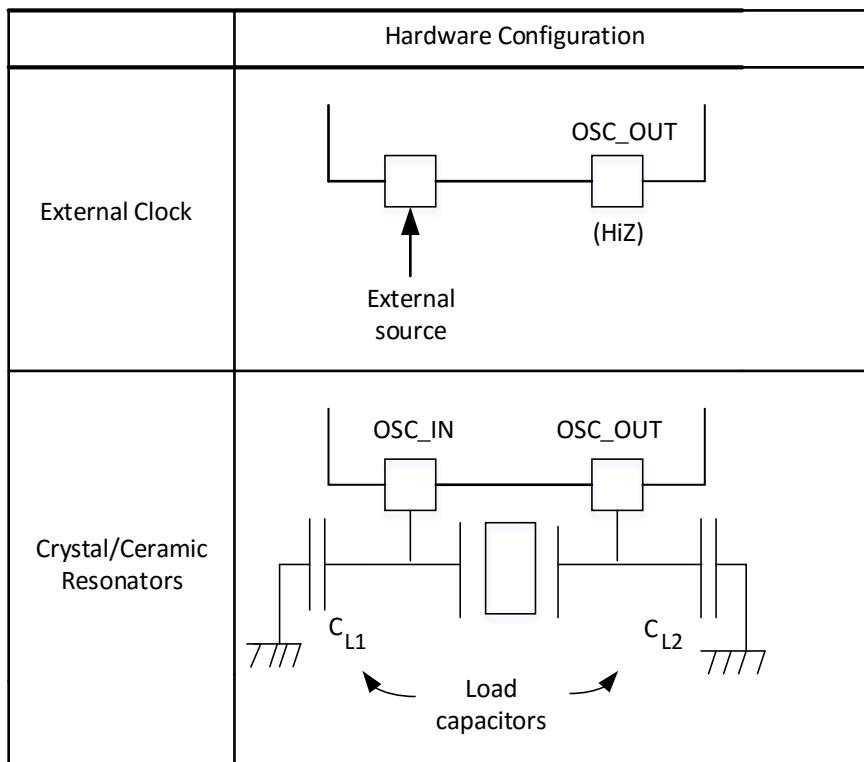
3.2.1 HSE Clock

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

- HSE external crystal/ceramic resonator
- HSE user external clock

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

Figure 3-3 HSE/LSE Clock Sources



External source (HSE bypass)

In this mode, an external clock source must be provided. Its frequency can be up to 25 MHz. Users can select this mode by setting the HSEBYP and HSEEN bits in the clock control register. The external clock signal (square, sinus, or triangle with ~50% duty cycle) should be connected to the OSC_IN pin while the OSC_OUT pin should be left HiZ. See [Figure 3-3](#).

External crystal/ceramic resonator (HSE crystal)

The 4 ~ 16 MHz external oscillator produces a highly accurate clock source as the system clock. Relevant hardware configuration is shown in [Figure 3-3](#). Please refer to the electrical characteristics section of the data sheet for more details.

The HSESTBL bit in the clock control register (RCC_CTRL) indicates if the high-speed external oscillator is stable. When powered up, the clock is not released until this bit is set by hardware. An interrupt can be generated if enabled in the clock interrupt register (RCC_CLKINT).

The HSE crystal can be switched on and off by configuring the HSEEN bit in the clock control register (RCC_CTRL).

3.2.2 HIS Clock

The HSI clock signal is generated from an internal 8 MHz RC oscillator and can be used directly as a system clock or divided by 2 to be used as PLL input.

The HSI RC oscillator can provide the system a clock source without any external components. Its startup time is shorter than the HSE crystal oscillator; however, even with calibration, its clock frequency is still less accurate.

Calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, and this is why each device is calibrated by ATK for 1% accuracy (25°C) in factory. After reset, the factory calibration value is loaded in the HSICAL[7:0] bits of the clock control register. HSICAL[7:0] can be reconfigured by software (HSICAL_KEY needs to be set as 0x5A). Once configured by software, a system reset is necessary to reload factory calibration value.

The RC oscillator speed may be affected by voltage or temperature variations. HSI frequency can be trimmed by using the HSITWK[4:0] bits in the clock control register.

The HSISTBL bit in the clock control register indicates if the HSI RC is stable. When the clock is powered up, the HSI RC output clock is not released until this bit is set by hardware.

The HSI RC can be switched on and off by using the HSIEN bit in the clock control register. If the HSE crystal oscillator fails, the HSI clock can be used as a backup source. Please refer to [Section 3.2.7](#).

3.2.3 PLL

The internal PLL can multiply HSI RC output clock or HSE crystal output clock. Please refer to [Figure 3-2](#) and clock control register.

The configuration of PLL (selecting the clock source of HSI oscillator divided by 2 or HSE oscillator and selecting multiplication factor) must be done before enabling the PLL. Once the PLL is enabled, these parameters cannot be changed.

If PLL interrupt is enabled in the clock interrupt register, an interrupt can be generated when the PLL is ready. If USB interface is used in the application, PLL should be set as output 48, 72, 96, 120, 144, 168, or 192 MHz clock and used to provide 48 MHz USBCLK clock.

3.2.4 LSE Clock

The LSE crystal is a 32.768 kHz low-speed External crystal or ceramic resonator. It provides highly accurate clock source to the real-time clock or other timing functions with low power consumption.

The LSE crystal is switched on and off by using the LSEEN bit in backup domain control register (RCC_BDC). The LSESTBL bit in backup domain control register (RCC_BDC) indicates if the LSE crystal is stable. When powered up, the LSE crystal output clock signal is not released until this bit is set by hardware. An interrupt can be generated if enabled in the Clock interrupt register.

External source (LSE bypass)

In this mode, an external clock source with the frequency of 32.768 kHz must be provided. Users can select this mode by setting the LSEBYP and LSEON bits in the backup domain control register (RCC_BDC). The external clock signal (square, sinus, or triangle with ~50% duty cycle) has to connect to the OSC32_IN pin while the OSC32_OUT pin should be left HiZ. See [Figure 3-3](#).

3.2.5 LSI Clock

The LSI RC acts as a low-power clock source that can be kept running in Stop and Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU). The clock frequency is around 40 kHz (between 30 kHz and 60 kHz). For more details, please refer to the electrical characteristics section of the data sheet.

The LSI RC can be switched on and off by using the LSIEN bit in the control/status register (RCC_CTRLSTS). The LSISTBL bit in the control/status register (RCC_CTRLSTS) indicates if the low-speed internal oscillator is stable. When powered up, the clock is not released until this bit is set by hardware. An LSI interrupt can be generated if enabled in the clock interrupt register (RCC_CLKINT).

LSI calibration

The frequency dispersion of the low-speed internal (LSI) oscillator can be calibrated to obtain RTC time base and IWDG timeout (when LSI is used as clock source for these peripherals) with an acceptable accuracy.

This calibration can be done by measuring the LSI clock frequency with TIM5 input clock (TIM5_CLK). According to this measurement done at the precision of the HSE oscillator, the software can adjust the

programmable 20-bit prescaler of the RTC to obtain an accurate RTC clock time base and compute accurate IWDG timeout.

LSI calibration procedures are as follows:

1. Enable TMR5 and configure channel 4 in input capture mode.
2. Set the TMR5_CH4_IREMAP bit in the AFIO_MAP to internally connect the LSI to channel 4 of TMR5.
3. Measure the frequency of LSI clock with the TIMR5 capture/compare 4 event or interrupt.
4. Configure the 20-bit prescaler according to the measured and desired RTC time base and IWDG timeout.

3.2.6 System Clock (SYSCLK) Selection

After a system reset, the HSI oscillator is selected as system clock. When a clock source is used directly or indirectly through the PLL as the system clock, it cannot be stopped.

A switch from one clock source to another occurs only if the target clock source is ready (the clock becomes stable after startup delay or PLL becomes stable). If the selected clock source is not ready, the switch will not occur until the target clock source is ready.

Status bits in the clock control register (RCC_CTRL) indicate which clock is ready and which clock is currently used as system clock.

3.2.7 Clock Failure Detection (CFD)

Clock Failure Detection can be activated by software. Once activated, the clock detector is enabled after the HSE oscillator startup delay, and disabled when this oscillator is stopped.

If a failure is detected on the HSE clock, the HSE oscillator is automatically disabled; a clock failure event is sent to the break input of the advanced control timer (TMR1 and TMR8), and a clock failure detection interrupt (CFDI) is generated to allow the device to perform rescue operation. The CFDI is linked to the Cortex®-M4F NMI (Non-Maskable Interrupt).

Note: Once the CFD is enabled, the CFD interrupt occurs if the HSE clock fails, and an NMI is automatically generated. The NMI will be executed indefinitely unless the CFD interrupt pending bit is cleared. As a consequence, in the NMI ISR, users must clear the CFD interrupt by setting the CFDFC bit in the clock interrupt register (RCC_CLKINT).

If the HSE oscillator is used directly or indirectly as the system clock (“indirectly” means that it is used as PLL input clock, and the PLL clock is used as system clock), a detected failure will cause a switch of the system clock to the HSI oscillator and disabling of the external HSE oscillator at the same time. When the failure occurs, if the HSE oscillator clock (divided or not) is the clock entry of the PLL used as system clock, the PLL will also be disabled.

3.2.8 RTC Clock

By programming the RTCSEL[1:0] bits in the backup domain control register (RCC_BDC), the RTCCLK clock source can be provided by HSE/128, LSE, or LSI clocks. This selection cannot be changed without resetting the backup domain.

The LSE clock is in the backup domain, whereas the HSE and LSI clocks are not. Consequently:

- If LSE is selected as RTC clock:
 - The RTC continues to work as long as VBAT supply is maintained.
- If LSI is selected as Auto-Wakeup Unit (AWU) clock:
 - The AWU state is not guaranteed if the VDD supply is powered off. Please refer to [Section 3.2.5](#) for more details on LSI calibration.
- If the HSE clock divided by 128 is used as RTC clock:
 - The RTC state is not guaranteed if the VDD supply is powered off or if the internal voltage regulator is powered off (1.2 V domain power supply is cut off).
 - The DPB bit (disable backup domain write protection) in the power controller register ([See Section 2.4.1](#)) must be set.

3.2.9 Watchdog Clock

If the Independent watchdog (IWDG) is started by either hardware option or software access, the LSI oscillator is forced ON and cannot be disabled. After the LSI oscillator temporization, the clock is provided to the IWDG.

3.2.10 Clock-out Capability

The microcontroller allows the clock to be output onto the external CLKOUT pin. The corresponding GPIO port register must be programmed as the corresponding function mode. The following seven clock signals can be selected as CLKOUT clock:

- ADC CLK
 - USB48M
 - SYSCLK
 - HSI
 - HSE
 - PLL clock divided by 2. The selection is controlled by the CLKOUT[3:0] bits of the clock configuration register (RCC_CFG).
 - PLL clock divided by 4. The selection is controlled by the CLKOUT[3:0] bits of the clock configuration register (RCC_CFG).

3.3 RCC Registers Description

Table 3-1 lists RCC register map reset values.

Table 3-1 RCC Register Map and Reset Values

014h	RCC_AH BEN Reset Value	Reserved																							
018h	RCC_APB 2EN Reset Value	Reserved																							
01Ch	RCC_APB 1EN Reset Value	Reserved		DACEN 0	PWREN 0	BKPN 0	I2C3CHEN 0	CANEN 0	USBEN 0	I2C2CHEN 0	I2C1CHEN 0	UART5CHEN 0	UART4CHEN 0	USART3CHEN 0	USART2CHEN 0	SP4CHEN 0	TMR15EN 0	TMR11EN 0	TMR10EN 0						
020h	RCC_BD C Reset Value	Reserved																							
024h	RCC_CTRLSTS Reset Value	LPRSTF 0	WWDG_RSTF 0	WDG_RSTF 0	SWRSTF 0	PORSTF 1	FINRSTF 1	Reserved 0	RSTFC 0	Reserved															
030h	RCC_MIS C Reset Value	Reserved						Reserved						CLKOUT[3] 0	Reserved										
		HSICAL_KEY[7:0]																							
		0 0 0 0 0 0 0 0 0																							

3.3.1 Clock Control Register (RCC_CTRL)

Address offset: 0x00

Reset value: 0x000 XX83, where X is undefined

Access: No wait state. Word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Reserved						PLL STBL	PLLEN	Reserved		CFDEN	HSE BYPS	HSE STBL	HSE EN						
res				r		rw		res		rw	rw	rw	r	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
HSICAL[7:0]								HSITWK[7:3]				Reserved	HSI STBL	HSI EN					
rw				rw				rw				rw	r	rw					
Bit 30:26		Reserved. Always read as 0.																	

Bit 25	PLLSTBL: PLL clock ready flag Set by hardware after PLL is locked. 0: PLL is unlocked. 1: PLL is locked.
Bit 24	PLLEN: PLL enable Set and cleared by software. Cleared by hardware when entering Stop or Standby mode. When PLL clock is used as or selected to become system clock, this bit cannot be cleared. 0: PLL is OFF. 1: PLL is ON.
Bit 23:20	Reserved. Always read as 0.
Bit 19	CFDEN: Clock Failure Detection enable Set and cleared by software to enable Clock Failure Detection. 0: Clock Failure Detection is OFF. 1: If the external 4 ~ 16 MHz oscillator is ready, Clock Failure Detection is ON.
Bit 18	HSEBYP_S: External high-speed clock bypass Set and cleared by software to bypass the external crystal oscillator. This bit can be written only if the HSE oscillator is disabled. 0: External 4 ~ 16 MHz crystal oscillator is not bypassed. 1: External 4 ~ 16 MHz crystal oscillator is bypassed.
Bit 17	HSESTBL: External high-speed clock ready flag Set by hardware to indicate that the external 4 ~ 16 MHz oscillator is stable. This bit needs 6 cycles of external 4~16 MHz oscillator to be cleared after HSEEN bit is reset. 0: External 4 ~ 16 MHz oscillator is not ready. 1: External 4 ~ 16 MHz oscillator is ready.
Bit 16	HSEEN: External high-speed clock enable Set and cleared by software. When entering Stop or Standby mode, this bit is cleared by hardware, and the 4 ~ 16 MHz oscillator is disabled. When external 4 ~ 16 MHz oscillator is used as or selected to become system clock, this bit cannot be cleared. 0: HSE oscillator is OFF. 1: HSE oscillator is ON.
Bit 15:8	HSICAL[7:0]: Internal high-speed clock calibration These bits are initialized automatically when the system is powered up. This byte can be written only when HSICAL_KEY[7:0] is 0x5A.
Bit 7:3	HSITWK[4: 0]: Internal high-speed clock trimming These bits are written by software to adjust internal high-speed clock, added to the HSICAL[5:0] values. These bits allow users to key in a trimming value that can adjust the frequency of the internal HSI RC oscillator according to voltage and temperature variations. The default value is 16, which can trim the HSI to 8 MHz \pm 1%. The trimming step is around 40 kHz between two consecutive HSICAL steps.
Bit 2	Reserved. Always read as 0.
Bit 1	HSISTBL: Internal high-speed clock ready flag Set by hardware to indicate that the internal 8 MHz oscillators is stable. After the HSIEN bit is cleared, this bit is cleared after 6 internal 8 MHz oscillator cycles. 0: Internal 8 MHz oscillator is not ready. 1: Internal 8 MHz oscillator is ready.
Bit 0	HSIEN: Internal high-speed clock enable Set and cleared by software. When leaving Stop or Standby mode or in case of the external 4 ~ 16 oscillator failure, this bit is set by hardware to enable the internal 8 MHz RC oscillator. This bit cannot be cleared if the internal 8 MHz oscillator is used directly or indirectly as system clock or is selected to become the system clock. 0: Internal 8 MHz oscillator is OFF. 1: Internal 8 MHz oscillator is ON.

3.3.2 Clock Configuration Register (RCC_CFG)

Address offset: 0x04

Reset value: 0x0000 0000

Access: 0 to 2 wait state. Word, half-word, and byte access. 1 or 2 wait states are inserted only when the access occurs during a clock source switch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PLL RANGE	PLLMUL [5:4]	ADC PSC[2]	USB PSC[2]	CLKOUT[2:0]	USBPSC [1:0]		PLLMUL[3:0]		PLLHSE PSRC	PLL SRC					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPSC[1:0]		APB2PSC[2:0]		APB1PSC[2:0]		AHBPSC[3:0]		SYSCLKSTS [1:0]		SYSCLKSEL [1:0]					
	rw		rw		rw		rw		r		rw				

Bit 31	PLL RANGE : PLL clock output range 0: PLL output \leq 72 MHz 1: PLL output > 72 MHz
Bit 26:24	CLKOUT[3:0] : Microcontroller clock output CLKOUT[3] is in bit 16 of the RCC_MISC register. Set and cleared by software. 00xx: No clock output 0100: System clock (SYSCLK) output 0101: Internal RC oscillator clock (HSI) output 0110: External RC oscillator clock (HSE) output 0111: Output after PLL clock divided by 2 1100: Output after PLL clock divided by 4 1101: USB clock output 1110: ADC clock output Note: This clock output may be disconnected when powered up or during CLKOUT clock source switching. -When the system clock is used as output to the CLKOUT pin, make sure that the frequency of output clock must not exceed 50 MHz (the maximum I/O frequency).
Bit 27 Bit 23:22	USBPSC[2:0] : USB prescaler Set and cleared by software to generate 48 MHz USB clock. This bit must be valid before enabling the USB clock in the RCC_APB1EN register. This bit cannot be cleared if USB clock is enabled. 000: PLL clock divided by 1.5 to be USB clock 001: PLL clock directly to be USB clock 010: PLL clock divided by 2.5 to be USB clock 011: PLL clock divided by 2 to be USB clock 100: PLL clock divided by 3.5 to be USB clock 101: PLL clock divided by 3 to be USB clock 110: PLL clock divided by 4 to be USB clock 111: PLL clock divided by 4 to be USB clock
Bit 30:29 Bit 21:18	PLLMUL[5:0] : PLL multiplication factor { Bit 30:29, Bit 21:18} These bits are written by software to define the PLL multiplication factor. They can be written only when PLL is disabled. Note: PLLRANGE register must be configured with PLL output. 000000: PLL output x 2 001111: PLL output x 16 010000: PLL output x 17 011111: PLL output x 32 100000: PLL output x 33 101111: PLL output x 48 110000: PLL output x 49 111111: PLL output x 64
Bit 17	PLLHSEPSRC : HSE divider for PLL entry Set and cleared by software to divide HSE as PLL input clock. This bit can be written only when PLL is disabled. 0: HSE is not divided. 1: HSE is divided by 2.

Bit 16	PLLCSR : PLL entry clock source Set and cleared by software to select PLL input clock source. This bit can be written only when PLL is disabled. 0: HSI oscillator clock divided by 2 to be PLL input clock. 1: HSE clock is used as PLL input clock
Bit 28 Bit 15:14	ADCPSC[2:0] : ADC prescaler ADCPSC[2] , Set by bit 28 Set and cleared by software to define ADC clock frequency. 000: PCLK2 divided by 2 to be ADC clock 001: PCLK2 divided by 4 to be ADC clock 010: PCLK2 divided by 6 to be ADC clock 011: PCLK2 divided by 8 to be ADC clock 100: PCLK2 divided by 2 to be ADC clock 101: PCLK2 divided by 12 to be ADC clock 110: PCLK2 divided by 8 to be ADC clock 111: PCLK2 divided by 16 to be ADC clock
Bit 13:11	APB2PSC[2:0] : APB high-speed prescaler (APB2) Set and cleared by software to control the division factor of high-speed APB2 clock (PCLK2). Caution: These bits must be configured by software to ensure that the frequency of APB2 clock is less than 100 MHz. 0xx: HCLK is not divided. 100: HCLK is divided by 2. 101: HCLK is divided by 4. 110: HCLK is divided by 8. 111: HCLK is divided by 16.
Bit 10:8	APB1PSC[2:0] : APB low-speed prescaler (APB1) Set and cleared by software to control the division factor of low-speed APB1 clock (PCLK1). Caution: These bits must be configured by software to ensure that the frequency of APB1 clock is less than 100 MHz. 0xx: HCLK is not divided. 100: HCLK is divided by 2. 101: HCLK is divided by 4. 110: HCLK is divided by 8. 111: HCLK is divided by 16.
Bit 7:4	AHBPS[3:0] : AHB prescaler Set and cleared by software to control the division factor of AHB clock 0xxx: SYSCLK is not divided. 1000: SYSCLK is divided by 2 1100: SYSCLK is divided by 64 1001: SYSCLK is divided by 4 1101: SYSCLK is divided by 128 1010: SYSCLK is divided by 8 1110: SYSCLK is divided by 256 1011: SYSCLK is divided by 16 1111: SYSCLK is divided by 512 Note: The prefetch buffer must be kept ON when the division factor of AHB clock is higher than 1.
Bit 3:2	SYCLKSTS[1:0] : System clock switch status Set and cleared by hardware to indicate which clock source is used as system clock. 00: HSI used as system clock 01: HSE used as system clock 10: PLL output used as system clock 11: Not allowed
Bit 1:0	SYCLKSEL[1:0] : System clock switch Set and cleared by software to select system clock source. HSI as system clock will be forced by hardware (if the clock security system is enabled) when leaving Stop and Standby mode or failure of HSE used directly or indirectly as system clock is detected. 00: HSI used as system clock 01: HSE used as system clock 10: PLL output used as system clock 11: Not allowed

3.3.3 Clock Interrupt Register (RCC_CLKINT)

Address offset: 0x08

Reset value: 0x0000 0000

Access: No wait state. Word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved				CFDFC	Reserved	PLLSTB LFC	HSEST BLFC	HSIST BLFC	LSES TBLFC	LSIST BLFC					
res				w	res	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PLLS TBLIE	HSES TBLIE	HSIS TBLIE	LSES TBLIE	LSIST BLIE	CFDF	Reserved	PLLS TBLF	HSES TBLF	HSIS TBLF	LSES TBLF	LSIST TBLF			
res	rw	rw	rw	rw	rw	r	res	r	r	r	r	r	r	r	r
Bit 31:24		Reserved. Always read as 0.													
Bit 23		CFDFC: Clock failure detection interrupt clear Write 1 by software to clear the CFDF interrupt flag in the CFDF security system 0: No effect 1: Clear the CFDF security system interrupt flag													
Bit 22:21		Reserved. Always read as 0.													
Bit 20		PLLSTBLFC: PLL ready interrupt clear Write 1 by software to clear the PLL ready interrupt flag PLLSTBLF 0: No effect 1: Clear the PLL ready interrupt flag PLLSTBLF													
Bit 19		HSESTBLFC: HSE ready interrupt clear Write 1 by software to clear the HSE ready interrupt flag HSESTBLF 0: No effect 1: Clear the HSE ready interrupt flag HSESTBLF													
Bit 18		HSISTBLFC: HSI ready interrupt clear Write 1 by software to clear the HSI ready interrupt flag HSISTBLF 0: No effect 1: Clear the HSI ready interrupt flag HSISTBLF													
Bit 17		LSESTBLFC: LSE ready interrupt clear Write 1 by software to clear the LSE ready interrupt flag LSESTBLF 0: No effect 1: Clear the LSE ready interrupt flag LSESTBLF													
Bit 16		LSISTBLFC: LSI ready interrupt clear Write 1 by software to clear the LSI ready interrupt flag LSISTBLF 0: No effect 1: Clear the LSI ready interrupt flag LSISTBLF													
Bit 15:13		Reserved. Always read as 0.													
Bit 12		PLLSTBLIE: PLL ready interrupt enable Set and cleared by software to enable/disable the PLL ready interrupt 0: PLL ready interrupt is disabled. 1: PLL ready interrupt is enabled.													
Bit 11		HSESTBLIE: HSE ready interrupt enable Set and cleared by software to enable/disable external 4~16 MHz oscillator ready interrupt 0: HSE ready interrupt is disabled. 1: HSE ready interrupt is enabled.													
Bit 10		HSISTBLIE: HSI ready interrupt enable Set and cleared by software to enable/disable internal 8 MHz RC oscillator ready interrupt 0: HSI ready interrupt is disabled. 1: HSI ready interrupt is enabled.													

Bit 9	LSESTBLIE: LSE ready interrupt enable Set and cleared by software to enable/disable external 32 kHz RC oscillator ready interrupt 0: LSE ready interrupt is disabled. 1: LSE ready interrupt is enabled.
Bit 8	LSISTBLIE: LSI ready interrupt enable Set and cleared by software to enable/disable internal 40 kHz RC oscillator ready interrupt 0: LSI ready interrupt is disabled. 1: LSI ready interrupt is enabled.
Bit 7	CFDF: Clock Failure Detection interrupt flag Set by hardware when a failure is detected in the external 4~16 MHz oscillator clock. It is cleared by setting the CFDFC bit. 0: No clock security system interrupt is caused by HSE clock failure. 1: Clock security system interrupt is caused by HSE clock failure.
Bit 6:5	Reserved. Always read as 0.
Bit 4	PLLSTBLF: PLL ready interrupt flag Set by hardware when the PLL is stable and the LLSTBLIE bit is set. It is cleared by setting the PLLSTBLFC bit. 0: No clock ready interrupt is caused by PLL lock. 1: Clock ready interrupt is caused by PLL lock.
Bit 3	HSESTBLF: HSE ready interrupt flag Set by hardware when external low-speed clock is stable and the HSESTBLIE bit is set. It is cleared by setting the HSESTBLFC bit. 0: No clock ready interrupt is caused by the external 4~16 MHz oscillator. 1: Clock ready interrupt is caused by the external 4~16 MHz oscillator.
Bit 2	HSISTBLF: HSI ready interrupt flag Set by hardware when internal high-speed clock is stable and the HSISTBLIE bit is set. It is cleared by setting the HSISTBLFC bit. 0: No clock ready interrupt is caused by the internal 8 MHz oscillator. 1: Clock ready interrupt is caused by the internal 8 MHz oscillator.
Bit 1	LSESTBLF: LSE ready interrupt flag Set by hardware when external low-speed clock is stable and the LSESTBLIE bit is set. It is cleared by setting the LSESTBLFC bit. 0: No clock ready interrupt is caused by the external 32 kHz oscillator. 1: Clock ready interrupt is caused by the external 32 kHz oscillator.
Bit 0	LSISTBLF: LSI ready interrupt flag Set by hardware when internal low-speed clock is stable and the LSISTBLIE bit is set. It is cleared by setting the LSISTBLFC bit. 0: No clock ready interrupt is caused by the internal 40 kHz oscillator. 1: Clock ready interrupt is caused by the internal 40 kHz oscillator.

3.3.4 APB2 Peripheral Reset Register (RCC_APB2RST)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: No wait state. Word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						TMR11 RST	TMR10 RST	TMR9 RST	Reserved		TMR15 RST				
res						rw	rw	rw	res		rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 RST	USRAR T1RST	TMR8 RST	SPI1 RST	TMR1 RST	ADC2 RST	ADC1 RST	GPIOG RST	GPIOF RST	GPIOE RST	GPIOD RST	GPIOC RST	GPIOB RST	GPIOA RST	Reserved	AFIO RST
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw
Bit 31:22		Reserved. Always set to 0.													

Bit 21	TMR11RST: TMR11 timer reset Set and cleared by software. 0: No effect 1: Reset TMR11 timer
Bit 20	TMR10RST: TMR10 timer reset Set and cleared by software. 0: No effect 1: Reset TMR10 timer
Bit 19	TMR9RST: TMR9 timer reset Set and cleared by software. 0: No effect 1: Reset TMR9 timer
Bit 18:17	Reserved. Always set to 0.
Bit 16	TMR15RST: TMR15 timer reset Set and cleared by software. 0: No effect 1: Reset TMR15 timer
Bit 15	ADC3RST: ADC3 interface reset Set and cleared by software. 0: No effect 1: Reset ADC3 interface
Bit 14	USART1RST: USART1 reset Set and cleared by software. 0: No effect 1: Reset USART1
Bit 13	TMR8RST: TMR 8 timer reset Set and cleared by software. 0: No effect 1: Reset TMR 8 timer
Bit 12	SPI1RST: SPI1 reset Set and cleared by software. 0: No effect 1: Reset SPI1
Bit 11	TMR1RST: TMR1 timer reset Set and cleared by software. 0: No effect 1: Reset TMR 1 timer
Bit 10	ADC2RST: ADC2 interface reset Set and cleared by software. 0: No effect 1: Reset ADC2 interface
Bit 9	ADC1RST: ADC1 interface reset Set and cleared by software. 0: No effect 1: Reset ADC1 interface
Bit 8	GPIOGRST: IO port G reset Set and cleared by software. 0: No effect 1: Reset IO port G
Bit 7	GPIOFRST: IO port F reset Set and cleared by software. 0: No effect 1: Reset IO port F

Bit 6	GPIOERST : IO port E reset Set and cleared by software. 0: No effect 1: Reset IO port E
Bit 5	GPIODRST : IO port D reset Set and cleared by software. 0: No effect 1: Reset IO port D
Bit 4	GPIOCRST : IO port C reset Set and cleared by software. 0: No effect 1: Reset IO port C
Bit 3	GPIOB_RST : IO port B reset Set and cleared by software. 0: No effect 1: Reset IO port B
Bit 2	GPIOARST : IO port A reset Set and cleared by software. 0: No effect 1: Reset IO port A
Bit 1	Reserved. Always set to 0.
Bit 0	AFIORST : Alternate function I/O reset Set and cleared by software. 0: No effect 1: Reset alternate function

3.3.5 APB1 Peripheral Reset Register (RCC_APB1RST)

Address offset: 0x10

Reset value: 0x0000 0000

Access: No wait state. Word, half-word, and byte access.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DAC RST	PWR RST	BKPR ST	I2C3 RST	CAN RST	Reserved	USB RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	SPI4 RST	
res	res	rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	Reserved	WWWDG RST	Reserved	TMR14 RST	TMR13 RST	TMR12 RST	TMR7 RST	TMR6 RST	TMR5 RST	TMR4 RST	TMR3 RST	TMR2 RST		
rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:30	Reserved. Always read as 0.
Bit 29	DACRST : DAC interface reset Set and cleared by software. 0: No effect 1: Reset DAC interface
Bit 28	PWRRST : Power interface reset Set and cleared by software. 0: No effect 1: Reset power interface
Bit 27	BKPRST : Backup interface reset Set and cleared by software. 0: No effect 1: Reset backup interface

Bit 26	I2C3RST: I2C 3 reset Set and cleared by software. 0: No effect 1: Reset I2C 3
Bit 25	CANRST: CCAN reset Set and cleared by software. 0: No effect 1: Reset CAN
Bit 24	Reserved. Always read as 0.
Bit 23	USBRST: USB reset Set and cleared by software. 0: No effect 1: Reset USB
Bit 22	I2C2RST: I2C 2 reset Set and cleared by software. 0: No effect 1: Reset I2C 2
Bit 21	I2C1RST: I2C 1 reset Set and cleared by software. 0: No effect 1: Reset I2C 1
Bit 20	UART5RST: UART 5 reset Set and cleared by software. 0: No effect 1: Reset UART5
Bit 19	UART4RST: UART 4R reset Set and cleared by software. 0: No effect 1: Reset UART4
Bit 18	USART3RST: USART 3 reset Set and cleared by software. 0: No effect 1: Reset USART3
Bit 17	USART2RST: USART 2 reset Set and cleared by software. 0: No effect 1: Reset USART2
Bit 16	SPI4RST: SPI 4 reset Set and cleared by software. 0: No effect 1: Reset SPI4
Bit 15	SPI3RST: SPI 3 reset Set and cleared by software. 0: No effect 1: Reset SPI3
Bit 14	SPI2RST: SPI 2 reset Set and cleared by software. 0: No effect 1: Reset SPI2
Bit 13:12	Reserved. Always read as 0.
Bit 11	WWDGRST: Window watchdog reset Set and cleared by software. 0: No effect 1: Reset Window watchdog
Bit 10:9	Reserved. Always read as 0.

Bit 8	TMR14RST: Timer 14 reset Set and cleared by software. 0: No effect 1: Reset TMR 14 timer
Bit 7	TMR13RST: Timer 13 reset Set and cleared by software. 0: No effect 1: Reset TMR 13 timer
Bit 6	TMR12RST: Timer 12 reset Set and cleared by software. 0: No effect 1: Reset TMR 12 timer
Bit 5	TMR7RST: Timer 7 reset Set and cleared by software. 0: No effect 1: Reset TMR 7 timer
Bit 4	TMR6RST: Timer 6 reset Set and cleared by software. 0: No effect 1: Reset TMR 6 timer
Bit 3	TMR5RST: Timer 5 reset Set and cleared by software. 0: No effect 1: Reset TMR 5 timer
Bit 2	TMR4RST: Timer 4 reset Set and cleared by software. 0: No effect 1: Reset TMR 4 timer
Bit 1	TMR3RST: Timer 3 reset Set and cleared by software. 0: No effect 1: Reset TMR 3 timer
Bit 0	TMR2RST: Timer 2 reset Set and cleared by software. 0: No effect 1: Reset TMR 2 timer

3.3.6 AHB Peripheral Clock Enable Register (RCC_AHBEN)

Address offset: 0x14

Reset value: 0x0000 0014

Access: No wait state. Word, half-word, and byte access.

Note: If peripheral clock is not enabled, the software will not be able to read the value of peripheral register, and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	SDIO2 EN	SDIO1 EN	Reserv ed	XMC EN	Res erve d	CCE	Res erve d	FLAS HEN	Re se rv ed	SRAM EN	DMA2 EN	DMA1 EN			
res	rw	rw	res	rw	res	rw	res	rw	res	rw	rw	rw			
Bit 31:12		Reserved. Always read as 0.													

Bit 11	SDIO2EN: SDIO2 clock enable Set and cleared by software. 0: SDIO2 clock is disabled. 1: SDIO2 clock is enabled.
Bit 10	SDIO1EN: SDIO1 clock enable Set and cleared by software. 0: SDIO1 clock is disabled. 1: SDIO1 clock is enabled.
Bit 9	Reserved. Always read as 0.
Bit 8	XMCEN: XMC clock enable Set and cleared by software. 0: XMC clock is disabled. 1: XMC clock is enabled.
Bit 7	Reserved. Always read as 0.
Bit 6	CCE: CRC clock enable Set and cleared by software. 0: CRC clock is disabled. 1: CRC clock is enabled.
Bit 5	Reserved. Always read as 0.
Bit 4	FLASHEN: FLITF clock enable Set and cleared by software to enable/disable FLITF clock in Sleep mode. 0: FLITF clock is disabled in Sleep mode. 1: FLITF clock is enabled in Sleep mode.
Bit 3	Reserved. Always read as 0.
Bit 2	SRAMEN: SRAM interface clock enable Set and cleared by software to enable/disable SRAM clock in Sleep mode. 0: SRAM clock is disabled in Sleep mode. 1: SRAM clock is enabled in Sleep mode.
Bit 1	DMA2EN: DMA2 clock enable Set and cleared by software. 0: DMA2 clock is disabled. 1: DMA2 clock is enabled.
Bit 0	DMA1EN: DMA1 clock enable Set and cleared by software. 0: DMA1 clock is disabled. 1: DMA1 clock is enabled.

3.3.7 APB2 Peripheral Clock Enable Register (RCC_APB2EN)

Address offset: 0x18

Reset value: 0x0000 0000

Access: Word, half-word, and byte access.

Usually no waiting state, but if peripherals on APB2 bus are accessed, waiting states will be inserted until the end of APB2 peripheral access.

Note: If peripheral clock is not enabled, the software will not be able to read the value of peripheral register, and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						TMR11 EN	TMR10 EN	TMR9 EN	Reserved		TMR15 EN				
						res			rw	rw	rw	res		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TMR8 EN	SPI1 EN	TMR1 EN	ADC2 EN	ADC1 EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN	Reserved	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw

Bit 31:22	Reserved. Always read as 0.
Bit 21	TMR11EN: Timer 11 clock enable Set and cleared by software. 0: TMR11 timer clock is disabled. 1: TMR11 timer clock is enabled.
Bit 20	TMR10EN: Timer 10 clock enable Set and cleared by software. 0: TMR10 timer clock is disabled. 1: TMR10 timer clock is enabled.
Bit 19	TMR9EN: Timer 9 clock enable Set and cleared by software. 0: TMR9 timer clock is disabled. 1: TMR9 timer clock is enabled.
Bit 18:17	Reserved. Always read as 0.
Bit 16	TMR15EN: Timer 15 clock enable Set and cleared by software. 0: TMR15 timer clock is disabled. 1: TMR15 timer clock is enabled.
Bit 15	ADC3EN: ADC 3 interface clock enable Set and cleared by software. 0: ADC3 interface clock is disabled. 1: ADC3 interface clock is enabled.
Bit 14	USART1EN: USART1 clock enable Set and cleared by software. 0: USART1 clock is disabled. 1: USART1 clock is enabled.
Bit 13	TMR8EN: Timer 8 clock enable Set and cleared by software. 0: TMR8 timer clock is disabled. 1: TMR8 timer clock is enabled.
Bit 12	SPI1EN: SPI 1 clock enable Set and cleared by software. 0: SPI1 clock is disabled. 1: SPI1 clock is enabled.
Bit 11	TMR1EN: Timer 1 clock enable Set and cleared by software. 0: TMR1 timer clock is disabled. 1: TMR1 timer clock is enabled.
Bit 10	ADC2EN: ADC 2 interface clock enable Set and cleared by software. 0: ADC2 interface clock is disabled. 1: ADC2 interface clock is enabled.
Bit 9	ADC1EN: ADC 1 interface clock enable Set and cleared by software. 0: ADC1 interface clock is disabled. 1: ADC1 interface clock is enabled.
Bit 8	GPIOGEN: I/O port G clock enable Set and cleared by software. 0: IO port G clock is disabled. 1: IO port G clock is enabled.
Bit 7	GPIOFEN: I/O port F clock enable Set and cleared by software. 0: IO port F clock is disabled. 1: IO port F clock is enabled.

Bit 6	GPIOEEN: I/O port E clock enable Set and cleared by software. 0: IO port E clock is disabled. 1: IO port E clock is enabled.
Bit 5	GIODEN: I/O port D clock enable Set and cleared by software. 0: IO port D clock is disabled. 1: IO port D clock is enabled.
Bit 4	GPIOCEN: I/O port C clock enable Set and cleared by software. 0: IO port C clock is disabled. 1: IO port C clock is enabled.
Bit 3	GPIOBEN: I/O port B clock enable Set and cleared by software. 0: IO port B clock is disabled. 1: IO port B clock is enabled.
Bit 2	GPIOAEN: I/O port A clock enable Set and cleared by software. 0: IO port A clock is disabled. 1: IO port A clock is enabled.
Bit 1	Reserved. Always read as 0.
Bit 0	AFOEN: Alternate function I/O clock enable Set and cleared by software. 0: Alternate function IO clock is disabled. 1: Alternate function IO clock is enabled.

3.3.8 APB1 Peripheral Clock Enable Register (RCC_APB1EN)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: Word, half-word, and byte access.

Usually no waiting state, but if peripherals on APB1 bus are accessed, waiting states will be inserted until the end of APB1 peripheral access.

Note: If peripheral clock is not enabled, the software will not be able to read the value of peripheral register, and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	DAC EN	PWR EN	BKP EN	I2C3 EN	CAN EN	Reserved	USB EN	I2C2 EN	I2C1 EN	UART5 EN	UART4 EN	USART3 EN	USART2 EN	SPI4 EN	
res	res	rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 EN	SPI2 EN	Reserved	WWDG EN	Reserved	TMR14 EN	TMR13 EN	TMR12 EN	TMR7 EN	TMR6 EN	TMR5 EN	TMR4 EN	TMR3 EN	TMR2 EN		
rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:30		Reserved. Always read as 0.													
Bit 29		DACEN: DAC interface clock enable Set and cleared by software. 0: DAC interface clock is disabled. 1: DAC interface clock is enabled.													
Bit 28		PWREN: Power interface clock enable Set and cleared by software. 0: Power interface clock is disabled. 1: Power interface clock is enabled.													

Bit 27	BKPen : Backup interface clock enable Set and cleared by software. 0: Backup interface clock is disabled. 1: Backup interface clock is enabled.
Bit 26	I2C3EN : I2C 3 clock enable Set and cleared by software. 0: I2C 3 clock is disabled. 1: I2C 3 clock is enabled.
Bit 25	CANEN : CAN clock enable Set and cleared by software. 0: CAN clock is disabled. 1: CAN clock is enabled.
Bit 24	Reserved. Always read as 0.
Bit 23	USBEN : USB clock enable Set and cleared by software. 0: USB clock is disabled. 1: USB clock is enabled.
Bit 22	I2C2EN : I2C 2 clock enable Set and cleared by software. 0: I2C 2 clock is disabled. 1: I2C 2 clock is enabled.
Bit 21	I2C1EN : I2C 1 clock enable Set and cleared by software. 0: I2C 1 clock is disabled. 1: I2C 1 clock is enabled.
Bit 20	UART5EN : UART 5 clock enable Set and cleared by software. 0: UART5 clock is disabled. 1: UART5 clock is enabled.
Bit 19	UART4EN : UART 4 clock enable Set and cleared by software. 0: UART4 clock is disabled. 1: UART4 clock is enabled.
Bit 18	USART3EN : USART 3 clock enable Set and cleared by software. 0: USART3 clock is disabled. 1: USART3 clock is enabled.
Bit 17	USART2EN : USART 2 clock enable Set and cleared by software. 0: USART2 clock is disabled. 1: USART2 clock is enabled.
Bit 16	SPI4EN : SPI 4 clock enable Set and cleared by software. 0: SPI 4 clock is disabled. 1: SPI 4 clock is enabled.
Bit 15	SPI3EN : SPI 3 clock enable Set and cleared by software. 0: SPI 3 clock is disabled. 1: SPI 3 clock is enabled.
Bit 14	SPI2EN : SPI 2 clock enable Set and cleared by software. 0: SPI 2 clock is disabled. 1: SPI 2 clock is enabled.
Bit 13:12	Reserved. Always read as 0.

Bit 11	WWDGEN : Window watchdog clock enable Set and cleared by software. 0: Window watchdog clock is disabled. 1: Window watchdog clock is enabled.
Bit 10:6	Reserved. Always read as 0.
Bit 5	TMR7EN : Timer 7 clock enable Set and cleared by software. 0: Timer 7 clock is disabled. 1: Timer 7 clock is enabled.
Bit 4	TMR6EN : Timer 6 clock enable Set and cleared by software. 0: Timer 6 clock is disabled. 1: Timer 6 clock is enabled.
Bit 3	TMR5EN : Timer 5 clock enable Set and cleared by software. 0: Timer 5 clock is disabled. 1: Timer 5 clock is enabled.
Bit 2	TMR4EN : Timer 4 clock enable Set and cleared by software. 0: Timer 4 clock is disabled. 1: Timer 4 clock is enabled.
Bit 1	TMR3EN : Timer 3 clock enable Set and cleared by software. 0: Timer 3 clock is disabled. 1: Timer 3 clock is enabled.
Bit 0	TMR2EN : Timer 2 clock enable Set and cleared by software. 0: Timer 2 clock is disabled. 1: Timer 2 clock is enabled.

3.3.9 Backup Domain Control Register (RCC_BDC)

Address offset: 0x20

Reset value: 0x0000 0000, can only be reset by backup domain reset.

Access: 0 to 3 waiting state. Word, half-word, and byte access. Wait states are inserted in the case of successive accesses to this register.

Note: LSEEN, LSEBYP, RTCSEL, and RTCEN bits of the backup domain control register (RCC_BDC) are in the backup domain. As a result, these bits are write protected after reset, and can only be modified by setting the DBP bit in the power control register (PWR_CTRL). Please refer to [Section 4.1](#) for further information. These bits could be reset only by backup domain reset (see [Section 3.1.3](#)). Any internal or external reset does not affect these bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														BDRST	
res															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCEN	Reserved			RTCSEL[1:0]		Reserved				LSE BYPS	LSE STBL	LSEEN			
rw	res			rw	rw	res				rw	r	rw			
Bit 31:17		Reserved. Always read as 0.													

Bit 16	BDRST : Backup domain software reset Set and cleared by software. 0: Reset is not activated. 1: Reset the entire backup domain.
Bit 15	RTCEN : RTC clock enable Set and cleared by software. 0: RTC clock is disabled. 1: RTC clock is enabled.
Bit 14:10	Reserved. Always read as 0.
Bit 9:8	RTCSEL[1:0] : RTC clock source selection Set by software to select the clock source for the RTC. Once the RTC clock source is selected, it cannot be changed until the next backup domain reset. It can be reset by setting the BDRST bit. 00: No clock 01: LSE oscillator is used as RTC clock. 10: LSI oscillator is used as RTC clock. 11: HSE oscillator divided by 128 is used as RTC clock.
Bit 7:3	Reserved. Always read as 0.
Bit 2	LSEBYP : External low-speed oscillator bypass Set and cleared by software in Debug mode to bypass LSE. This bit can be written only when the external 32 kHz oscillator is disabled. 0: LSE clock is not bypassed. 1: LSE clock is bypassed.
Bit 1	LSESTBL : External low-speed oscillator ready Set and cleared by hardware to indicate if external 32 kHz oscillator is stable. After the LSEEN bit is cleared, this bit is cleared after 6 external low-speed oscillator clock cycles. 0: External 32 kHz oscillator is not ready. 1: External 32 kHz oscillator is ready.
Bit 0	LSEEN : External low-speed oscillator enable Set and cleared by software. 0: External 32 kHz oscillator is OFF. 1: External 32 kHz oscillator is ON.

3.3.10 Control/Status register (RCC_CTRLSTS)

Address offset: 0x24

Reset value: 0x0C00 0000, reset flag can only be cleared by power reset, while others are cleared by system reset.

Access: 0 to 3 waiting state. Word, half-word, and byte access. Wait states are inserted in the case of successive accesses to this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP RSTF	WWDG RSTF	IWDG RSTF	SW RSTF	POR RSTF	PIN RSTF	Reserved	RST FC	Reserved							
rw	rw	rw	rw	rw	rw	rw	rw	res							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															LSI STBL LSI EN
res															r rw

Bit 31	LPRSTF : Low-power reset flag Set by hardware when a low-power management reset occurs. It is cleared by writing to the RSTFC bit. 0: No low-power management reset occurs. 1: Low-power management reset occurs. For further information on low-power management reset, please refer to Section 3.1.1 “Low-Power Management Reset.”
--------	---

Bit 30	WWDGRSTF: Window watchdog reset flag Set by hardware when a window watchdog reset occurs. It is cleared by writing to the RSTFC bit. 0: No window watchdog reset occurs. 1: Window watchdog reset occurs.
Bit 29	IWDGRSTF: Independent watchdog reset flag Set by hardware when an independent watchdog reset from VDD domain occurs. It is cleared by writing to the RSTFC bit. 0: No independent watchdog reset occurs. 1: Independent watchdog reset occurs.
Bit 28	SWRSTF: Software reset flag Set by hardware when a software reset occurs. It is cleared by writing to the RSTFC bit. 0: No software reset occurs. 1: Software reset occurs.
Bit 27	PORRSTF: POR/PDR reset flag Set by hardware when a POR/PDR reset occurs. It is cleared by writing to the RSTFC bit. 0: No POR/PDR reset occurs. 1: POR/PDR Reset occurs.
Bit 26	PINRSTF: NRST PIN reset flag Set by hardware when a reset in the NRST pin occurs. It is cleared by writing to the RSTFC bit. 0: No NRST pin reset occurs. 1: NRST pin reset occurs.
Bit 25	Reserved. Return 0 after being read.
Bit 24	RSTFC: Reset flag clear Set by software to clear the reset flags. 0: No effect 1: Clear the reset flags.
Bit 23:2	Reserved. Return 0 after being read.
Bit 1	LSISTBL: Internal low-speed oscillator ready Set and cleared by hardware to indicate if the internal 40 kHz RC oscillator is stable. After the LSIEN bit is cleared, LSISTBL is cleared after 3 internal 40 kHz RC oscillator clock cycles. 0: Internal 40 kHz RC oscillator clock is not ready. 1: Internal 40 kHz RC oscillator clock is ready.
Bit 0	LSIEN: Internal low-speed oscillator enable Set and reset by software. 0: Internal 40 kHz RC oscillator is OFF. 1: Internal 40 kHz RC oscillator is ON.

3.3.11 Additional Register (RCC_MISC)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved				USB 768B		Reserved				CLKOUT[3]													
res							rw								rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Reserved								HSICAL_KEY[7:0]															
res								rw															
Bit 31:25		Reserved. Return 0 after being read.																					
Bit 24		USB768B: USB buffer size 0: The buffer is 512 byte. 1: The buffer is 768 byte.																					

Bit 23:17	Reserved. Return 0 after being read.
Bit 16	CLKOUT[3]: Microcontroller clock output Used with RCC_CFG register bit 26:24
Bit 15:8	Reserved. Return 0 after being read.
Bit 7:0	HSICAL_KEY[7:0]: HSICAL written value HSICAL [7:0] can be written only if this bit is 0x5A.

4 Backup Registers (BKPR)

4.1 BKPR Introduction

The backup registers are 42 16-bit registers for storing 84 bytes of user application data. These registers are located in the backup domain and are powered-on by VDD/VBAT. These registers will not be reset when the device wakes up from Standby mode or by system reset and power reset.

In addition, the BKPR control registers can be used to control tamper detection and RTC calibration function. After reset, accesses to the backup registers and RTC are disabled, and the backup domain is protected against possible write access. To enable accesses to the backup registers and the RTC, proceed as follows:

- Enable the power and backup interface clocks by setting the PWREN and BKPN bits in the RCC_APB1EN register.
- Set the DBP bit in the power control register (PWR_CTRL) to enable access to the backup registers and the RTC.

4.2 BKPR Main Features

- 84 bytes data backup registers
- Status/Control register which can control tamper detection with interrupt capability
- Calibration register which can store the RTC calibration value
- RTC calibration clock, RTC alarm pulse, or second pulse output on the TAMPER pin, PC13 (when this pin is not used for tamper detection).

4.3 BKPR Function Overview

4.3.1 Tamper Detection

The TAMPER pin generates a tamper detection event when the pin changes from 0 to 1 or from 1 to 0. (It depends on the TPALV bit in the backup control register (BKPR_CTRL).) A tamper detection event will reset all data backup registers.

However, to prevent tamper events from losing, the tamper detection signal is the edge detection signal logically ANDed with the tamper detection enable bit, to detect tamper events that occur before the TAMPER pin is enabled.

- When TPALV = 0: If the TAMPER pin is already high before enabled (by setting the TPEN bit), an extra tamper event is detected as soon as the TAMPER function is enabled (even if there was no rising edge on the TAMPER pin after TPEN was set).
- When TPALV = 1: If the TAMPER pin is already low before enabled (by setting the TPEN bit), an extra tamper event is detected as soon as the TAMPER function is enabled (even if there was no falling edge on the TAMPER pin after TPEN was set).

By setting the TPIEN bit in the BKPR_CTRLSTS register, an interrupt is generated when a tamper event is detected. After a tamper event has been detected and cleared, the TAMPER pin should be disabled and then re-enabled by the TPEN bit before writing to the backup data registers again. This can prevent software from writing to the backup data registers when the TAMPER pin value still indicates a tamper detection. This is equivalent to a level detection on the TAMPER pin.

Note: Tamper detection is still active when VDD power is switched off. To avoid unnecessary resetting of the data backup registers, the TAMPER pin should be externally tied to the correct level.

4.3.2 RTC Calibration

For the measurement purpose, the RTC clock with a frequency divided by 64 can be output on the TAMPER pin. This is enabled by setting the OT1CAL bit in the RTC clock calibration register (BKPR_RTCCAL).

The clock can be slowed down up to 121 ppm by configuring CAL[6:0] bits.

4.4 BKPR Registers

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

BKPR register is a 16-bit addressable register.

Table 4-1 BKPR Register Map and Reset Value

	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x54	BKPR_DR16	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x58	BKPR_DR17	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x5C	BKPR_DR18	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x60	BKPR_DR19	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x64	BKPR_DR20	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x68	BKPR_DR21	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x6C	BKPR_DR22	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x70	BKPR_DR23	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x74	BKPR_DR24	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x78	BKPR_DR25	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x7C	BKPR_DR26	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x80	BKPR_DR27	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x84	BKPR_DR28	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x88	BKPR_DR29	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x8C	BKPR_DR30	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x90	BKPR_DR31	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x94	BKPR_DR32	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x98	BKPR_DR33	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0x9C	BKPR_DR34	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xA0	BKPR_DR35	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xA4	BKPR_DR36	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xA8	BKPR_DR37	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xAC	BKPR_DR38	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xB0	BKPR_DR39	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0xB4	BKPR_DR40	Reserved	D[15:0]
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0xB8	BKPR_DR41	Reserved	D[15:0]											
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0											
0xBC	BKPR_DR42	Reserved	D[15:0]											
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0											

4.4.1 Backup Data Register x (BKPR_DRx) (x = 1, ..., 42)

Address offset: 0x04 to 0x28, 0x40 to 0xBC

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0		D[15:0]: Backup data These bits can be written with user data. <i>Note:</i> The BKPR_DRx registers are not reset by system reset or power reset or when the device wakes up from Standby mode. They are reset by a backup domain reset or by a TAMPER pin event (if the TAMPER pin function is activated).													

4.4.2 RTC Clock Calibration Register (BKPR_RTCCAL)

Address offset: 0x2C

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							ASOS	ASOE	CCO	CAL[6:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:10		Reserved. Always read as 0.													
Bit 9		ASOS : Alarm or second output selection When the ASOS bit is set, the ASOS bit can select either the RTC second pulse signal or the alarm pulse signal as signal output on the TAMPER pin. 0: RTC Alarm pulse output is selected. 1: Second pulse output is selected. <i>Note:</i> This bit is reset only by a backup domain reset.													
Bit 8		ASOE : Alarm or second output enable According to the setting of the ASOS bit, this bit allows either the RTC alarm pulse signal or the second pulse signal output on the TAMPER pin. The output pulse duration is one RTC clock cycle. The TAMPER function cannot be enabled once the ASOE bit is set. <i>Note:</i> This bit is reset only by a backup domain reset.													
Bit 7		CCO : Calibration clock output 0: No effect 1: Setting this bit can output the RTC clock with a frequency divided by 64 on the TAMPER pin. The TAMPER pin must be disabled when the CCO bit is set, to avoid undesired tamper detection. <i>Note:</i> This bit is reset when the VDD supply is powered off.													
Bit 6:0		CAL[6:0] : Calibration value The calibration value indicates the number of clock pulses that will be ignored every 2^{20} clock pulses. This allows the calibration of the RTC, slowing down the clock by steps of $1000000/2^{20}$ ppm. The RTC clock can be slowed down from 0 ppm to 121 ppm.													

4.4.3 Backup Control Register (BKPR_CTRL)

Address offset: 0x30

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
								Reserved														TPAL	TPE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 15:2		Reserved. Always read as 0.																					
Bit 1		TPALV: TAMPER pin active level 0: A high level on the TAMPER pin resets all data backup registers (if the TPEN bit is set). 1: A low level on the TAMPER pin resets all data backup registers (if the TPEN bit is set).																					
Bit 0		TPE: TAMPER pin enable 0: The TAMPER pin is free for general purpose I/O. 1: The TAMPER pin is enabled for tamper detection.																					

Note: Setting the TPALV and TPEN bits at the same time is always safe. However, resetting both at the same time will generate a spurious tamper event. Therefore, changing the TPALV bit only when the TPEN bit is 0 is suggested.

4.4.4 Backup Control/Status Register (BKPR_CTRLSTS)

Address offset: 0x34

Reset value: 0x0000 0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0									
								Reserved														TPIE	CTI	CTE
rw	rw	rw	rw	rw	rw	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
Bit 15:0		Reserved. Always read as 0.																						
Bit 9		TIF: Tamper interrupt flag This bit is set by hardware when a tamper event is detected and the TPIEN bit is set. This bit is cleared by writing 1 to the CTPIF bit (also clears the interrupt). This bit is cleared if the TPIEN bit is reset. 0: No tamper interrupt occurred. 1: A tamper interrupt occurred. Note: This bit is reset only by system reset or by waking up from Standby mode.																						
Bit 8		TEF: Tamper event flag This bit is set by hardware when a tamper event is detected. It is cleared by writing 1 to the CTPEF bit. 0: No tamper event occurred. 1: A tamper event occurred. Note: A tamper event resets all the BKPR_DRx registers. They are held in reset as long as the TEF bit is set. If a write to the BKPR_DRx registers is performed when this bit is set, the written value will not be stored.																						
Bit 7:3		Reserved. Always read as 0.																						
Bit 2		TPIE: Tamper pin interrupt enable 0: Tamper interrupt is disabled 1: Tamper interrupt is enabled. (The TPEN bit must also be set in the BKPR_CTRL register.) Note 1: A tamper interrupt does not wake up the core from low-power mode. Note 2: This bit is reset only by system reset or by waking up from Standby mode.																						
Bit 1		CTI: Clear tamper interrupt This bit is write only, and is always read as 0. 0: No effect 1: Clear the tamper interrupt and the TIF tamper interrupt flag.																						
Bit 0		CTE: Clear tamper event This bit is write only, and is always read as 0. 0: No effect 1: Clear the TPEF tamper event flag (and reset the tamper detector).																						

5 Flash Memory Controller (FMC)

5.1 FMC Introduction

Embedded Flash memory can be used for In-Circuit Programming (ICP) or In-Application Programming (IAP).

In-Circuit Programming (ICP) is used to update all the contents in the Flash memory. It can download user application to the microcontroller through JTAG, SWD protocol, or boot loader. ICP is a fast and effective programming which is free from the inconvenience of package and sockets.

Different from ICP, In-Application Programming (IAP) can use any communication interface supported by the microcontroller (such as I/O interface, USB, CAN, USART, I²C, SPI, etc.) to download application or data to the memory. IAP allows users to re-program contents in the Flash memory while applications are running. However, IAP requires that some of the applications are already programmed in the Flash memory with ICP.

Flash memory interface realizes the access to instructions and data on AHB protocols. It speeds up the access to the memory with the help of memory prefetch buffer. Flash memory interface also realizes the logic circuits required by Flash memory program and erase under all operating voltages, including access, write protection, and control of option bytes.

5.2 Main Features

- Up to 1024 KB of Flash memory
- Two banks are adopted. Bank 1 is used for the first 512 KB, and Bank 2 is for the rest capacity.
- Flash memory interface of external memory (Bank 3) provided by SPIM interface
- Read interface with prefetch buffer
- Option byte loader
- Flash memory program/erase operation
- Read/Write protection
- Low-power mode

5.2.1 Flash Memory Architecture

Flash memory includes 1024K main Flash memory block and an information block. Please refer to [Table 5-1](#):

Table 5-1 Flash Memory Architecture (256K and beyond)

Block	Name	Address Range	Size (bytes)
Main Flash Memory	Page 0	0x0800 0000 – 0x0800 07FF	2 K
	Page 1	0x0800 0800 – 0x0800 0FFF	2 K
	Page 2	0x0800 1000 – 0x0800 17FF	2 K
	Page 3	0x0800 1800 – 0x0800 1FFF	2 K
	Page 4	0x0800 2000 – 0x0800 27FF	2 K
	.	.	.
	Page 255	0x0807 F800 – 0x0807 FFFF	2 K
	Page 256	0x0808 0000 – 0x0808 07FF	2 K
	Page 257	0x0808 0800 – 0x0808 0FFF	2 K
	Page 258	0x0808 1000 – 0x0808 17FF	2 K
Bank 2 3584 KB	Page 259	0x0808 1800 – 0x0808 1FFF	2 K
	Page 260	0x0808 2000 – 0x0808 27FF	2 K
	.	.	.
	Page 2048	0x083F F800 – 0x083F FFFF	2 K

External Memory	Bank 3		0x0840 0000 – 0x093F FFFF	16 M
		Boot Loader	0x1FFF B000 – 0x1FFF F7FF	18 K
		User Option Bytes	0x1FFF F800 – 0x1FFF F82F	48
Flash Memory Interface Register		FLASH_ACR	0x4002 2000 – 0x4002 2003	4
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
		FLASH_STS	0x4002 200C – 0x4002 200F	4
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
		FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
		Reserved	0x4002 2018 – 0x4002 201B	4
		FLASH_UOB	0x4002 201C – 0x4002 201F	4
		FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
		Reserved	0x4002 2024 - 0x4002 2043	32
		FLASH_FCKEY2	0x4002 2044 - 0x4002 2047	4
		Reserved	0x4002 2048 - 0x4002 204B	4
		FLASH_STS2	0x4002 204C - 0x4002 204F	4
		FLASH_CTRL2	0x4002 2050 - 0x4002 2053	4
		FLASH_ADDR2	0x4002 2054 - 0x4002 2057	4
		Reserved	0x4002 2058 - 0x4002 2083	44
		FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
		FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
		FLASH_STS3	0x4002 208C - 0x4002 208F	4
		FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
		FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
		FLASH_DA	0x4002 2098 – 0x4002 209B	4

Flash Memory Architecture (128K)

Block		Name	Address Range	Size (bytes)
Main Flash Memory	Bank1 128KB	Page 0	0x0800 0000 – 0x0800 03FF	1K
		Page 1	0x0800 0400 – 0x0800 07FF	1K
		Page 2	0x0800 0800 – 0x0800 0BFF	1K
		.	.	.
		Page 127	0x0801 FC00 – 0x0801 FFFF	1K
External Memory	Bank3		0x0840 0000 – 0x093F FFFF	16M
Information Block		Boot Loader	0xFFFF B000 – 0xFFFF F7FF	18K
		User Option Bytes	0xFFFF F800 – 0xFFFF F82F	48
Flash Memory Interface Register		FLASH_ACR	0x4002 2000 – 0x4002 2003	4
		FLASH_FCKEY	0x4002 2004 – 0x4002 2007	4
		FLASH_OPTKEYR	0x4002 2008 – 0x4002 200B	4
		FLASH_STS	0x4002 200C – 0x4002 200F	4
		FLASH_CTRL	0x4002 2010 – 0x4002 2013	4
		FLASH_ADDR	0x4002 2014 – 0x4002 2017	4
		Reserved	0x4002 2018 – 0x4002 201B	4
		FLASH_UOB	0x4002 201C – 0x4002 201F	4
		FLASH_WRPRT	0x4002 2020 – 0x4002 2023	4
		Reserved	0x4002 2024 - 0x4002 2083	96
		FLASH_FCKEY3	0x4002 2084 - 0x4002 2087	4
		FLASH_SELECT	0x4002 2088 - 0x4002 208B	4
		FLASH_STS3	0x4002 208C - 0x4002 208F	4
		FLASH_CTRL3	0x4002 2090 - 0x4002 2093	4
		FLASH_ADDR3	0x4002 2094 - 0x4002 2097	4
		FLASH_DA	0x4002 2098 – 0x4002 209B	4

Flash memory is a 32-bit wide memory unit, which stores codes and data constants. Every Flash memory block in AT32F403 microcontroller has a specific address.

Information block is divided into two parts:

- System memory is used for boot loader stored in the system memory bootstrapping mode. This area is reserved only for AT32F403, and the boot loader uses USART1,USART2 or USB (DFU) serial interface to program the Flash memory. This area is programmed and locked during manufacturing to prevent users from erase or write.
- Option bytes

Write to main memory and information block is controlled by embedded Flash program/erase controller (FPEC). High voltages of program and erase are generated internally.

Flash memory provides two kinds of protection to avoid illegal access (read, write, and erase):

- Page write protection
- Read protection (Please refer to [Section 5.3.3](#) for further information.)

During Flash memory write operation, any read operation to the Flash memory will lock the bus. Read operation can only be executed successfully after write operation is completed. That is, when write or erase operation is in process, reading code or data is not allowed.

When Flash programming (write or erase) is in process, internal RC oscillator (HSI) must be ON. Flash memory can be programmed with ICP or IAP.

Note: In low-power mode, all operations in the Flash memory are stopped.

5.2.2 External Flash Memory Organization

Flash memory provides external Flash memory block, which controls external SPI Flash memory (Bank 3) through SPIM transmission interface. The maximum capacity for read and write is 16 MB, and the operation is as same as Bank 1 and Bank 2. For additional encryption function, users can decide whether to encrypt the data according to option byte access, and control the range of encryption with the FLASH_DA register.

SPIM = External SPI Flash memory expansion (application execution/data storing/program and data can be encrypted). SPIM and USB interface use the same pin, and the two functions cannot be used at the same time.

Note: *External Flash memory organization only supports word/half-word operation.*

Figure 5-1 Programming Process

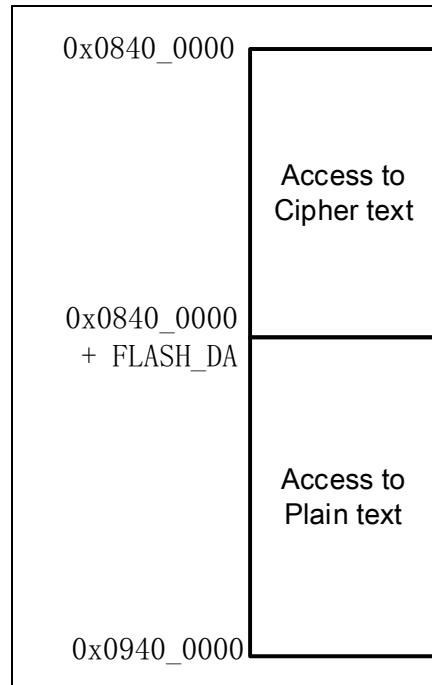
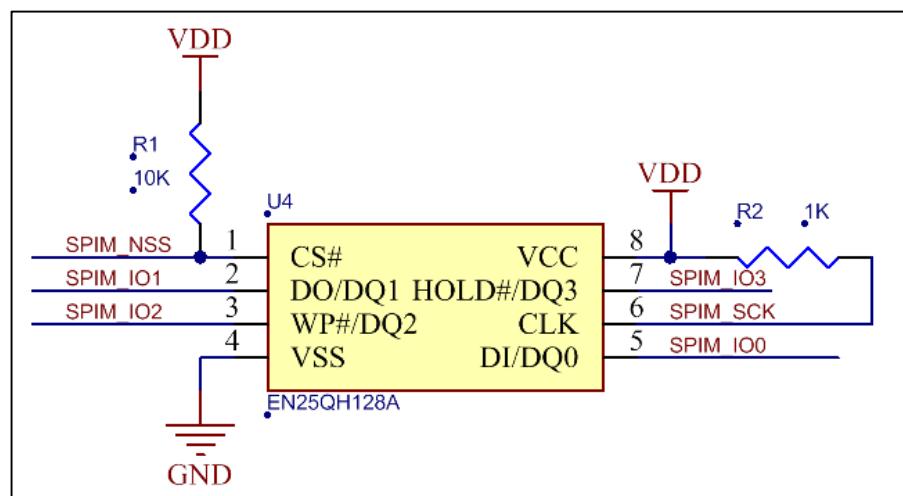


Figure 5-2 Reference Circuit for External Memory (Bank 3)



5.3 Function Overview

5.3.1 Read Operation

The embedded Flash memory block can be addressed directly in the common memory space. Any 32-bit data read operation can access the contents in Flash memory block and obtain the corresponding data.

Read interface includes a read controller in the Flash memory and an AHB interface connected to the CPU. The main function of this interface is to generate control signal of reading Flash memory and to prefetch the instruction block required by the CPU. Prefetch instruction block is only used for fetching instructions on I-Code bus, and the data constants are accessed through D-Code bus. The access target of these two buses are the same Flash memory block, and access to D-Code has higher priority than prefetch instruction.

5.3.1.1 Instruction Fetch

Cortex®-M4F fetches instructions on I-Code bus and fetches data on D-Code bus. Prefetch instruction block can effectively increase the efficiency of I-Code bus access.

Prefetch buffer includes two data blocks, each of which is 8 bytes. Prefetch instruction (data) directly maps to the Flash memory, and since the size of data block is the same as the bandwidth of the Flash memory, reading prefetch instruction can be completed within a read cycle.

Setting prefetch buffer can make CPU execution faster, as when the CPU fetches one word, the next word is readily available in the prefetch buffer at the same time.

Prefetch buffer is ON after system reset.

5.3.1.2 D-Code Interface

D-Code interface includes simple AHB interface on CPU and logic circuit that requests access to the arbiter of Flash access controller. Access to D-code has higher priority than access to prefetch instruction. This interface uses access time regulator block of prefetch buffer.

5.3.1.3 Flash Access Controller

This block is the arbiter of the instruction prefetch request on I-Code and read request on D-Code interface. Request from D-Code interface has higher priority than request from I-Code.

5.3.2 Flash Program/Erase Controller (FPEC)

FPEC block controls Flash program/erase operation, and it includes 17 32-bit registers. As long as CPU does not access Flash memory, Flash operation will not delay CPU execution.

- FPEC key register (FLASH_FCKEY)
- Option bytes key register (FLASH_OPTKEYR)
- Flash status register (FLASH_STS)
- Flash control register (FLASH_CTRL)
- Flash address register (FLASH_ADDR)
- Option bytes register (FLASH_UOB)
- Write protection register (FLASH_WRPRT)
- FPEC key register 2 (FLASH_FCKEY2)
- Flash status register 2 (FLASH_STS2)
- Flash control register 2 (FLASH_CTRL2)
- Flash address register 2 (FLASH_ADDR2)
- FPEC key register 3 (FLASH_FCKEY3)
- Flash option register (FLASH_SELECT)
- Flash status register 3 (FLASH_STS3)
- Flash control register 3 (FLASH_CTRL3)
- Flash address register 3 (FLASH_ADDR3)
- Flash control register 3 (FLASH_CTRL3)
- Flash decode address register (FLASH_DA)

5.3.2.1 Key Value

There are three key values:

- RDPRTEN key = 0x00A5

- KEY1 = 0x45670123
- KEY2 = 0xCDEF89AB

5.3.2.2 Unlock the Flash Memory

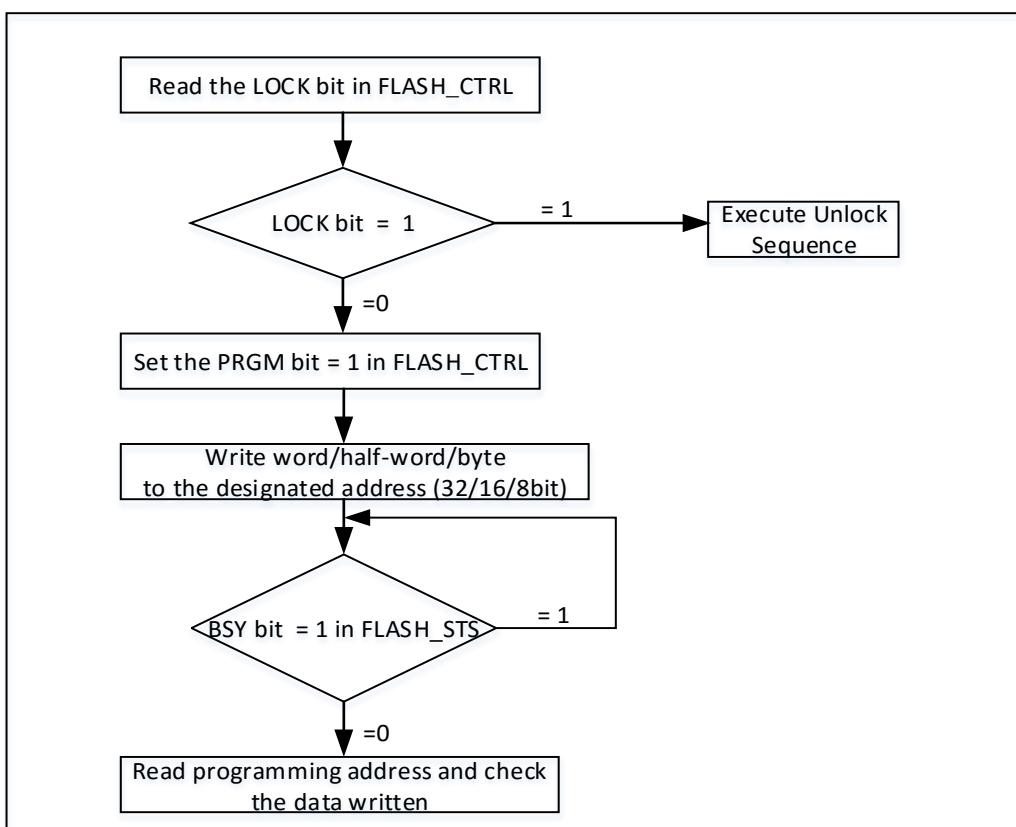
After reset, FPEC block is protected. FLASH_CTRLx register cannot be written. FPEC block can be unlocked by writing certain sequence to FLASH_FCKEYx register. This sequence is to write two key values (KEY1 and KEY2, see [Section 5.3.2.1](#)) to FLASH_FCKEYx. Incorrect sequence will lock FPEC block and FLASH_CTRLx register until next reset.

Writing incorrect key sequence also leads to bus error. Bus error occurs when the first key value written is not KEY1, or when the first key value written is KEY1 but the second key value written is not KEY2. FPEC block and FLASH_CTRLx register could be locked by setting the LOCK bit in the FLASH_CTRLx register. At this time, FPEC can be unlocked by writing the correct key values to FLASH_FCKEYx.

5.3.2.3 Main Flash Programming

The Flash memory can be programmed with 32/16/8 bits at a time. When the PRGM bit in FLASH_CTRLx register is '1', writing a word/half-word /byte to a Flash address will activate one programming operation. In the process of programming (the BSY bit is '1'), any read/write operation to the Flash memory will stop the CPU until this Flash programming is completed.

Figure 5-3 Process of the Programming



Standard programming

In this mode, CPU programs/writes Flash memory with standard half-word method, and the PRGM bit in the FLASH_CTRLx register must be set. FPEC first reads the contents of the designated address and checks if it is erased. If it is not erased, then programming is not executed, and warning on the PRGMFLR bit in the FLASH_STSx register is issued (The only exception is when the value being programmed is 0x0000, 0x0000 can be programmed successfully without setting the PRGMFLR bit). If the designated address is under write protection in FLASH_WRPRT, then programming is not executed and setting the WRPRTFLR bit in FLASH_STSx to issue a warning. When the PRCDN bit in the FLASH_STSx register becomes '1', the programming is completed.

The process of standard Flash programming is in the following order:

- Check the BSY bit in the FLASH_STSx register to confirm that there is no other

- programming operation in process;
- Set the PRGM bit in the FLASH_CTRLx register;
- Write the half-word to be programmed to the designated address;
- Wait until the BSY bit becomes '0';
- Read the address written and verify the data.

Note: When the BSY bit in the FLASH_STSx register is '1', write operation to any register is not allowed.

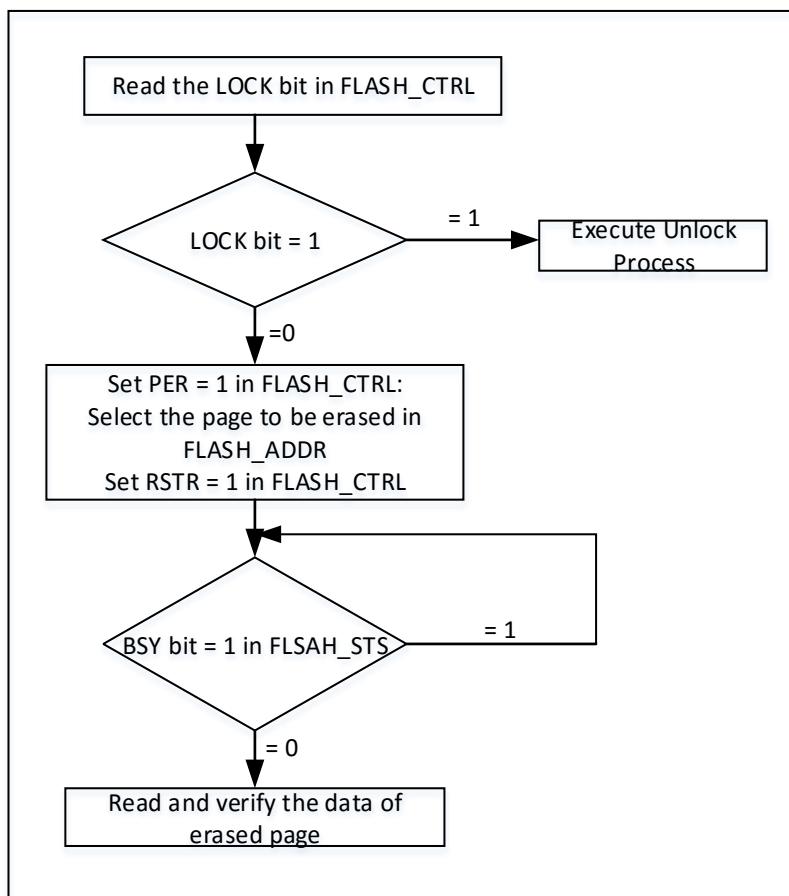
5.3.2.4 Flash Erase

Flash memory erase includes mass erase and page erase.

Any page in the Flash memory can be erased with the FPEC page erase function. Below should be followed during page erase:

- Check the BSY bit in the FLASH_STSx register to confirm that there is no other programming operation in process;
- Set the PGERS bit in the FLASH_CTRLx register;
- Select the page to be erased with the FLASH_ADDRx register;
- Set the RSTR bit in the FLASH_CTRLx register;
- Wait until the BSY bit becomes '0';
- Read and verify the erased page.

Figure 5-4 Process of Flash Memory Page Erase



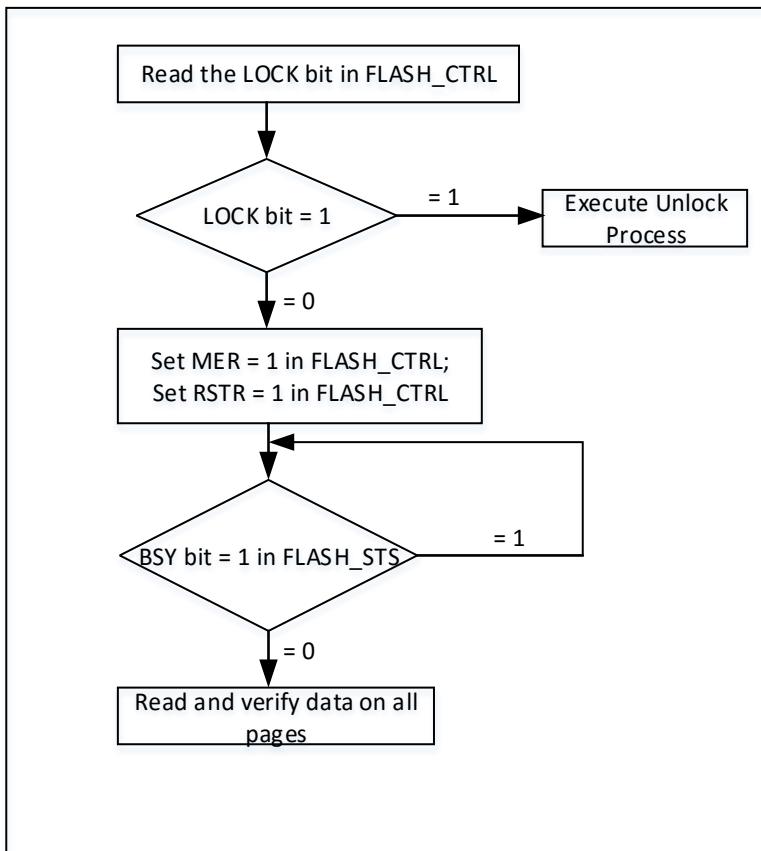
Mass Erase

Mass erase function can erase all the Flash memory in the user area. Information block will not be affected by this operation. The following process is recommended:

- Check the BSY bit in the FLASH_STSx register to confirm that there is no other programming operation in process;

- Set the CHPERS bit in the FLASH_CTRLx register;
- Set the RSTR bit in the FLASH_CTRLx register;
- Wait until the BSY bit becomes ‘0’;
- Read and verify all the pages.

Figure 5-5 Process of Flash Memory Mass Erase



5.3.2.5 Option Byte Programming

Option byte programming is different from common user address. The number of option bytes is only 24 bytes (4 bytes for write protection, 1 byte for read protection, 1 byte for configuration option, 8 bytes for storing user data). After unlocking FPEC, KEY1 and KEY 2 (See [Section 5.3.2.1](#)) should be written to the FLASH_OPTKEYR register respectively, and the UOBWE bit in the FLASH_CTRL register will be set. At this time, option byte programming can be executed: after setting the UOBPRGM bit in the FLASH_CTRL register, write the half-word to the designated address.

FPEC will first read the contents of option bytes in the designated address and checks if it is erased. If it is not, then the programming will not be executed and a warning on the WRPRTFLR bit in the FLASH_STS register will be issued. When the PRCDN bit in the FLASH_STS register becomes ‘1’, the programming is completed.

FPEC uses the low bytes in half-word to calculate high byte automatically (High bytes are the opposites of the low bytes), and activates the programming operation. This can ensure that the option bytes and their opposites are always correct.

The programming operation is in the following order:

- Check the BSY bit in the FLASH_STS register to confirm that there is no other programming operation in process;
- Unlock the UOBWE bit in the FLASH_CTRL register;
- Set the UOBPRGM bit in the FLASH_CTRL register;
- Write the half-word to be programmed to the designated address;
- Wait until the BSY bit becomes ‘0’;
- Read the address written and verify the data.

When the read protection option of Flash memory changes from “protected” to “unprotected”, a mass erase of user Flash memory will be executed automatically before reset read protection option. If users change other options aside from read protection, then mass erase will not be executed. This erase operation protects the contents of Flash memory from illegal read out.

Process of erase

The erase operation (OPTERASE) of option bytes is in the following order:

- Check the BSY bit in the FLASH_STS register to confirm that there is no other programming operation in process;
- Unlock the UOBWE bit in the FLASH_CTRL register;
- Set the UOBERS bit in the FLASH_CTRL register;
- Set the RSTR bit in the FLASH_CTRL register;
- Wait until the BSY bit becomes ‘0’;
- Read and verify the erased option bytes.

5.3.3 Protection

User code area in the Flash memory can prevent illegal read out and also protect the pages in the Flash memory from inadvertent operation when programming errors occur. The base unit of write protection is two pages for 256K and beyond, and four pages for 128K.

5.3.3.1 Write Protection

Write protection is executed every two pages.

If programming or erase operation is executed on a protected page, an error flag will be returned in the Flash status register (FLASH_STS).

The following steps can be used to disable write protection:

- Use the UOBERS bit in the Flash control register (FLASH_CTRL) to erase the whole option byte area;
- Write correct RDP code 0xA5 to enable read access;
- Reset the system and reload option bytes (including new WRPRTBMP[3:0] byte); write protection is disabled.

5.3.3.2 Read Protection

This protection is activated by setting the RDP option bytes, and RDP option bytes are loaded by system reset. When the protection byte is written with the corresponding values, the following protection will be executed:

- Only read operation from user code to main Flash memory is allowed (enabled by main Flash memory with non-Debug method).
- Page 0~3 (for 128K) or page 0~1 (for 256K and beyond) are automatically added with write protection, and other memories can be programmed with the codes executed in the main Flash memory (to enable functions like IAP or data storing), but write or erase operation are not allowed in Debug mode or after internal SRAM is enabled (except for mass erase).

All the functions that load codes from internal SRAM through JTAG/SWD and execute the codes are still valid, and can also be activated from internal SRAM through JTAG/SWD. This function can be used to disable read protection. When the option bytes of read protection turn to the unprotected values, mass erase process will be proceeded.

When RDP option bytes and their opposites include the following value pairs, the Flash memory is in protection:

Table 5-2 Flash Memory Protection Status

RDP byte value	Opposite of RDP value	Read Protection Status
0xFF	0xFF	Protected
0xA5	0x5A	Unprotected

Any value (except for 0xA5)	Opposite of any value	Protected
-----------------------------	-----------------------	-----------

Note: Erasing the option byte block will not cause automatic mass erase, since the result of erase 0xFF is equal to the protected status.

The process of disabling read protection is as follows:

- Erase the whole option byte area, and the read protection code (RDP) will turn to 0xFF. Read protection is still valid at this time;
- Write correct RDP code 0xA5 to disable memory protection, and this action will first cause mass erase to all user Flash memories.
- Reset (Power-on reset) and reload option bytes (and new RDP code). Read protection will be disabled at this time.

Note: Boot loader can also be used to disable the read protection (In this case, option bytes can be reloaded just by system reset).

5.3.3.3 Option Byte Block Write Protection

By default, option byte block can always be read and is under write protection. To execute write operation (program and erase), write correct key sequence (the same as LOCK) to OPTKEYR, and enable the write operation to option bytes. The UOBWE bit in the FLASH_CTRL register then shows that write operation is allowed. Write operation will be disabled if this bit is cleared.

5.3.4 Option Byte Description

Option bytes include 24 bytes, which are configured according to users' needs; for example, users can select hardware watchdog or software watchdog.

In option bytes, every 32-bit word is categorized into the following format:

Table 5-3 Option Byte Format

Bit 31:24	Bit 23:16	Bit 15:8	Bit 7:0
Opposite of option byte1	Option byte1	Opposite of option byte0	Option byte0

Table 5-4 shows how option bytes are organized in the option byte block:

Table 5-4 Information Block Organization

Address	[31: 24]	[23: 16]	[15: 8]	[7: 0]
0x1FF_F800	uUSER	USER	nRDP	RDP
0x1FF_F804	nData1	Data1	nData0	Data0
0x1FF_F808	nWRP1	WRP1	nWRP0	WRP0
0x1FF_F80C	nWRP3	WRP3	nWRP2	WRP2
0x1FF_F810	Reserved	Reserved	nEOPB0	EOPB0
0x1FF_F814	nData3	Data3	nData2	Data2
0x1FF_F818	nData5	Data5	nData4	Data4
0x1FF_F81C	nData7	Data7	nData6	Data6
0x1FF_F820	nBANK3KEY1	BANK3KEY1	nBANK3KEY0	BANK3KEY0
0x1FF_F824	nBANK3KEY3	BANK3KEY3	nBANK3KEY2	BANK3KEY2
0x1FF_F828	nBANK3KEY5	BANK3KEY5	nBANK3KEY4	BANK3KEY4
0x1FF_F82C	nBANK3KEY7	BANK3KEY7	nBANK3KEY6	BANK3KEY6

Table 5-5 User Option Byte Description

<p>RDP: Read Protection Option Byte</p> <p>Read protection can help users protect the codes stored in the Flash Memory. This function is enabled by setting one option byte in the information block. After writing the correct value (RDPRTEN Key=0x00A5) to this option byte, the Flash memory allows read access.</p>
<p>USR: User option byte. This byte is used for the configuration of the following functions:</p> <ul style="list-style-type: none"> - Select main memory boot: Bank 1 or Bank 2 - Select watchdog event: hardware or software - Reset event when entering Stop mode (STOP) - Reset event when entering Standby mode

Bit 23:20	0x0F: Unused
Bit 19	BTOPT 0: When configuring boot from main memory, boot from Bank 2. If there is no boot loader in Bank 2, boot from Bank 1. 1: When configuring boot from main memory (by default), boot from Bank 1.
Bit 18	nRST_STDBY 0: Reset is generated when entering Standby mode. 1: Reset is not generated when entering Standby mode.
Bit 17	nRST_STOP 0: Reset is generated when entering Stop mode (STOP). 1: Reset is not generated when entering Stop mode (STOP).
Bit 16	WDG_SW 0: Hardware watchdog 1: Software watchdog
WRPx: Flash memory write protection option byte For 128K Flash, every bit in the WRPx option byte is used to protect the four memory pages (1K byte/page) in the main memory. 0: Write protection is enabled. 1: Write protection is disabled. Four user bytes are used to protect the 128K main memory and SPIM. WRP0: Page 0 ~ 31 write protection (located in the register FLASH_WRPRT[7:0]) WRP1: Page 32 ~ 63 write protection (located in the register FLASH_WRPRT[15:8]) WRP2: Page 64 ~ 95 write protection (located in the register FLASH_WRPRT[23:16]) WRP3: Page 96 ~ 127 write protection (located in the register FLASH_WRPRT[31:24]); Bit 7 is for page 124 ~ 127 write protection and bank3 write protection. For 256K and above Flash, every bit in the WRPx option byte is used to protect the two memory pages (2K byte/page) in the main memory. However, Bit 7 in WRP3 is used to protect page 62 and beyond: 0: Write protection is enabled. 1: Write protection is disabled. Four user bytes are used to protect the 256K (or 512K, 1024K)main memory and SPIM. WRP0: Page 0 ~ 15 write protection (located in the register FLASH_WRPRT[7:0]) WRP1: Page 16 ~ 31 write protection (located in the register FLASH_WRPRT[15:8]) WRP2: Page 32 ~ 47 write protection (located in the register FLASH_WRPRT[23:16]) WRP3: Bit 0~6 is for page 48 ~ 61 write protection (located in the register FLASH_WRPRT[31:24]) ; Bit 7 for page 62 and beyond, as well as bank3 write protection.	
Datax: user data This address can be programmed with the programming methods of option bytes. Data7: user data 7 Data6: user data 6 Data5: user data 5 Data4: user data 4 Data3: user data 3 Data2: user data 2 Data1: user data1(stored in FLASH_UOB[25:18]) Data0: user data0(stored in FLASH_UOB[17:10])	
EOPB0: Extended option byte	
Bit 7:0	224KB_MODE 0xFE: On-chip memory is 224 KB. 0xFF: On-chip memory is 96 KB. Other configurations are reserved.
BANK3KEYx: Key value of cypher text access area in Bank 3. Setting conditions for non-encryption as follows: --Both BANK3KEYx and nBANK3KEYx are 0xFFFF (Default erase state) --Write 0x00 to BANK3KEYx That is, {nBANK3KEYx, BANK3KEYx} is set as 0xFFFF, and 0xFF00	

After every system reset, option byte loader reads the data in information block and stores the data in the register. Every option byte has its opposite bit in the information block. When loading the option bytes, the opposite bits can be used to verify whether the option bytes are correct. If there is any mismatch, an option byte error flag (UOBFLLR) will be generated. When the option byte error is generated, the corresponding option byte is forced to be 0xFF. When both the option byte and its opposite bits are 0xFF (the status after being erased), disable the above-mentioned verification function.

All the option bytes (excluding their opposites) are used to configure this microcontroller. CPU can select registers. Please refer to [Section 5.4](#) for more details.

5.4 FMC Registers

Table 5-6 Flash Memory Interface—Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																														
0x00	FLASH_ACR	Reserved																									PRFTBS	PRFTBE	HLFCYA	Reserved	Reserved																																																																																																																																																																																																																																																																																																																																																																																																
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	1	0																																																																																																																																																																																																																																																																																																																																																																																																			
0x04	FLASH_FCKEY	KEY[31:0]																																																																																																																																																																																																																																																																																																																																																																																																																													
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x																																																																																																																																																																																																																																																																																																																																																																																																
0x08	FLASH_OPTKEYR	OPTKEYR[31:0]																																																																																																																																																																																																																																																																																																																																																																																																																													
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x																																																																																																																																																																																																																																																																																																																																																																																																
0x0C	FLASH_STS	Reserved																																																																																																																																																																																																																																																																																																																																																																																																																													
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x																																																																																																																																																																																																																																																																																																																																																																																																
0x10	FLASH_CTRL	Reserved																																																																																																																																																																																																																																																																																																																																																																																																																													
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																																																																																																																																																																																	
0x14	FLASH_ADDR	TA[31:0]																																																																																																																																																																																																																																																																																																																																																																																																																													
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																																																																																																																																																																																																																																																																																																																																																																																																	
0x18		Reserved																																																																																																																																																																																																																																																																																																																																																																																																																													
0x1C	FLASH_UOB	Reserved	Data1						Data0						Unused						BTOPT	nSTDBY_RST	nSTP_RST	SYSCCLKSEL_WD	RDPRTEN	UOBFLR	BSY	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	PRGMFLR	PGERS	Reserved	PRCDN	WRPRTFLR	Reserved	

5.4.1 Flash Access Control Register (FLASH_ACR)

Address offset: 0x00

Reset value: 0x0000 0030

Bit 3~0

Reserved.

5.4.2 FPEC Key Register (FLASH_FCKEY)

Only used in Flash memory Bank 1.

Address offset: 0x04

Reset value: xxxx xxxx

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

Note: All bits are write-only, and will return 0 after being read.

Bit 31:0

KEY: FPEC key

These bits are used for FPEC unlock key.

5.4.3 Flash OPTKEY Register (FLASH_OPTKEYR)

Address offset: 0x08

Reset value: xxxx xxxx

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

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

Bit 31:0

OPTKEYR: Option bytes keyThese bits are used for option bytes key to unlock **UOBWE**.

5.4.4 Flash Status Register (FLASH_STS)

Only used in Flash memory Bank 1.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
res								rw		rw		rw		rw	
PRC DN								WRPR TFLR		Reserved		PRG MFLR		Reserved	
res								r		rw		rw		rw	

Bit 31:6	Reserved. Must be kept at reset value '0'.
Bit 5	PRCDN: Process Done When Flash memory operation (program/erase) is completed, this bit is set by hardware. This bit is cleared by writing '1'. Note: Every successful program or erase will set PRCDN status.
Bit 4	WRPRTFLR: Write protection error When programming write-protected Flash memory address, this bit is set by hardware. This bit is cleared by writing '1'.
Bit 3	Reserved. Must be kept at reset value '0'.
Bit 2	PRGMFLR: Programming error When programming address of which the content is not '0xFFFF', this bit is set by hardware. This bit is cleared by writing '1'. Note: Before programming, the RSTR bit in the FLASH_CTRL register must be cleared.
Bit 1	Reserved. Must be kept at reset value '0'.
Bit 0	BSY: Busy flag This bit indicates that Flash memory operation is in process. This bit is set when Flash memory operation starts; it is cleared when operation is completed or an error occurs.

5.4.5 Flash Control Register (FLASH_CTRL)

Only used in Flash memory Bank 1.

Address offset: 0x10

Reset value: 0x0000 0080

3	3	2	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
1	1	1	12	11	10	9	8	7	6	5	4	3	2	1	0
5	4	3													
Reserved	PRC DNI E	Reserve d	ER RI E	UO B WE	Reserve d	LOC K	RST R	UO B ERS	UOB P RGM	Reserve d	CH P ER S	PG E RS	PR G M		
res	rw	res	rw	rw	res	rw	rw	rw	rw	res	rw	rw	rw		

Bit 31:13	Reserved. Must be kept at reset value '0'.
Bit 12	PRCDNIE: Process done interrupt enable This bit enables interrupt when the PRCDN bit in the FLASH_STS register becomes '1'. 0: Interrupt is disabled; 1: Interrupt is enabled.
Bit 11, 8, 3	Reserved. Must be kept at reset value '0'.
Bit 10	ERRIE: Error status interrupt enable This bit enables interrupt when FPEC errors occur. (When the PRGMFLR/WRPRTFLR in FLASH_STS register is set.) 0: Interrupt is disabled; 1: Interrupt is enabled.
Bit 9	UOBWE: Option bytes write enable When this bit is set, programming to option bytes is enabled. This bit is set when writing correct key sequence to FLASH_OPTKEYR register. This bit can be cleared by software.

Bit 7	LOCK: Lock This bit can only be written '1'. When this bit is set, FPEC and FLASH_CTRL are locked. After correct unlock sequence is detected, this bit can be cleared by hardware. After an unsuccessful unlock operation, this bit cannot be modified until the next system reset.
Bit 6	RSTR: Start An erase operation will be triggered when this bit is set. This bit can only be set by software, and be cleared when BSY becomes '0'.
Bit 5	UOBERS: Option bytes erase Erase option bytes
Bit 4	UOBPRGM: Option bytes program Program option bytes
Bit 2	CHPERS: Mass erase Select erasing all user pages
Bit 1	PGERS: Page erase Select erasing pages
Bit 0	PRGM: Programming Select programming

5.4.6 Flash Address Register (FLASH_ADDR)

Only used in Flash memory Bank 1.

Address offset: 0x14

Reset value: 0x0000 0000

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

These bits are modified by hardware as the current/last address being used. In page erase operation, this register must be modified by software to select the pages to be erased.

Bit 31:0	TA: Flash address Select the address to be programmed in programming operation, and select the pages to be erased in page erase operation. Note: When the BSY bit in FLASH_STS is '1', this register cannot be written.
----------	--

5.4.7 Option Byte Register (FLASH_UOB)

Address offset: 0x1C

Reset value: 0x03FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						USER_D1[7:0]						USER_D0[7:6]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USER_D0[5:0]						Unused			BTO_PT	nSTDBY_RST	nSTP_RST	SYSCL_KSEL_WD	RDPR_TEN	UOB_FLR	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:26		Reserved. Must be kept at reset value '0'.													

Bit 25:18	USER_D1: User data 1
Bit 17:10	USER_D0: User data 0
Bit 9:2	USR: User option bytes, including user option bytes loaded with OBL Bit [9: 6]: Unused Bit 5: BTOPT Bit 4: nSTDBY_RST Bit 3: nSTP_RST Bit 2: WDG_SW
Bit 1	RDPRTEN: When read protection is '1', read protection to the Flash memory is valid. Note: This bit is read-only.
Bit 0	UOBFLR: Option bytes error When option bytes and their opposites are mismatched, this bit is '1'. Note: This bit is read-only.

5.4.8 Write Protection Register (FLASH_WRPRT)

Address offset: 0x20

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRPRTBMP [31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRPRTBMP [15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:0		WRPRTBMP: Write protection This register includes write protection option bytes loaded with OBL. 0: Write protection is enabled. 1: Write protection is disabled. Note: These bits are read-only.													

5.4.9 FPEC Key Register 2 (FLASH_FCKEY2)

Only used in Flash memory Bank 2.

Address offset: 0x44

Reset value: xxxx xxxx

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

Note: All bits are write-only, and will return 0 after being read.

Bit 31:0	KEY: FPEC key These bits are used for FPEC unlock key in Bank 2.
----------	---

5.4.10 Flash Status Register 2 (FLASH_STS2)

Only used in Flash memory Bank 2.

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
res															
Bit 31:6		Reserved. Must be kept at reset value '0'.													
Bit 5		PRCDN: Process Done When Flash memory operation (program/erase) is completed, this bit is set by hardware. This bit is cleared by writing '1'. Note: Every successful program or erase will set PRCDN status.													
Bit 4		WRPRTFLR: Write protection error When programming write-protected Flash memory address, this bit is set by hardware. This bit is cleared by writing '1'.													
Bit 3		Reserved. Must be kept at reset value '0'.													
Bit 2		PRGMFLR: Programming error When programming address of which the content is not '0xFFFF', this bit is set by hardware. This bit is cleared by writing '1'. Note: Before programming, the RSTR bit in FLASH_CTRL2 register must be cleared.													
Bit 1		Reserved. Must be kept at reset value '0'.													
Bit 0		BSY: Busy This bit indicates that Flash memory operation is in process. This bit is set when Flash memory operation starts; it is cleared when operation is completed or an error occurs.													

5.4.11 Flash Control Register 2 (FLASH_CTRL2)

Only used in Flash memory Bank 2.

Address offset: 0x50

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:13		Reserved. Must be kept at reset value '0'.													
Bit 12		PRCDNIE: Process done interrupt enable This bit enables interrupt when the PRCDN bit in FLASH_STS2 register becomes '1'. 0: Interrupt is disabled; 1: Interrupt is enabled.													
Bit 11,9,8		Reserved. Must be kept at reset value '0'.													

Bit 10	ERRIE: Error status interrupt enable This bit enables interrupt when FPEC errors occur. (When the PRGMFLR/WRPRTFLR bit in FLASH_STS2 register is set.) 0: Interrupt is disabled; 1: Interrupt is enabled.
Bit 7	LOCK: Lock This bit can only be written '1'. When this bit is set, FPEC and FLASH_CTRL2 are locked. After correct unlock sequence is detected, this bit can be cleared by hardware. After an unsuccessful unlock operation, this bit cannot be modified until the next system reset.
Bit 6	RSTR: Start An erase operation will be triggered when this bit is set. This bit can only be set by software, and be cleared when BSY becomes '0'.
Bit 5:3	Reserved. Must be kept at reset value '0'.
Bit 2	CHPERS: Mass erase Select erasing all user pages.
Bit 1	PGERS: Page erase Select erasing pages.
Bit 0	PRGM: Programming Select programming.

5.4.12 Flash Address Register 2 (FLASH_ADDR2)

Only used in Flash memory Bank 2.

Address offset: 0x54

Reset value: 0x0000 0000

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

These bits are modified by hardware as the current/last address being used. In page erase operation, this register must be modified by software to select the pages to be erased.

Bit 31:0	TA: Flash address Select the address to be programmed in programming operation, and select the pages to be erased in page erase operation. Note: When the BSY bit in FLASH_STS2 is '1', this register cannot be written.
----------	---

5.4.13 FPEC Key Register 3 (FLASH_FCKEY3)

Only used in Flash memory Bank 3.

Address offset: 0x84

Reset value: xxxx xxxx

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Note: All bits are write-only, and will return 0 after being read.

Bit 31:0

KEY: FPEC key

These bits are used for **FPEC** unlock key in Bank 3.

5.4.14 Flash Option Byte Register (FLASH_SELECT)

Only used in Flash memory Bank 3.

Address offset: 0x88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SELECT[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

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

Bit 31:0

SELECT[31:0] Bank 3 supports extended SPI Flash chip models selection
0x0001: GD25Q127C, GD25Q64C, GD25Q32C, GD25Q16C, GD25Q80C

0x0002: EN25F20A, EN25QH128A, W25Q128V

Others: Reserved

Note: In addition to the above SPI Flash models, SPI Flash chips compatible with the above models' instruction sets are also supported.

5.4.15 Flash Status Register 3 (FLASH_STS3)

Only used in Flash memory Bank 3.

Address offset: 0x8C

Reset value: 0x0000 0000

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

Bit 31:6

Reserved. Must be kept at reset value '0'.

Bit 5

PRCDN: Process Done

When Flash memory operation (program/erase) is completed, this bit is set by hardware. This bit is cleared by writing '1'.

Note: Every successful program or erase will set **PRCDN** status.

Bit 4

WRPRTFLR: Write protection error

When programming write-protected Flash memory address, this bit is set by hardware. This bit is cleared by writing '1'.

Bit 3

Reserved. Must be kept at reset value '0'.

Bit 2	PRGMFLR: Programming error When programming address of which the content is not '0xFFFF', this bit is set by hardware. This bit is cleared by writing '1'. Note: Before programming, the RSTR bit in FLASH_CTRL3 register must be cleared.
Bit 1	Reserved. Must be kept at reset value '0'.
Bit 0	BSY: Busy This bit indicates that Flash memory operation is in process. This bit is set when Flash memory operation starts; it is cleared when operation is completed or an error occurs.

5.4.16 Flash Control Register 3 (FLASH_CTRL3)

Only used in Flash memory Bank 3.

Address offset: 0x90

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PRC DNIE	Reserved	ER RIE	Reserved	LO CK	RS TR	Reserved	Reserved	Reserved	CHP ERS	PGE RS	PR GM	res	rw	res

Bit 31:13	Reserved. Must be kept at reset value '0'.
Bit 12	PRCDNIE: Process done interrupt enable This bit enables interrupt when the PRCDN bit in FLASH_STS3 register becomes '1'. 0: Interrupt is disabled; 1: Interrupt is enabled.
Bit 11,9,8	Reserved. Must be kept at reset value '0'.
Bit 10	ERRIE: Error status interrupt enable This bit enables interrupt when FPEC errors occur. (When the PRGMFLR/WRPRTFLR bit in FLASH_STS3 register is set.) 0: Interrupt is disabled; 1: Interrupt is enabled.
Bit 7	LOCK: Lock This bit can only be written '1'. When this bit is set, FPEC and FLASH_CTRL3 are locked. After correct unlock sequence is detected, this bit can be cleared by hardware. After an unsuccessful unlock operation, this bit cannot be modified until the next system reset.
Bit 6	RSTR: Start An erase operation will be triggered when this bit is set. This bit can only be set by software, and be cleared when BSY becomes '0'.
Bit 5:3	Reserved. Must be kept at reset value '0'.
Bit 2	CHPERS: Mass erase Select erasing all user pages
Bit 1	PGEERS: Page erase Select erasing pages
Bit 0	PRGM: Programming Select programming

5.4.17 Flash Address Register 3 (FLASH_ADDR3)

Only used in Flash memory Bank 3.

Address offset: 0x94

Reset value: 0x0000 0000

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

These bits are modified by hardware as the current/last address being used. In page erase operation, this register must be modified by software to select the pages to be erased.

Bit 31:0	TA: Flash address Select the address to be programmed in programming operation, and select the pages to be erased in page erase operation. Note: When the BSY bit in FLASH_STS3 is '1', this register cannot be written.
----------	---

5.4.18 Flash Decode Address Register (FLASH_DA)

Only used in Flash memory Bank 3.

Address offset: 0x98

Reset value: 0x0000 0000

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

Bit 31:0	FDA: Flash decode address When programming the external Flash memory, encryption programming will not be executed when the written address is higher than this temporary value. Note: When the BSY bit in FLASH_STS3 is '1', this register cannot be written.
----------	--

6 CRC Calculation Unit (CRC)

6.1 CRC Introduction

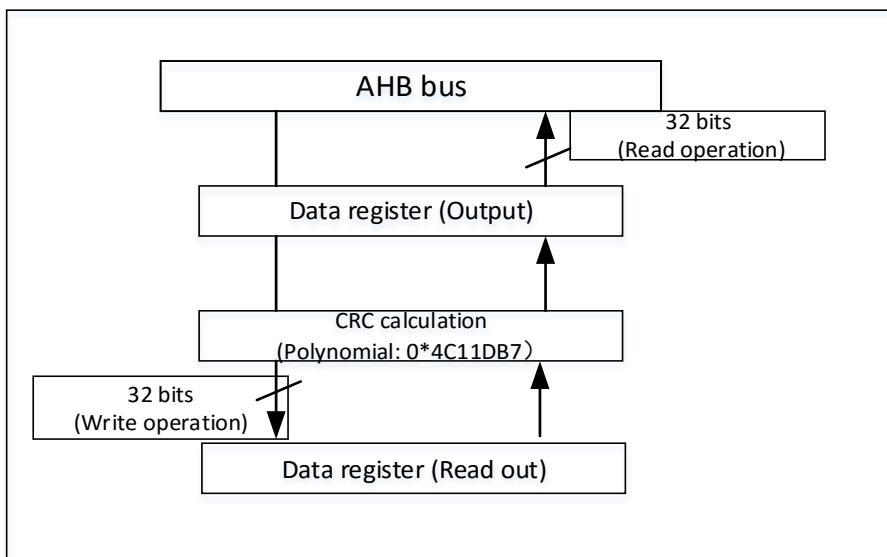
The Cyclic Redundancy Check (CRC) calculation unit is used to calculate a 32-bit CRC code based on a fixed generator polynomial. In other applications, CRC-based techniques are applied to verify the correctness and integrity of data transmission or data storage. EN/IEC 60335-1 standard provides a method to verify the Flash memory integrity. The CRC calculation unit can compute a signature of the software during runtime, compare it with a reference signature generated at linktime, and finally store it at a given memory location.

6.2 CRC Main Features

- Use CRC-32 (Ethernet) polynomial: 0x4C11DB7
 - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^4 + X^2 + X + 1$
- A 32-bit data register for input/output
- CRC calculation period: 4 AHB clock cycles (HCLK)
- General-purpose 8-bit register (Can be used for temporary storage)

Figure 6-1 is the block diagram of CRC calculation unit.

Figure 6-1 Block Diagram of CRC Calculation Unit



6.3 CRC Function Overview

CRC calculation unit includes one 32-bit data register:

- When being written, this register is used as input register, and can receive new data to be calculated with CRC.
- When being read, it returns the last CRC calculation result. Every write operation into the data register creates a combination of the last CRC calculation result and the new one (CRC calculation is done to the whole 32-bit word, instead of byte by byte).

CPU write operation will be suspended during CRC calculation. Hence, back-to-back write or consecutive write-read operation can be done to the CRC_DR register.

The CRC_DR register can be reset to 0xFFFF FFFF with the RESET bit in the CRC_CTRL register. This operation does not affect the data in the CRC_IDR register

6.4 CRC Registers

CRC calculation unit includes two data registers and one control register.

Table 6-1 lists CRC register map and reset values.

Table 6-1 CRC Calculation Unit Register Map

Offset	Register	31~24	23~16	15~8	7	6	5	4	3	2	1	0	
0x00	CRC_DR	Data Register 0xFFFF FFFF											
	Reset Value												
0x04	CRC_IDR	Reserved	Independent Data Register 0x00										
	Reset Value												
0x08	CRC_CTRL	Reserved	RESET 0										
	Reset Value												

6.4.1 Data Register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[31:16]															
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
rw															
Bit 31:0	Data register bits Used as input register when writing new data of CRC calculator. Return CRC calculation results when it is read.														

6.4.2 Independent Data Register (CRC_IDR)

Address offset: 0x04

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved																							
res																							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Reserved								IDR[7:0]															
res																							
Bit 31:8	Reserved																						
Bit 7:0	General-purpose 8-bit data register bits These bits can be used for temporary storage of 1-byte data. CRC reset generated by the RESET bit in the CRC_CTRL register will not affect this register.																						

6.4.3 Control Register (CRC_CTRL)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved																							
res																							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Reserved								RESET															
res																							
Bit 31:8	Reserved																						
Bit 7:0	RESET bit Reset CRC calculation unit, set data register as 0xFFFF FFFF. This bit can only be written '1', and is cleared by hardware automatically.																						

7 General-purpose and Alternate-function I/Os (GPIOs and AFIOs)

7.1 Introduction

GPIO interface includes 7 sets of general-purpose I/O ports.

Each GPIO set provides 16 general-purpose I/O pins. Each GPIO port has relevant control and configuration registers to fulfill specific functions. External interrupts on the GPIO pins also have relevant control and configuration registers in external interrupt controller. Please refer to [Chapter 8](#).

GPIO ports are pin-shared with other alternate functions to obtain maximum flexibility on the given package. GPIO pins can be used as alternate functional pins by configuring relevant registers as the alternate function input and output.

Each GPIO pin can be configured by software as output (push-pull or open-drain), input (pull-up, pull-down, or non pull-up/ pull-down), or peripheral alternate function. Most GPIO pins provide digital or analog alternate functions. All GPIOs are high-current capable.

7.2 Main Features

- Input/output direction control
- Each pin has weak pull-up/pull-down function.
- Output push-pull/open drain enable control
- Output set/reset control
- External interrupt with programmable trigger edge – in external interrupt configuration registers
- Analog input/output configuration
- Alternate function input/output configuration
- Port configuration lock

7.3 Function Overview

7.3.1 GPIO Pin Configuration

Each GPIO port has two 32-bit configuration registers (GPIOx_CTRLR, GPIOx_CTRLH), two 32-bit data registers (GPIOx_IPTDT, GPIOx_OPTDT), a 32-bit set/reset register (GPIOx_BSRE), a 16-bit reset register (GPIOx_BRE), and a 32-bit locking register (GPIOx_LOCK).

According to the specific hardware characteristics of each I/O port listed in the data sheet, each bit of GPIO interface can be configured into several modes by software.

- Input floating
- Input pull-up
- Input pull-down
- Analog
- Output open-drain
- Output push-pull
- Alternate function push-pull
- Alternate function open-drain

Each I/O port bit can be programmed flexibly. However, I/O port registers must be accessed by 32-bit words (half-word and byte access are not allowed). The GPIOx_BSRE and GPIOx_BRE registers allow independent read/modify access to any GPIO registers. In this way, IRQ generated between read and modify access will not cause risks.

Figure 7-1 shows a basic structure of an I/O port bit.

Figure 7-1 Basic Structure of an I/O Port Bit

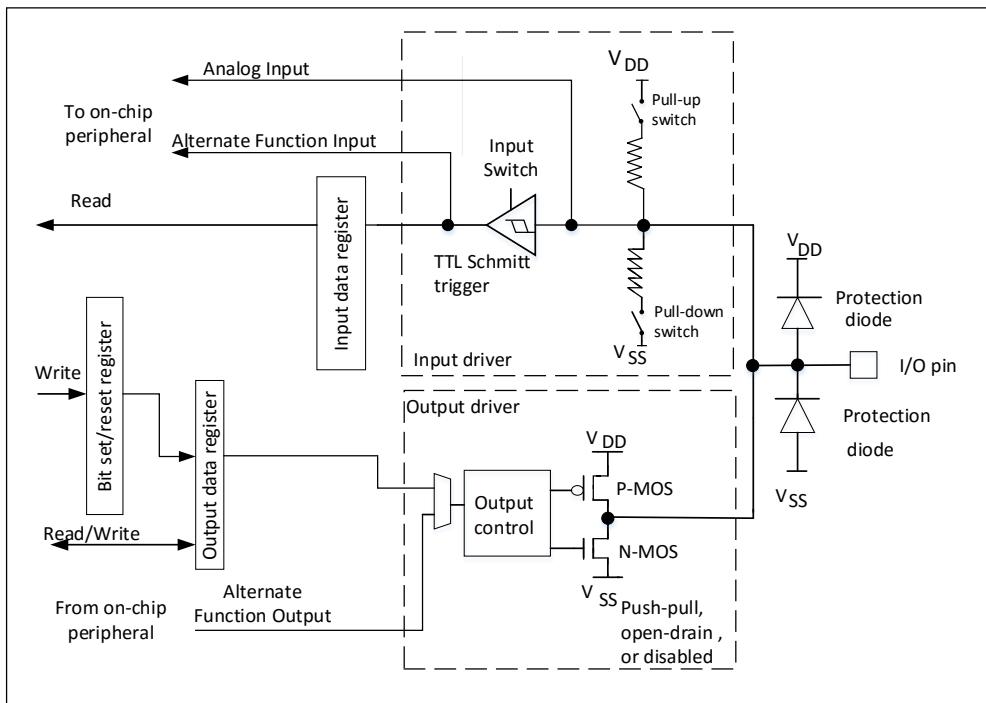
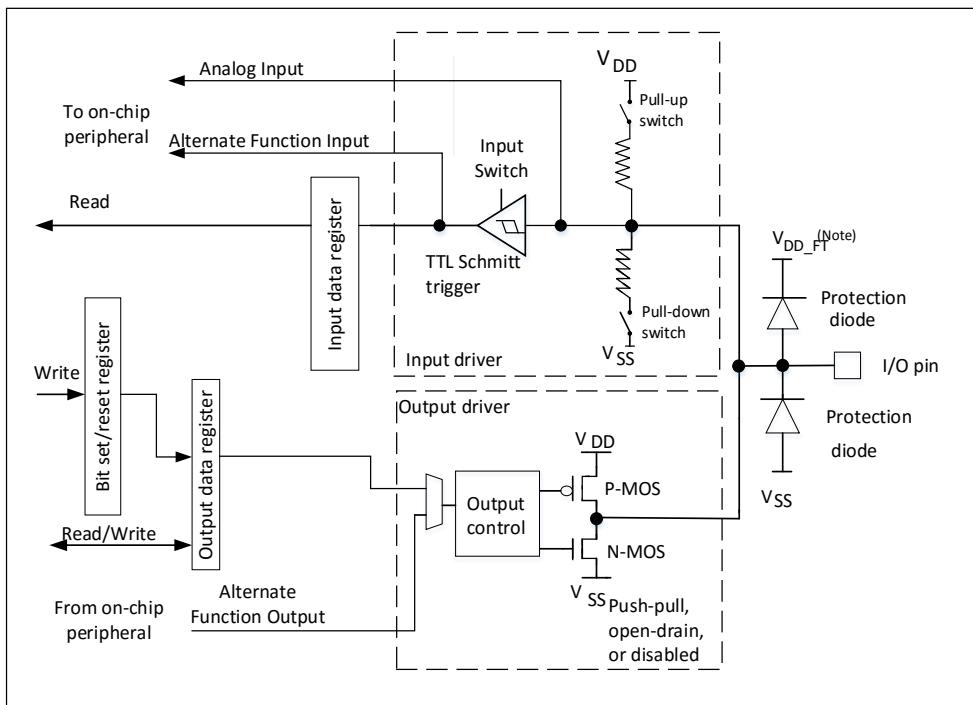


Figure 7-2 Basic Structure of a 5-V Tolerant I/O Port Bit



Note: V_{DD_FT} is specific to 5-V tolerant I/Os, and is different from V_{DD} .

Table 7-1 Port Bit Assignment

Configuration Mode		CONF1	CONF0	MDE1	MDE0	PxOPTDT Register
General-purpose Output	Push-Pull	0	0	01 10 11	01 10 11	0 or 1
	Open-Drain		1			0 or 1
Alternate Function Output	Push-Pull	1	0			Unused
	Open-Drain		1			Unused
Input	Analog	0	0	00	00	Unused
	Floating		1			Unused
	Pull-Down	1				0
	Pull-Up		0			1

During and after reset, alternate function will not be enabled. I/O interface is configured in input floating mode (CONFx[1:0] = 01b, MDEx[1:0] = 00b).

After reset, JTAG pin is set as input pull-up or pull-down mode:

- PA15: JTDI in pull-up mode
- PA14: JTCK in pull-down mode
- PA13: JTMS in pull-up mode
- PB4: JNTRST in pull-down mode

When configured as output, the value written to the output data register (GPIOx_OPTDT) will be output to the corresponding I/O pin. Output driver can be used in push-pull mode or open-drain mode (when output is 0, only the N-MOS is activated).

Input data register (GPIOx_IPTDT) collects the data on the I/O pin at every APB2 clock cycle. All GPIO pins have an internal weak pull-up and weak pull-down, which can be activated and stopped when configured as input.

When programming to individual bit in GPIOx_OPTDT, software does not have to disable interrupts: in a single APB2 write operation, it is possible to modify only one bit or several bits. This can be achieved by writing '1' to the bit to be modified in set/reset register (GPIOx_BSRE, reset: GPIOx_BRE). Unselected bits will not be modified.

7.3.2 External Interrupt/Wakeup Lines

All ports have external interrupt capability. To use external interrupt lines, the ports must be configured in input mode. For more information on external interrupt, please refer to [Section 8.2](#).

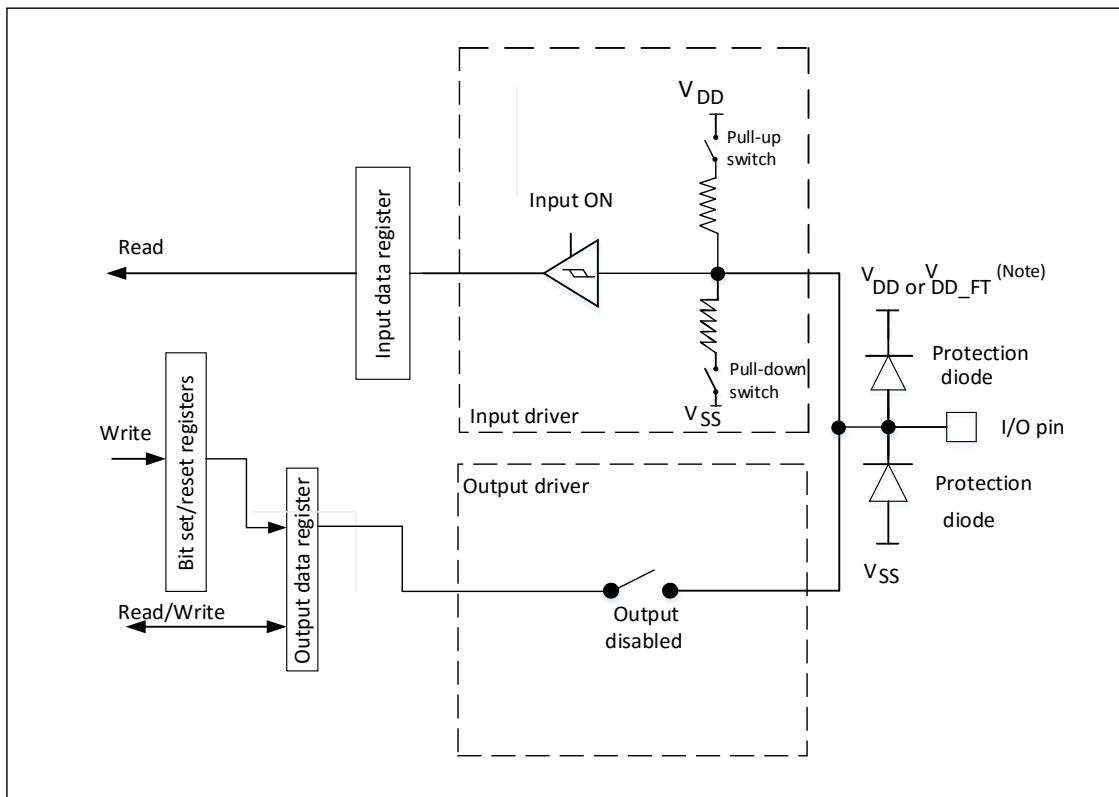
7.3.3 Input Configuration

When I/O port is configured as input:

- Output buffer is disabled.
- Schmitt-trigger input is activated.
- Weak pull-up and pull-down resistors are connected depending on input configuration (pull-up, pull-down or floating).
- Data present on the I/O pin is sampled into the input data register every APB2 clock cycle.
- Read access to input data register can obtain I/O states.

Figure 7-3 shows the input configuration of the I/O port bit.

Figure 7-3 Input Floating/Pull-up/Pull-down Configuration



Note: V_{DD_FT} is specific to 5-V tolerant I/Os, and is different from V_{DD} .

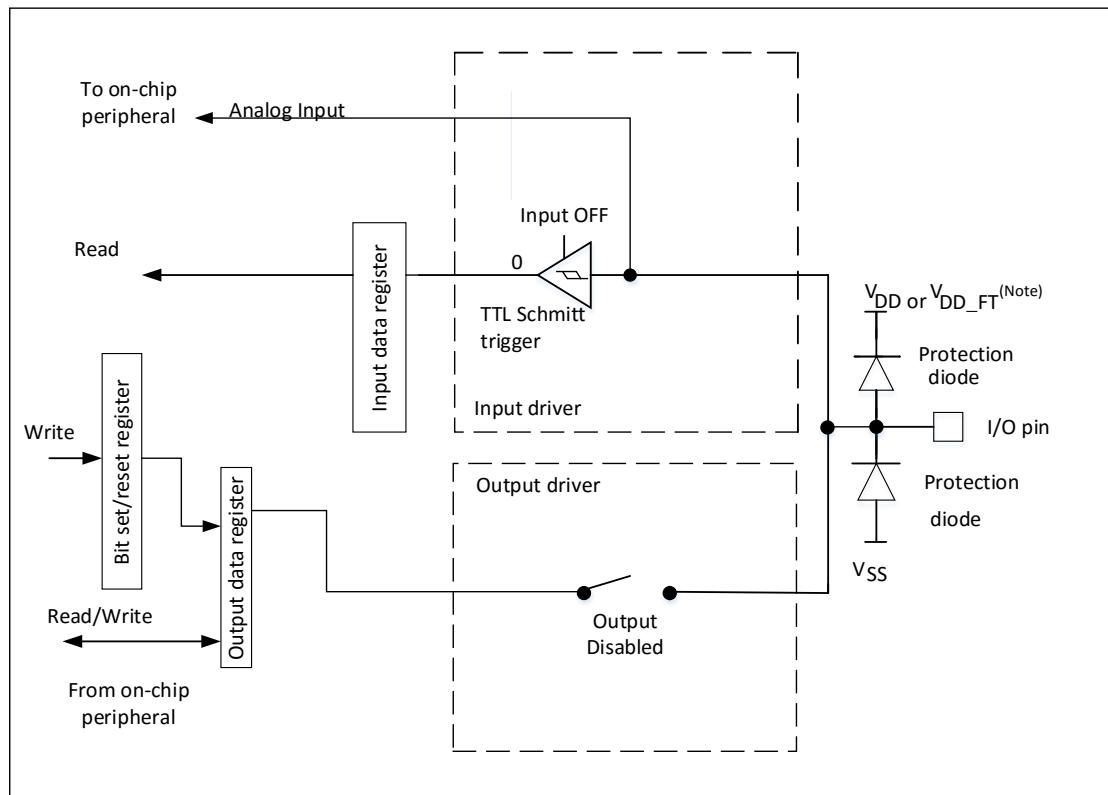
7.3.4 Analog Input Configuration

When I/O port is configured as analog input:

- Output buffer is disabled.
- Schmitt-trigger input is disabled, achieving zero consumption for every analog I/O pin. The output value of Schmitt-trigger is forced to be '0'.
- Weak pull-up and pull-down resistors are disabled.
- Input data register value is '0' when being read.

Figure 7-4 shows the high impedance analog input configuration of I/O port bit:

Figure 7-4 High Impedance Analog Input Configuration



Note: V_{DD_FT} is specific to 5-V tolerant I/Os, and is different from V_{DD} .

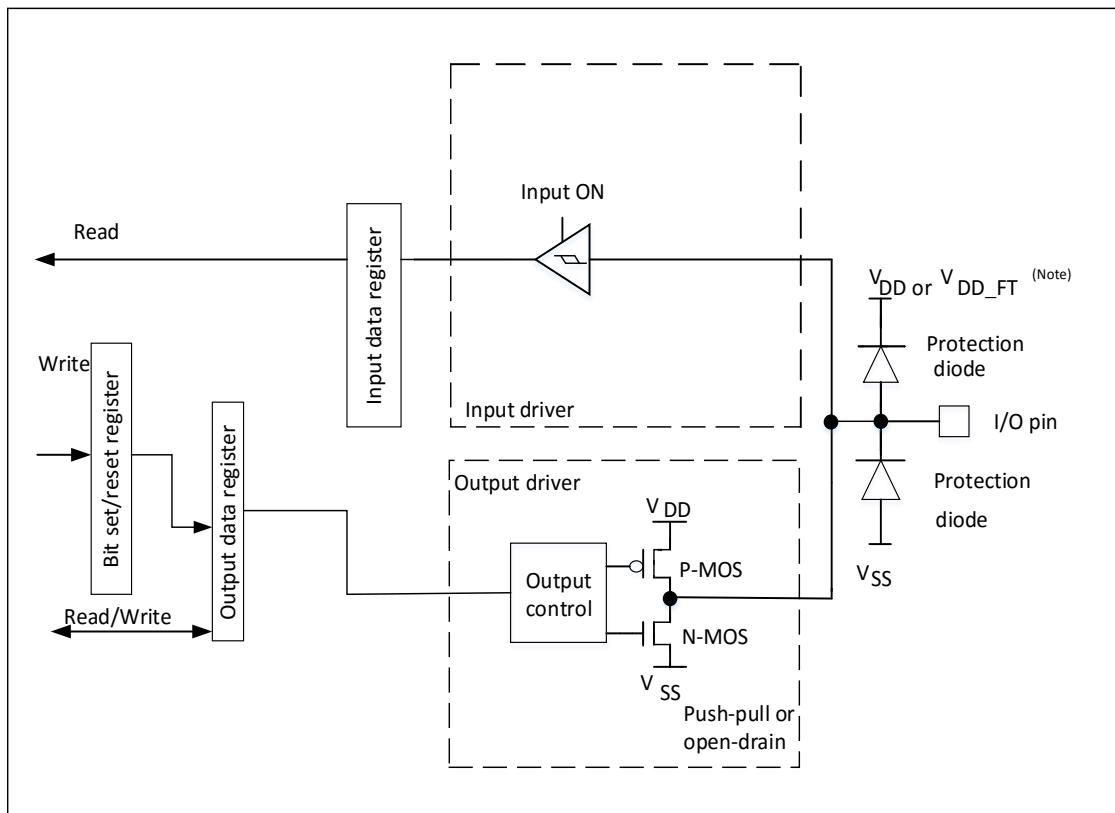
7.3.5 Output Configuration

When I/O port is configured as output:

- Output buffer is enabled.
 - Open-drain mode: A ‘0’ in the output register activates N-MOS, while a ‘1’ in the output register leaves the port in HiZ (the P-MOS is never activated).
 - Push-pull mode: A ‘0’ in the output register activates N-MOS, while a ‘1’ in the output register activates P-MOS.
- Schmitt-trigger input is enabled.
- Weak pull-up and pull-down resistors are disabled.
- Data present on the I/O pin is sampled into the input data register every APB2 clock cycle.
- In open-drain mode, read access to input data register can obtain I/O states.
- In push-pull mode, read access to output data register can obtain the last written value.

Figure 7-5 shows the output configuration of I/O port bit.

Figure 7-5 Output Configuration



Note: V_{DD_FT} is specific to 5-V tolerant I/Os, and is different from V_{DD} .

7.3.6 GPIO Locking Mechanism

Locking mechanism can freeze the I/O configuration. When LOCK is applied to a port bit, its configuration cannot be modified until the next reset.

7.3.7 Alternate Function (AF)

Before using the default alternate function, the port bit configuration register must be programmed.

- For alternate input function, the port must be configured in input mode (floating, pull-up, or pull-down), and the input pin must be driven externally.

Note: *Alternate function input pin can also be simulated by software. This kind of simulation can be achieved by programming to GPIO controller. In this case, a port should be set as alternate function output mode. Apparently, the corresponding pins are no longer driven externally. Instead, they are driven by software through GPIO controller.*

- For alternate output function, the port must be configured in alternate function output mode (push-pull or open-drain).
- For bidirectional alternate function, the port bit must be configured in alternate function output mode (push-pull or open-drain). At this time, input driver is configured in floating input mode.

If the port is configured as alternate output function, pins are disconnected to output register, and connected to output signal of on-chip peripherals. If a GPIO pin is configured as alternate output function by software without activating the peripherals, its output will not be specified.

When I/O ports are configured as alternate function:

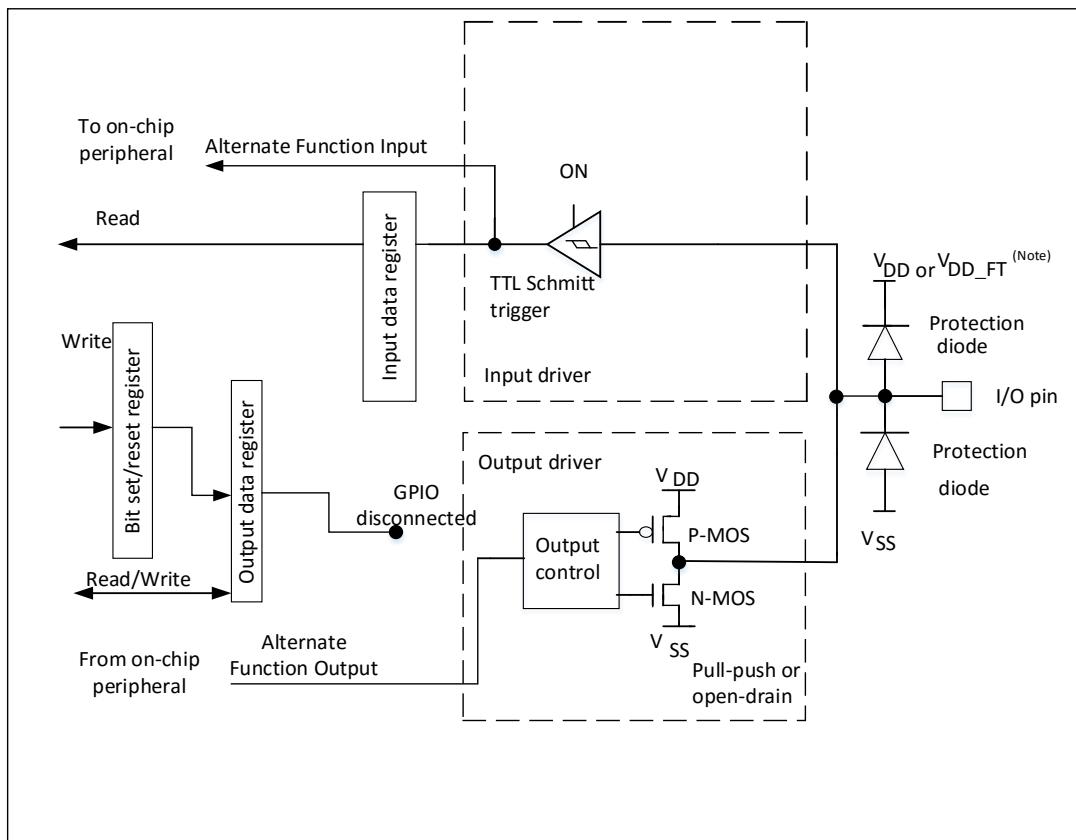
- Output buffer is turned ON in open-drain or push-pull configuration.
- Output buffer is driven by embedded peripheral signals (alternate function output).
- Schmitt-trigger input is activated.
- Weak pull-up and pull-down resistors are disabled.

- If pins are configured as multiple-function output, please refer to the data sheet for the priority of each alternate function.
- Data present on the I/O pin is sampled into the input data register every APB2 clock cycle.
- In open-drain mode, read access to input data register can obtain I/O states.
- In push-pull mode, read access to output data register can obtain the last written value.

Figure 7-6 shows the alternate function configuration of I/O port bit. Please refer to [Section 7.4](#) for more details.

A set of alternate function I/O registers allows users to remap some alternate functions to different pins.

Figure 7-6 Alternate Function Configuration



Note: V_{DD_FT} is specific to 5-V tolerant I/Os, and is different from V_{DD} .

To optimize the number of I/O functions of different device packages, some alternate functions can be remapped to other pins. This can be done by configuring the corresponding registers with software (Please refer to [Section 7.5](#)). In this case, alternate functions no longer map to their original pins.

Table 7-2 lists the pin configuration for each peripheral.

Table 7-2 Advanced Timers TMR1/8/15

TMR1/8/15 Pin	Configuration	GPIO Configuration
TMR1/8/15_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TMR1/8/15_CHxN	Complementary output channel x	Alternate function push-pull
TMR1/8/15_BKIN	Break input	Input floating
TMR1/8/15_ETR	External trigger timer input	Input floating

Table 7-3 General-purpose Timer TMR2-5/TMR9-14

TMRx Pin	Configuration	GPIO Configuration
TMRx_CHx	Input capture channel x	Input floating
	Output compare channel x	Alternate function push-pull
TMRx_ETR	External trigger timer input	Input floating

Table 7-4 USART

USART Pin	Configuration	GPIO Configuration
USARTx_TX	Full duplex	Alternate function push-pull
	Half duplex synchronous mode	Alternate function push-pull
USARTx_RX	Full duplex	Input floating/Input pull-up
	Half duplex synchronous mode	Unused. Can be used as a GPIO.
USARTx_CK	Synchronous mode	Alternate function push-pull
USARTx_RTS	Hardware flow control	Alternate function push-pull
USARTx_CTS	Hardware flow control	Input floating/Input pull-up

Table 7-5 SPI

SPI Pin	Configuration	GPIO Configuration
SPIx_SCK	Master	Alternate function push-pull
	Slave	Input floating
SPIx_MOSI	Full duplex/Master	Alternate function push-pull
	Full duplex/Slave	Input floating/Input pull-up
	Simplex bidirectional data wire/Master	Alternate function push-pull
	Simplex bidirectional data wire/Slave	Unused. Can be used as a GPIO.
SPIx_MISO	Full duplex/Master	Input floating/Input pull-up
	Full duplex/Slave	Alternate function push-pull
	Simplex bidirectional data wire/Master	Unused. Can be used as a GPIO.
	Simplex bidirectional data wire/Slave	Alternate function push-pull
SPIx_NSS	Hardware Master/Slave	Input floating/Input pull-up/Input pull-down
	Hardware Master/NSS output enabled	Alternate function push-pull
	Software	Unused. Can be used as a GPIO.

Table 7-6 I²S

I ² S Pin	Configuration	GPIO Configuration
I2Sx_WS	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_CK	Master	Alternate function push-pull
	Slave	Input floating
I2Sx_SD	Transmitter	Alternate function push-pull
	Receiver	Input floating/Input pull-up /Input pull-down
I2Sx_MCLK	Master	Alternate function push-pull
	Slave	Unused. Can be used as a GPIO.

Table 7-7 I²C Interface

I ² C Pin	Configuration	GPIO Configuration
I2Cx_SCL	I ² C clock	Alternate function open-drain
I2Cx_SDA	I ² C data	Alternate function open-drain

Table 7-8 BxCAN

BxCAN Pin	GPIO Configuration
CAN_TX	Alternate function push-pull
CAN_RX	Input floating/Input pull-up

Table 7-9 USB ^(Note)

USB Pin	GPIO Configuration
USB_DM/USB_DP	Once the USB block is enabled, these pins are automatically connected to the USB internal transceiver.

Table 7-10 SDIO

SDIO Pin	GPIO Configuration
SDIO_CK	Alternate function push-pull
SDIO_CMD	Alternate function push-pull
SDIO[D7:D0]	Alternate function push-pull

Table 7-11 ADC/DAC

ADC/DAC Pin	GPIO Configuration
ADC/DAC	Analog input

Note: ADC input pin must be configured as analog input.

Table 7-12 XMC

XMC Pin	GPIO Configuration
XMC_A[25:0] XMC_D[15:0]	Alternate function push-pull
XMC_CK	Alternate function push-pull
XMC_NOE XMC_NWE	Alternate function push-pull
XMC_NE[4:1] XMC_NCE[3:2] XMC_NCE4_1 XMC_NCE4_2	Alternate function push-pull
XMC_NWAIT XMC_CD	Input floating/Input pull-up
XMC_NIOS16 XMC_INTR XMC_INT[3:2]	Input floating
XMC_NL XMC_NBL[1:0]	Alternate function push-pull
XMC_NIORD XMC_NIOWR XMC_NREG	Alternate function push-pull

Table 7-13 Other I/O Functions

Pin	Alternate Function	GPIO Configuration
TAMPER-RTC	RTC output	Forced by hardware when configuring the BRKP_CTRL and BRKP_RTCCAL registers.
	Tamper event input	
CLKOUT	Clock output	Alternate function push-pull
EXTINT input lines	External input interrupts	Input floating/Input pull-up/Input pull-down

7.4 IO Mapping Function Configuration

To optimize the number of peripherals on 64-pin or 100-pin packages, some alternate functions can be remapped on other pins. Pin remapping can be achieved by setting the AF remap and debug I/O configuration register (AFIO_MAP). In this case, alternate functions no longer map to their original assignation.

7.4.1 OSC32_IN/OSC32_OUT as GPIO Interface PC14/PC15

When LSE oscillator is OFF, LSE oscillator pins, OSC32_IN/OSC32_OUT, can be used as the PC14/PC15 of GPIO. LSE function always has higher priority than general-purpose I/O port.

Note: Please refer to [Section 2.3.1.2](#) for the restriction of I/O port use.

7.4.2 OSC_IN/OSC_OUT Pin as GPIO Interface PD0/PD1

External oscillator pin, OSC_IN/OSC_OUT, can be used as PD0/PD1 of GPIO. This is achieved by configuring the AF remap and debug I/O configuration register (AFIO_MAP).

This remapping only applies to 48-pin and 64-pin packages (100-pin and 144-pin packages have independent PD0 and PD1 pins, so remapping is not needed).

Note: External interrupts/events function is not remapped. On 48-pin and 64-pin packages, PD0 and PD1 cannot be used to generate external interrupts/events.

7.4.3 CAN Alternate Function Remapping

CAN signal can be mapped to port A, port B, or port D, as shown in Table 7-14. For port D, there is no remapping function in 48-pin and 64-pin packages.

Table 7-14 CA1 Alternate Function Remapping

Alternate Function	CAN_REMAP[1:0] = '00'	CAN_REMAP[1:0] = '10'	CAN_REMAP[1:0] = '11'
CAN_RX	PA11	PB8	PD0
CAN_TX	PA12	PB9	PD1

Note: When PD0 and PD1 are not remapped on OSC_IN and OSC_OUT, remapping function only applies to 100-pin and 144-pin packages.

7.4.4 JTAG/SWD Alternate Function Remapping

Debug interface signal is mapped to GPIO port, as shown in Table 7-15.

Table 7-15 Debug Interface Signal

Alternate Function	GPIO Port
JTMS/SWDIO	PA13
JTCK/SWCLK	PA14
JTDI	PA15
JTDO/TRACESWO	PB3
JNTRST	PB4
TRACECK	PE2
TRACED0	PE3
TRACED1	PE4
TRACED2	PE5
TRACED3	PE6

To use more GPIOs during debugging, configuring the SWJTAG_CONF[2:0] bits in the AF remap and debug I/O configuration register (AFIO_MAP) can modify the above-mentioned remapping configuration.

Table 7-16 Debug Port Mapping

SWJTAG_CONF [2:0]	Possible Debug Port	SWJ/I/O Pin Assignment				
		PA13/JTMS/SWDIO	PA14/JTCK/SWCLK	PA15/JTDI	PB3/JTDO/TRACESWO	PB4/NJTRST
000	Full SWJ (JTAG-DP+SW-DP) (reset status)	I/O not available	I/O not available	I/O not available	I/O not available	I/O not available
001	Full SWJ (JTAG-DP+SW-DP) Without JNTRST	I/O not available	I/O not available	I/O not available	I/O not available	I/O available
010	Disable JTAG-DP, Enable SW-DP	I/O not available	I/O not available	I/O available	I/O available (Note)	I/O available
100	Disable JTAG-DP, Disable SW-DP	I/O available	I/O available	I/O available	I/O available	I/O available
Other	Forbidden	-	-	-	-	-

Note: I/O port can be used only when asynchronous trace is not used.

7.4.5 ADC Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register (AFIO_MAP).

Table 7-17 ADC1 External Trigger Injected Conversion Alternate Function Remapping

Alternate Function	ADC1_EXTRGINJ_REMAP = 0	ADC1_EXTRGINJ_REMAP = 1
ADC1 External Trigger Injected Conversion	ADC1 External Trigger Injected Conversion is connected to EXTINT15	ADC1 External Trigger Injected Conversion is connected to TMR8_CH4

Table 7-18 ADC1 External Trigger Regular Conversion Alternate Function Remapping

Alternate Function	ADC1_EXTRGREG_REMAP = 0	ADC1_EXTRGREG_REMAP = 1
ADC1 External Trigger Regular Conversion	ADC1 External Trigger Regular Conversion is connected to EXTINT11	ADC1 External Trigger Regular Conversion is connected to TMR8_TRGO

Table 7-19 ADC2 External Trigger Injected Conversion Alternate Function Remapping

Alternate Function	ADC2_EXTRGINJ_REMAP = 0	ADC2_EXTRGINJ_REMAP = 1
ADC2 External Trigger Injected Conversion	ADC2 External Trigger Injected Conversion is connected to EXTINT15	ADC2 External Trigger Injected Conversion is connected to TMR8_CH4

Table 7-20 ADC2 External Trigger Regular Conversion Alternate Function Remapping

Alternate Function	ADC1_EXTRGREG_REMAP = 0	ADC2_EXTRGREG_REMAP = 1
ADC2 External Trigger Regular Conversion	ADC2 External Trigger Regular Conversion is connected to EXTINT11	ADC2 External Trigger Regular Conversion is connected to TMR8_TRGO

7.4.6 Timer Alternate Function Remapping

Remapping of other timers are listed in Table 7-21 ~ 31.

Please refer to the AF remap and debug I/O configuration register (AFIO_MAP).

Table 7-21 TMR5 Alternate Function Remapping

Alternate Function	TMR5CH4_INTLRE = 0	TMR5CH4_INTLRE = 1
TMR5_CH4	TMR5 channel 4 is connected to PA3	LSI internal clock is connected to TIM5_CH4 input for calibration purpose.

Table 7-22 TMR4 Alternate Function Remapping

Alternate Function	TMR4_REMAP = 0	TMR4_REMAP = 1 <small>(Note)</small>
TMR4_CH1	PB6	PD12
TMR4_CH2	PB7	PD13
TMR4_CH3	PB8	PD14
TMR4_CH4	PB9	PD15

Note: Remapping only applies to 100-pin and 144-pin packages.

Table 7-23 TMR3 Alternate Function Remapping

Alternate Function	TMR3_REMAP[1:0] = 00 (No remapping)	TMR3_REMAP[1:0] = 10 (Partial remapping)	TMR3_REMAP[1:0] = 11 (Full remapping) <small>(Note)</small>
TMR3_CH1	PA6	PB4	PC6
TMR3_CH2	PA7	PB5	PC7
TMR3_CH3	PB0		PC8
TMR3_CH4	PB1		PC9

Note: Remapping only applies to 64-pin, 100-pin, and 144-pin packages.

Table 7-24 TMR2 Alternate Function Remapping

Alternate Function	TMR2_REMAP[1:0] = 00 (No remapping)	TMR2_REMAP[1:0] = 01 (Partial remapping)	TMR2_REMAP[1:0] = 10 (Partial remapping)	TMR2_REMAP[1:0] = 11 (Full remapping)
TMR2_CH1_ETR	PA0	PA15	PA0	PA15
TMR2_CH2	PA1	PB3	PA1	PB3
TMR2_CH3	PA2		PB10	
TMR2_CH4	PA3		PB11	

Table 7-25 TMR1 Alternate Function Remapping

Alternate Function Mapping	TMR1_REMAP[1:0] = 00 (No remapping)	TMR1_REMAP[1:0] = 01 (Partial remapping)	TMR1_REMAP[1:0] = 11 (Full remapping) <small>(Note)</small>
TMR1_ETR	PA12		PE7
TMR1_CH1	PA8		PE9
TMR1_CH2	PA9		PE11
TMR1_CH3	PA10		PE13
TMR1_CH4	PA11		PE14
TMR1_BKIN	PB12	PA6	PE15
TMR1_CH1N	PB13	PA7	PE8
TMR1_CH2N	PB14	PB0	PE10
TMR1_CH3N	PB15	PB1	PE12

Note: Remapping only applies to 100-pin and 144-pin packages.

Table 7-26 TMR9 Alternate Function Remapping

Alternate Function Mapping	TMR9_REMAP = 0	TMR9_REMAP = 1
TMR9_CH1	PA2	PE5
TMR9_CH2	PA3	PE6

Table 7-27 TMR10 Alternate Function Remapping

Alternate Function Mapping	TMR10_REMAP = 0	TMR10_REMAP = 1
TMR10_CH1	PB8	PF6

Table 7-28 TMR11 Alternate Function Remapping

Alternate Function Mapping	TMR11_REMAP = 0	TMR11_REMAP = 1
TMR11_CH1	PB9	PF7

Table 7-29 TMR13 Alternate Function Remapping

Alternate Function Mapping	TMR13_REMAP = 0	TMR13_REMAP = 1
TMR13_CH1	PA6	PF8

Table 7-30 TMR14 Alternate Function Remapping

Alternate Function Mapping	TMR14_REMAP = 0	TMR14_REMAP = 1
TMR14_CH1	PA7	PF9

Table 7-31 TMR15 Alternate Function Remapping

Alternate Function Mapping	TMR15_REMAP = 0	TMR15_REMAP = 1
TMR15_ETR	PF7	PF14
TMR15_CH1	PF0	PG2
TMR15_CH2	PF2	PG4
TMR15_CH3	PF4	PG6
TMR15_CH4	PF6	PF13
TMR15_BKIN	PF8	PF15
TMR15_CH1N	PF1	PG3
TMR15_CH2N	PF3	PG5
TMR15_CH3N	PF5	PG7

Note: Remapping only applies to 144-pin packages.

7.4.7 USART Alternate Function Remapping

Please refer to AF remap and debug I/O configuration register (AFIO_MAP).

Table 7-32 USART3 Remapping

Alternate Function	USART3_REMAP[1:0] = 00 (No remapping)	USART3_REMAP[1:0] = 01 (Partial remapping) ^(Note 1)	USART3_REMAP[1:0] = 11 (Full remapping) ^(Note 2)
USART3_TX	PB10	PC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

Note: 1. Remapping only applies to 64-pin, 100-pin, and 144-pin packages.

2. Remapping only applies to 100-pin and 144-pin packages.

Table 7-33 USART2 Remapping

Alternate Function	USART2_REMAP = 0	USART2_REMAP = 1 ^(Note)
USART2_CTS	PA0	PD3
USART2_RTS	PA1	PD4
USART2_TX	PA2	PD5
USART2_RX	PA3	PD6
USART2_CK	PA4	PD7

Note: Remapping only applies to 100-pin and 144-pin packages.

Table 7-34 USART1 Remapping

Alternate Function	USART1_REMAP = 0	USART1_REMAP = 1
USART1_TX	PA9	PB6
USART1_RX	PA10	PB7

7.4.8 I²C Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register (AFIO_MAP).

Table 7-35 I²C1 Remapping

Alternate Function	I ² C1_REMAP = 0	I ² C1_REMAP = 1
I ² C1_SCL	PB6	PB8
I ² C1_SDA	PB7	PB9

Table 7-36 I²C3 Remapping

Alternate Function	I ² C3_REMAP = 0	I ² C3_REMAP = 1
I ² C3_SCL	PA8	
I ² C3_SDA	PC9 ^(Note)	PB4
I ² C3_SMBA	PA9	

Note: This does not apply to 48-pin packages.

7.4.9 SPI1/I2S1 Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register (AFIO_MAP).

Table 7-37 SPI1/I2S1 Remapping

Alternate Function	SPI1_REMAP[1:0] = 00	SPI1_REMAP[1:0] = 01	SPI1_REMAP[1:0] = 10
SPI1_NSS/I2S1_WS	PA4	PA15	PA4
SPI1_SCK/I2S1_CK	PA5	PB3	PA5
SPI1_MISO	PA6	PB4	PG0
SPI1_MOSI/I2S1_SD	PA7	PB5	PG1
I2S1_MCLK		PB0	

7.4.10 SPI4/I2S4 Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register 2 (AFIO_MAP2).

Table 7-38 SPI4 Remapping

Alternate Function	SPI4_REMAP = 0	SPI4_REMAP = 1
SPI4_NSS/I2S4_WS	PE4	PE12
SPI4_SCK/I2S4_CK	PE2	PE11
SPI4_MISO/	PE5	PE13
SPI4_MOSI/I2S4_SD	PE6	PE14
I2S4_MCLK		PC8

Note: Remapping only applies to 100-pin and 144-pin packages.

7.4.11 SDIO2 Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register 2 (AFIO_MAP2).

Table 7-39 SDIO2 Dx Alternate Function Remapping

Alternate Function	SDIO2_REMAP[0] = 0	SDIO2_REMAP[0] = 1
SDIO2_D0	PC0	PA4

SDIO2_D1	PC1	PA5
SDIO2_D2	PC2	PA6
SDIO2_D3	PC3	PA7
SDIO2_D4	PA4	-
SDIO2_D5	PA5	-
SDIO2_D6	PA6	-
SDIO2_D7	PA7	-

Table 7-40 SDIO2 CK/CMD Alternate Function Remapping

Alternate Function	SDIO2_REMAP[1] = 0	SDIO2_REMAP[1] = 1
SDIO2_CK	PC4	PA2
SDIO2_CMD	PC5	PA3

7.4.12 External SPIF Alternate Function Remapping

Please refer to the AF remap and debug I/O configuration register 2 (AFIO_MAP2).

Table 7-41 External SPIF Alternate Function Remapping

Alternate Function	EXT_SPIF_EN = 0	EXT_SPIF_EN = 1
SPIF_SCK	NA	PB1
SPIF_CS	NA	PA8
SPIF_TX	NA	PA11
SPIF_RX	NA	PA12
SPIF_HOLD_N	NA	PB6
SPIF_WP_N	NA	PB7

7.5 GPIO and AFIO Registers

Table 7-42 lists GPIO register map and reset values. These peripheral registers must be accessed by words (32-bit).

Table 7-42 GPIO Register Map and Reset Values

014h	GPIOx_BRE Reset Value	Reserved								BRE[15:0]											
										0 0											
018h	GPIOx_LOC K Reset Value	Reserved								LOCK[15:0]											
										0 0											

Table 7-43 lists AFIO register map and reset values. These peripheral registers must be accessed by words (32-bit).

Table 7-43 AFIO Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
000h	AFIO_EVCTRL RL	Reserved																															
004h	AFIO_MAP	SPI1_REMAP[1]	Reserved				Reserved				Reserved				Reserved				Reserved				Reserved				Reserved						
			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008H	AFIO_EXTIC 1	Reserved													EXTINT3[3: 0]				EXTINT2[3: 0]				EXTINT1[3: 0]				EXTINT0[3: 0]						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
00CH	AFIO_EXTIC 2	Reserved													EXTINT7[3: 0]				EXTINT6[3: 0]				EXTINT5[3: 0]				EXTINT4[3: 0]						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
010H	AFIO_EXTIC 3	Reserved													EXTINT11[3:0]				EXTINT10[3:0]				EXTINT9[3: 0]				EXTINT8[3: 0]						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
014H	AFIO_EXTIC 4	Reserved													EXTINT15[3:0]				EXTINT14[3:0]				EXTINT13[3:0]				EXTINT12[3: 0]						
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
01CH	AFIO_MAP2	Reserved													Reserved				Reserved				Reserved				Reserved						
		0	0	0	0	0	0	0	0	0	0	0	0	0	XMC_NADV_EN	0	0	0	TMR14_REMAP	0	0	0	TMR11_REMAP	0	0	0	TMR10_REMAP	0	0	0	TMR9_REMAP	0	0

Note: Before read and write access to the AFIO_EVCTRL, AFIO_MAP, and AFIO_EXTICRX registers, AFIO clock should be ON first. Please refer to [Section 3.3.7](#).

7.5.1 Port Configuration Register Low (GPIOx_CTRLLL) (x = A...E)

Address offset: 0x00

Reset value: 0x44444444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16														
CONF7[1:0]	MDE7[1:0]	CONF6[1:0]	MDE6[1:0]	CONF5[1:0]	MDE5[1:0]	CONF4[1:0]	MDE4[1:0]																						
rw 15	rw 14	rw 13	rw 12	rw 11	rw 10	rw 9	rw 8	rw 7	rw 6	rw 5	rw 4	rw 3	rw 2	rw 1	rw 0														
CONF3[1:0]	MDE3[1:0]	CONF2[1:0]	MDE2[1:0]	CONF1[1:0]	MDE1[1:0]	CONF0[1:0]	MDE0[1:0]																						

rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CONFy[1:0]: Portx configuration bit y (y = 0...7) Set by software to configure the corresponding I/O ports. Please refer to Table 7-1 for port bit configuration. In input mode (MDE[1:0] = 00): 00: Analog mode 01: Floating input (Reset state) 10: Input with pull-up/pull-down 11: Reserved In output mode (MDE[1:0] > 00): 00: General-purpose output push-pull 01: General-purpose output open-drain 10: Alternate function output push-pull 11: Alternate function output open-drain						
Bit 29:28 25:24 21:20 17:16 13:12 9:8,5:4 1:0	MDEy[1:0]: Portx Mode bit y (y = 0...7) Set by software to configure the corresponding I/O ports. Please refer to Table 7-1 for port bit configuration. 00: Input mode (state after reset) 01: Output mode; stronger push/pull capability (Typical value: 6 mA) 10: Output mode; moderate push/pull capability (Typical value: 4 mA; recommended, applicable for GPIO frequency below 50 MHz) 11: Output mode; maximum push/pull capability (Typical value: 15 mA) For detailed characteristics of push/pull capability, please refer to the I/O ports characteristics section in AT32F403 Series Data Book.						

Note: Some port registers have different reset values. For example, some pins of PA are JTAG/SWD input with pull-up by default.

7.5.2 Port Configuration Register High (GPIOx_CTRLH) (A...E)

Address offset: 0x04

Reset value: 0x44444444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONF15[1:0]	MDE15[1:0]	CONF14[1:0]	MDE14[1:0]	CONF13[1:0]	MDE13[1:0]	CONF12[1:0]	MDE12[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CONF11[1:0]	MDE11[1:0]	CONF10[1:0]	MDE10[1:0]	CONF9[1:0]	MDE9[1:0]	CONF8[1:0]	MDE8[1:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	CONFy[1:0]: Portx configuration bit y (y = 8...15) Set by software to configure the corresponding I/O ports. Please refer to Table 7-1 for port bit configuration. In input mode (MDE[1:0] = 00): 00: Analog mode 01: Floating input (Reset state) 10: Input with pull-up/pull-down 11: Reserved In output mode (MDE[1:0] > 00): 00: General-purpose output push-pull 01: General-purpose output open-drain 10: Alternate function output push-pull 11: Alternate function output open-drain
--	--

Bit 29:28	MDEy[1:0]: Portx Mode bit y (y = 8...15) Set by software to configure the corresponding I/O ports. Please refer to Table 7-1 for port bit configuration.
25:24	00: Input mode (Reset state)
21:20	01: Output mode, max. speed of 10 MHz
17:16	10: Output mode, max. speed of 2 MHz
13:12	11: Output mode, max. speed of 50 MHz
9:8	01: Output mode; stronger push/pull capability
5:4	10: Output mode; moderate push/pull capability
1:0	11: Output mode; maximum push/pull capability For detailed output characteristics, please refer to the I/O ports characteristics section in AT32F403 Series Data Book.

Note: Some port registers have different reset values. For example, some pins of PB are JTAG/SWD input with pull-up by default.

7.5.3 Port Input Data Register (GPIOx_IPTDT) (x = A...E)

Address offset: 0x08

Reset value: 0x0000XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPTD T[15]	IPTD T[14]	IPTD T[13]	IPTD T[12]	IPTD T[11]	IPTD T[10]	IPTD T[9]	IPTD T[8]	IPTD T[7]	IPTD T[6]	IPTD T[5]	IPTD T[4]	IPTD T[3]	IPTD T[2]	IPTD T[1]	IPTD T[0]
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:16		Reserved. Always read as 0.													
Bit 15:0		IPTDTy[15:0]: Port input data (y = 0...15) These bits are read-only and can be read only by word (16 bits). The value read is the status of the corresponding I/O port.													

7.5.4 Port Output Data Register (GPIOx_OPTDT) (x = A...E)

Address offset: 0x0C

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
res																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
OPTD T[15]	OPTD T[14]	OPTD T[13]	OPTD T[12]	OPTD T[11]	OPTD T[10]	OPT DT[9]	OPT DT[8]	OPT DT[7]	OPT DT[6]	OPT DT[5]	OPT DT[4]	OPT DT[3]	OPT DT[2]	OPT DT[1]	OPT DT[0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:16		Reserved. Always read as 0.														
Bit 15:0		OPTDTy[15:0]: Port output data (y = 0...15) These bits can be read or written, and are accessed only by word (16 bits). Note: For GPIOx_BSRE (x = A...E), each OPTDT bit can be set/reset individually.														

7.5.5 Port Bit Set/Reset Register (GPIOx_BSRE) (x = A...E)

Address offset: 0x10

Reset value: 0x00000000

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

Bit 31:16	BREy: Portx Reset bit y (y = 0...15) These bits are write-only and are accessed only by word (16 bits). 0: No action to the corresponding OPTDTy bits. 1: Reset the corresponding OPTDTy bit. Note: If both the corresponding bits of BSTy and MREy are set, the BSTy bit has higher priority.
Bit 15:0	BSTy: Portx Set bit y (y = 0...15) These bits are write-only and are accessed only by word (16 bits). 0: No action to the corresponding OPTDTy bits. 1: Set the corresponding OPTDTy bit.

7.5.6 Port Bit Reset Register (IOx_BRE) (x = A...E)

Address offset: 0x14

Reset value: 0x00000000

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

Bit 31:16	Reserved
Bit 15:0	BREy: Portx Reset bit y (y = 0...15) These bits are write-only and are accessed only by words (16 bits). 0: No action to the corresponding OPTDTy bits. 1: Reset the corresponding OPTDTy bit.

7.5.7 Port Configuration Lock Register (GPIOx_LOCK) (x = A...E)

When Bit 16 (LOCKK) is written with the correct sequence, this register is used to lock the configuration of port bit. Bit[15:0] is used to lock GPIO port configuration. During the specified write operation, LOCK[15:0] cannot be modified. After the corresponding port bit is written with LOCK sequence, the configuration of port bit cannot be modified before the next reset.

Each lock bit locks the corresponding 4 bits of the control register (CTRLL, CTRLH).

Address offset: 0x18

Reset value: 0x00000000

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

Bit 31:17	Reserved
Bit 16	<p>LOCKK: Lock key This bit can be read anytime, and is modified only by writing the lock key sequence. 0: Port configuration lock key is not activated. 1: Port configuration lock key is activated, and the GPIOx_LOCK register is locked before the next reset.</p> <p>Lock key writing sequence: Write 1-> Write 0-> Write 1-> Read 0-> Read 1 The last read operation is optional, but it can be used to confirm that the lock key is activated.</p> <p>Note: During the lock key writing sequence, the value of LCK[15:0] cannot be changed. Any errors in the lock key writing sequence will fail the lock key activation.</p>
Bit 15:0	<p>LOCKy: Portx Lock bit y (y = 0...15) These bits can be read and written, but they can be written only when the LOCKK bit is 0. 0: Port configuration is not locked. 1: Port configuration is locked.</p>

7.5.8 Alternate Event Control Register (AFIO_EVCTRL)

Address offset: 0x00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved																							
res																							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Reserved								EVOEN	PORT[2:0]		PIN[3:0]												
res								rw		rw		rw											
Bit 31:8																							
Reserved																							
Bit 7																							
EVOEN: Event output enable This bit can be read and written by software. After setting this bit, the EVENTOUT of Cortex®-M4 will connect to the I/O port selected by PORT[2:0] and PIN[3:0].																							
Bit 6:4																							
PORT[2:0]: Port selection Used to select the port for Cortex® EVENTOUT signal output: 000: Select PA 001: Select PB 010: Select PC 011: Select PD 100: Select PE																							
Bit 3:0																							
PIN[3:0]: Pin selection (x = A...E) Used to select the pin for Cortex EVENTOUT signal output: 0000: Select Px0 0001: Select Px1 0010: Select Px2 0011: Select Px3 0100: Select Px4 0101: Select Px5 0110: Select Px6 0111: Select Px7 1000: Select Px8 1001: Select Px9 1010: Select Px10 1011: Select Px11 1100: Select Px12 1101: Select Px13 1110: Select Px14 1111: Select Px15																							

7.5.9 AF Remap and Debug I/O Configuration Register (AFIO_MAP)

Address offset: 0x04

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
SPI1_RE MAP[1]	Reserved			SWJTAG_CONF[2:0]			Reserved			ADC2_EXTRG REG_RE MAP	ADC2_EXTRG INJ_RE MAP	ADC1_EXTRG REG_RE MAP	ADC1_EXTRG INJ_RE MAP	TMR5 CH4_INTRE		
	rw			res			rw			rw			rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PD01_REMAP	CAN_REMAP[1:0]	TMR4_REMAP	TMR3_REMAP[1:0]	TMR2_REMAP[1:0]	TMR1_REMAP[1:0]	USART3_REMAP[1:0]	USART2_REMAP	USART1_REMAP	I2C1_REMAP	SPI1_REMAP[0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31	SPI1_REMAP[1]: SPI1 remapping Specify the reference bit 0 of SPI1_REMAP[1:0].
Bit 30:27	Reserved
Bit 26:24	SWJTAG_CONF[2:0]: Serial wire JTAG configuration Only be written by software (reading these bits will return undefined values), and are used to configure SWJ and trace the I/O ports of alternate functions. SWJ (Serial Wire JTAG) supports JTAG or SWD access to the Cortex® debug port. The default state after system reset is SWJ ON without trace function. In this case, JTAG or SYSLKSEL (serial wire) mode can be selected with specific signal on the JTMS/JTCK pin. 000: Full SWJ (JTAG-DP+SW-DP): Reset state 001: Full SWJ (JTAG-DP+SW-DP) but without NJTRST 010: Disable JTAG-DP, enable SW-DP 100: Disable JTAG-DP, disable SW-DP Other combinations: No effect
Bit 23:21	Reserved
Bit 20	ADC2_EXTRGREG_REMAP: ADC2 external trigger regular conversion remapping Set and cleared by software. It controls the trigger input connected to ADC2 external trigger regular conversion. When it is cleared, ADC2 external trigger regular conversion is connected to EXTINT11; when it is set, ADC2 external trigger regular conversion is connected to TMR8_TRGO.
Bit 19	ADC2_EXTRGINJ_REMAP: ADC2 external trigger injected conversion remapping Set and cleared by software. It controls the trigger input connected to ADC2 external trigger injected conversion. When it is cleared, ADC2 external trigger injected conversion is connected to EXTINT15; when it is set, ADC2 external trigger injected conversion is connected to TMR8 channel 4.
Bit 18	ADC1_EXTRGREG_REMAP: ADC1 external trigger regular conversion remapping Set and cleared by software. It controls the trigger input connected to ADC2 external trigger regular conversion. When it is cleared, ADC1 external trigger regular conversion is connected to EXTINT11; when it is set, ADC1 external trigger regular conversion is connected to TMR8_TRGO.
Bit 17	ADC1_EXTRGINJ_REMAP: ADC1 external trigger injected conversion remapping Set and cleared by software. It controls the trigger input connected to ADC2 external trigger injected conversion. When it is cleared, ADC2 external trigger injected conversion is connected to EXTINT15; when it is set, ADC1 external trigger injected conversion is connected to TMR8 channel 4.
Bit 16	TMR5CH4_INTLRE: TMR5 channel4 internal remapping Set and cleared by software. It controls TMR5 channel 4 internal mapping. When it is cleared, TMR5_CH4 is connected to PA3; when it is set, LSI internal oscillator is connected to TMR5_CH4, with the purpose to calibrate LSI.
Bit 15	PD01_REMAP: PortD0/PortD1 mapping on OSC_IN/OSC_OUT Set and cleared by software. It controls PD0 and PD1 GPIO function mapping. When main oscillator HSE is not used (system running on internal 8 MHz RC), PD0 and PD1 can map to OSC_IN and OSC_OUT pins. This function only applies to 48-pin and 64-pin packages (PD0 and PD1 are available in 100-pin and 144-pin packages, remapping is not needed). 0: No PD0 and PD1 remapping 1: PD0 is remapped on OSC_IN, and PD1 is remapped on OSC_OUT.

Bit 14:13	CAN_REMAP[1:0]: CAN alternate function remapping Set and cleared by software. They control the remapping of alternate functions CAN_RX and CAN_TX in devices with a single CAN interface. 00: CAN_RX is remapped on PA11, and CAN_TX is remapped on PA12. 01: Unused combination 10: CAN_RX is remapped on PB8, and CAN_TX is remapped on PB9. 11: CAN_RX is remapped on PD0, and CAN_TX is remapped on PD1.
Bit 12	TMR4_REMAP: TMR4 remapping Set and cleared by software. It controls the mapping of TIM4 channels 1 to 4 on the GPIO ports. 0: No remapping (TMR4_CH1/PB6, TMR4_CH2/PB7, TMR4_CH3/PB8, TMR4_CH4/PB9) 1: Full remapping (TMR4_CH1/PD12, TMR4_CH2/PD13, TMR4_CH3/PD14, TMR4_CH4/PD15) Note: Remapping does not affect TMR4_ETR on PE0.
Bit 11:10	TMR3_REMAP[1:0]: TMR3 remapping Set and cleared by software. It controls the mapping of TIM3 channels 1 to 4 on the GPIO ports. 00: No remapping (CH1/PA6, CH2/PA7, CH3/PB0, CH4/PB1) 01: Unused combination 10: Partial remapping (CH1/PB4, CH2/PB5, CH3/PB0, CH4/PB1) 11: Full remapping (CH1/PC6, CH2/PC7, CH3/PC8, CH4/PC9). Note: Remapping does not affect TMR3_ETR on PD2.
Bit 9:8	TMR2_REMAP[1:0]: TMR2 remapping Set and cleared by software. It controls the mapping of TIM2 channels 1 to 4 and external trigger (ETR) on the GPIO ports. 00: No remapping (CH1/ETR/PA0, CH2/PA1, CH3/PA2, CH4/PA3) 01: Partial remapping (CH1/ETR/PA15, CH2/PB3, CH3/PA2, CH4/PA3) 10: Partial remapping (CH1/ETR/PA0, CH2/PA1, CH3/PB10, CH4/PB11) 11: Full remapping (CH1/ETR/PA15, CH2/PB3, CH3/PB10, CH4/PB11)
Bit 7:6	TMR1_REMAP[1:0]: TMR1 remapping Set and cleared by software. It controls the mapping of TIM1 channels 1 to 4, 1N to 3N, external trigger (ETR), and break input (BKIN) on the GPIO ports. 00: No remapping (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PB12, CH1N/PB13, CH2N/PB14, CH3N/PB15) 01: Partial remapping (ETR/PA12, CH1/PA8, CH2/PA9, CH3/PA10, CH4/PA11, BKIN/PA6, CH1N/PA7, CH2N/PB0, CH3N/PB1) 10: Unused combination 11: Full remapping (ETR/PE7, CH1/PE9, CH2/PE11, CH3/PE13, CH4/PE14, BKIN/PE15, CH1N/PE8, CH2N/PE10, CH3N/PE12)
Bit 5:4	USART3_REMAP[1:0]: USART3 remapping Set and cleared by software. It controls the mapping of USART3 CTS, RTS, CK, TX, and RX alternate function on the GPIO ports. 00: No remapping (TX/PB10, RX/PB11, CK/PB12, CTS/PB13, RTS/PB14) 01: Partial remapping (TX/PC10, RX/PC11, CK/PC12, CTS/PB13, RTS/PB14) 10: Unused combination 11: Full remapping (TX/PD8, RX/PD9, CK/PD10, CTS/PD11, RTS/PD12)
Bit 3	USART2_REMAP: USART2 remapping Set and cleared by software. It controls the mapping of USART2 CTS, RTS, CK, TX, and RX alternate function on the GPIO ports. 0: No remapping (CTS/PA0, RTS/PA1, TX/PA2, RX/PA3, CK/PA4) 1: Remapping (CTS/PD3, RTS/PD4, TX/PD5, RX/PD6, CK/PD7)
Bit 2	USART1_REMAP: USART1 remapping Set and cleared by software. It controls the mapping of USART1 TX and RX alternate function on the GPIO ports. 0: No remapping (TX/PA9, RX/PA10) 1: Remapping (TX/PB6, RX/PB7)

Bit 1	I2C1_REMAP: I2C1 remapping Set and cleared by software. It controls the mapping of I2C1 SCL and SDA alternate function on the GPIO ports. 0: No remapping (SCL/PB6, SDA/PB7) 1: Remapping (SCL/PB8, SDA/PB9)
Bit 0	SPI1_REMAP[0]: SPI1 remapping SPI1_REMAP[1] set in Bit 31. SPI1_REMAP[1:0] can be set '00', '01', or '10' by software to control the mapping of SPI1 NSS, SCK, MISO, and MOSI alternate function on the GPIO ports. 00: No remapping (NSS/PA4, SCK/PA5, MISO/PA6, MOSI/PA7) 01: Remapping (NSS/PA15, SCK/PB3, MISO/PB4, MOSI/PB5) 10: Partial remapping (NSS/PA4, SCK/PA5, MISO/PG0, MOSI/PG1) 11: Unused combination

7.5.10 Alternate External Interrupt Configuration Register 1 (AFIO_EXTIC1)

Address offset: 0x08

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16												
Reserved																											
res																											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
EXTINT3[3:0]				EXTINT2[3:0]				EXTINT1[3:0]				EXTINT0[3:0]															
rw				rw				rw				rw															
Bit 31:16		Reserved																									
Bit 15:0		EXTINTx[3:0]: EXTINTx configuration (x = 0...3) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin																									

7.5.11 Alternate External Interrupt Configuration Register 2 (AFIO_EXTIC2)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTINT7[3:0]				EXTINT6[3:0]				EXTINT5[3:0]				EXTINT4[3:0]			
rw				rw				rw				rw			
Bit 31:16		Reserved													

Bit 15:0	EXTINTx[3:0]: EXTINTx configuration (x = 4...7) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin
----------	---

7.5.12 Alternate External Interrupt Configuration Register 3 (AFIO_EXTIC3)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Reserved																			
res																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
EXTINT11[3:0]				EXTINT10[3:0]				EXTINT9[3:0]				EXTINT8[3:0]							
rw				rw				rw				rw							
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Bit 31:16</td><td style="padding: 2px;">Reserved</td></tr> <tr> <td style="padding: 2px;">Bit 15:0</td><td style="padding: 2px; vertical-align: top;"> EXTINTx[3:0]: EXTINTx configuration (x = 8...11) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin </td></tr> </table>		Bit 31:16	Reserved													Bit 15:0	EXTINTx[3:0]: EXTINTx configuration (x = 8...11) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin		
Bit 31:16	Reserved																		
Bit 15:0	EXTINTx[3:0]: EXTINTx configuration (x = 8...11) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin																		

7.5.13 Alternate External Interrupt Configuration Register 4 (AFIO_EXTIC4)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Reserved																			
res																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
EXTINT15[3:0]				EXTINT14[3:0]				EXTINT13[3:0]				EXTINT12[3:0]							
rw				rw				rw				rw							
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Bit 31:16</td><td style="padding: 2px;">Reserved</td></tr> <tr> <td style="padding: 2px;">Bit 15:0</td><td style="padding: 2px; vertical-align: top;"> EXTINTx[3:0]: EXTINTx configuration (x = 12...15) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin </td></tr> </table>		Bit 31:16	Reserved													Bit 15:0	EXTINTx[3:0]: EXTINTx configuration (x = 12...15) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin		
Bit 31:16	Reserved																		
Bit 15:0	EXTINTx[3:0]: EXTINTx configuration (x = 12...15) Can be read and written by software. Used to select the input source for EXTINTx external interrupt. 0000: PA[x] pin 0100: PE[x] pin 0001: PB[x] pin 0101: PF[x] pin 0010: PC[x] pin 0110: PG[x] pin 0011: PD[x] pin																		

7.5.14 AF Remap and Debug I/O Configuration Register 2 (AFIO_MAP2)

Address offset: 0x1C

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
EXT_SPIF_EN	SDIO2_REMA_P[1:0]	I2C3_REMA_P	SPI4_REMA_P	Reserve d											
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					XMC_NADV_REMAP_AP	TMR1_4_REMAP	TMR1_3_REMAP	TMR1_1_REMAP	TMR1_0_REMAP	TMR9_REMAP_AP		Reserved		TMR1_5_REMAP	
res					rw	rw	rw	rw	rw	rw	rw	res		rw	
Bit 31:22	Reserved														
Bit 21	EXT_SPIF_EN: External SPI Flash Interface enable Set and cleared by software. It controls whether to use external SPI Flash. Please refer to Section 7.4.12 .														
Bit 20:19	SDIO2_REMAP[1:0]: SDIO2 internal remap Set and cleared by software. It controls SDIO2 internal remapping. Please refer to Section 7.4.11 .														
Bit 18	I2C3_REMAP: I2C3 internal remap Set and cleared by software. It controls I2C3 internal remapping. Please refer to Section 7.4.8 .														
Bit 17	SPI4_REMAP: SPI4 internal remap Set and cleared by software. It controls SPI4 internal remapping. Please refer to Section 7.4.10 .														
Bit 16:11	Reserved														
Bit 10	XMC_NADV_REMAP: XMCNADV connect Set and cleared by software. It controls whether to use XMC_NADV signal. 0: XMC_NADV is connected to pin. (By default) 1: XMC_NADV is unused, and the corresponding pin can be used by other peripherals.														
Bit 9	TMR14_REMAP: TMR14 channel1 internal remap Set and cleared by software. It controls TMR14 channel1 internal remapping. When it is cleared, TMR14_CH1 is connected to PA7. When it is set, TMR14_CH1 is connected to PF9.														
Bit 8	TMR13_REMAP: TMR13 channel1 internal remap Set and cleared by software. It controls TMR13 channel1 internal remapping. When it is cleared, TMR13_CH1 is connected to PA6. When it is set, TMR13_CH1 is connected to PF8.														
Bit 7	TMR11_REMAP: TMR11 channel1 internal remap Set and cleared by software. It controls TMR11 channel1 internal remapping. When it is cleared, TMR11_CH1 is connected to PB9. When it is set, TMR11_CH1 is connected to PF7.														
Bit 6	TMR10_REMAP: TMR10 channel1 internal remap Set and cleared by software. It controls TMR10 channel1 internal remapping. When it is cleared, TMR10_CH1 is connected to PB8. When it is set, TMR10_CH1 is connected to PF6.														
Bit 5	TMR9_REMAP: TMR9 channel1/2 internal remap Set and cleared by software. It controls TMR9 channel1/2 internal remapping. When it is cleared, TMR9_CH1 is connected to PA2, TMR9_CH2 is connected to PA3. When it is set, TMR9_CH1 is connected to PE5, TMR9_CH2 is connected to PE6.														
Bit 4:1	Reserved														
Bit 0	TMR15_REMAP: TMR15 internal remap Set and cleared by software. It controls TMR15 internal remapping. Please refer to Section 7.4.6 .														

8 Interrupts and Events

8.1 Nested Vectored Interrupt Controller

Features

- 68 maskable interrupt channels (not including 16 Cortex®-M4F interrupt lines);
- 16 programmable priority levels (4-bit interrupt priority configuration);
- Low-latency processing of exceptions and interrupts;
- Power management control;
- Implementation of system control register;

Nested vectored interrupt controller (NVIC) is closely connected to the processor core interface, which enables low-latency interrupt processing and highly efficient late-arriving interrupt processing.

Nested vectored interrupt controller also manages interrupts such as core exception.

8.1.1 System Tick (SysTick) Calibration Value Register

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 (max. of HCLK/8).

8.1.2 Interrupt and Exception Vectors

Table 8-1 lists the vector table of AT32F403 series.

Table 8-1 Vector Table of AT32F403 Series

Position	Priority	Priority Type	Name	Description	Address
-	-	-	-	Reserved	0x0000_0000
-3	Fixed		Reset	Reset	0x0000_0004
-2	Fixed		NMI	Non maskable interrupt The RCC Clock Failure Detection (CFD) is linked to the NMI vector.	0x0000_0008
-1	Fixed		HardFault	All class of fault	0x0000_000C
0	Configurable		MemoryManage	Memory management	0x0000_0010
1	Configurable		BusFault	Pre-fetch fault, 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		DebugMonitor	Debug Monitor	0x0000_0030
-	-	-	-	Reserved	0x0000_0034
5	Configurable		PendSV	Pendable request for system service	0x0000_0038
6	Configurable		SysCNTRick	System tick timer	0x0000_003C
0	7	Configurable	WWDG	Window Watchdog interrupt	0x0000_0040
1	8	Configurable	PVD	PVD from EXTI interrupt	0x0000_0044
2	9	Configurable	TAMPER	Tamper interrupt	0x0000_0048
3	10	Configurable	RTC	RTC global interrupt	0x0000_004C

4	11	Configurable	FLASH	Flash global interrupt	0x0000_0050
5	12	Configurable	RCC	Reset and RCC interrupt	0x0000_0054
6	13	Configurable	EXTINT0	EXTI Line0 interrupt	0x0000_0058
7	14	Configurable	EXTINT1	EXTI Line1 interrupt	0x0000_005C
8	15	Configurable	EXTINT2	EXTI Line2 interrupt	0x0000_0060
9	16	Configurable	EXTINT3	EXTI Line3 interrupt	0x0000_0064
10	17	Configurable	EXTINT4	EXTI Line4 interrupt	0x0000_0068
11	18	Configurable	DMA1 channel1	DMA1 channel1 global interrupt	0x0000_006C
12	19	Configurable	DMA1 channel2	DMA1 channel2 global interrupt	0x0000_0070
13	20	Configurable	DMA1 channel3	DMA1 channel3 global interrupt	0x0000_0074
14	21	Configurable	DMA1 channel4	DMA1 channel4 global interrupt	0x0000_0078
15	22	Configurable	DMA1 channel5	DMA1 channel5 global interrupt	0x0000_007C
16	23	Configurable	DMA1 channel6	DMA1 channel6 global interrupt	0x0000_0080
17	24	Configurable	DMA1 channel7	DMA1 channel7 global interrupt	0x0000_0084
18	25	Configurable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088
19	26	Configurable	USB_HP_CAN_TX	USB high priority or CAN TX interrupt	0x0000_008C
20	27	Configurable	USB_LP_CAN_RXS0	USB low priority or CAN RX0 interrupt	0x0000_0090
21	28	Configurable	CAN_RXS1	CAN RX1 interrupt	0x0000_0094
22	29	Configurable	CAN_SCE	CAN SCE interrupt	0x0000_0098
23	30	Configurable	EXTINT9_5	EXTI 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_OV_TMR10	TMR1 update interrupt and TMR10 global interrupt	0x0000_00A4
26	33	Configurable	TMR1_TRG_COM_TM_R11	TMR1 trigger and communication interrupt and TMR11 global interrupt	0x0000_00A8
27	34	Configurable	TMR1_CC	TMR1 capture compare 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_EV	I ² C1 event interrupt	0x0000_00BC
32	39	Configurable	I2C1_ER	I ² C1 error interrupt	0x0000_00C0
33	40	Configurable	I2C2_EV	I ² C2 event interrupt	0x0000_00C4
34	41	Configurable	I2C2_ER	I ² C2 error interrupt	0x0000_00C8
35	42	Configurable	SPI1	SPI1 global interrupt	0x0000_00CC
36	43	Configurable	SPI2	SPI2 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	EXTINT15_10	EXTI Line[15:10] interrupt	0x0000_00E0
41	48	Configurable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4
42	49	Configurable	USBWakeUp	USB Wakeup through EXTI line interrupt	0x0000_00E8
43	50	Configurable	TMR8_BRK_TMR12	TMR8 Break interrupt and TMR12 global interrupt	0x0000_00EC
44	51	Configurable	TMR8_OV_TMR13	TMR8 update interrupt and TMR13 global interrupt	0x0000_00F0
45	52	Configurable	TMR8_TRG_COM_TM R14	TMR8 trigger and communication interrupt and TMR14 global interrupt	0x0000_00F4
46	53	Configurable	TMR8_CC	TMR8 capture compare interrupt	0x0000_00F8
47	54	Configurable	ADC3	ADC3 global interrupt	0x0000_00FC
48	55	Configurable	XMC	XMC global interrupt	0x0000_0100
49	56	Configurable	SDIO	SDIO global interrupt	0x0000_0104
50	57	Configurable	TMR5	TMR5 global interrupt	0x0000_0108
51	58	Configurable	SPI3	SPI3 global interrupt	0x0000_010C
52	59	Configurable	UART4	UART4 global interrupt	0x0000_0110
53	60	Configurable	UART5	UART5 global interrupt	0x0000_0114
54	61	Configurable	TMR6	TMR6 global interrupt	0x0000_0118
55	62	Configurable	TMR7	TMR7 global interrupt	0x0000_011C
56	63	Configurable	DMA2 channel1	DMA2 channel1 global interrupt	0x0000_0120
57	64	Configurable	DMA2 channel2	DMA2 channel2 global interrupt	0x0000_0124
58	65	Configurable	DMA2 channel3	DMA2 channel3 global interrupt	0x0000_0128
59	66	Configurable	DMA2 channel4_5	DMA2 channel4 and DMA2 channel5 global interrupt	0x0000_012C
60	67	Configurable	SDIO2	SDIO2 global interrupt	0x0000_0130
61	68	Configurable	I2C3_EV	I2C3 event interrupt	0x0000_0134
62	69	Configurable	I2C3_ER	I2C3 error interrupt	0x0000_0138
63	70	Configurable	SPI4	SPI4 global interrupt	0x0000_013C
64	71	Configurable	TMR15_BRK	TMR15 break interrupt	0x0000_0140
65	72	Configurable	TMR15_OV	TMR15 update interrupt	0x0000_0144
66	73	Configurable	TMR15_TRG_COM	TMR15 trigger and communication interrupt	0x0000_0148
67	74	Configurable	TMR15_CC	TMR15 capture compare interrupt	0x0000_014C

8.2 External Interrupt/Event Controller (EXTI)

In connectivity line devices, the external interrupt/event controller consists of 20 edge detectors which generate event/interrupt requests. In other devices, there are 19 edge detectors for generating event/interrupt requests. Each input line can be independently configured to select input type (event or interrupt) and the corresponding trigger event (rising edge, falling edge, or both edges). Each input line can be masked independently. Pending register maintains the interrupt requests of status line.

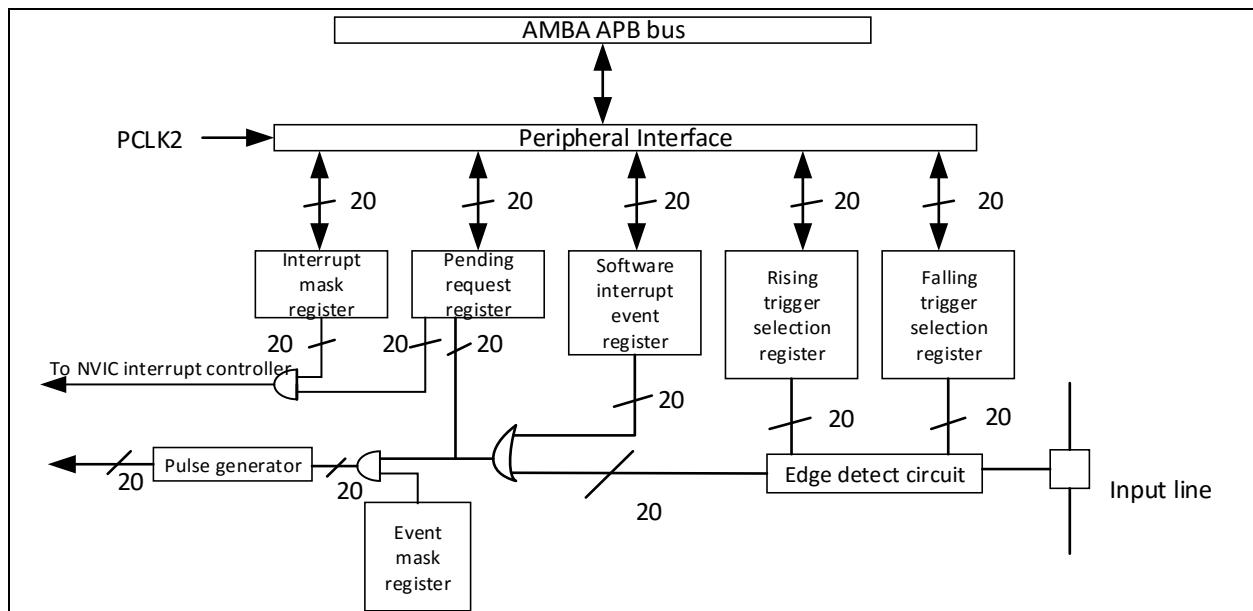
8.2.1 Main Features

The main features of EXTI controller are as follows:

- Independent trigger and mask on each interrupt/event
- Dedicated status bit for each interrupt line
- Up to 20 software event/interrupt requests
- Detection of external signal with pulse width lower than APB2 clock period. Please refer to the electrical characteristics section of the data sheet for detail on this parameter.

8.2.2 Block Diagram

Figure 8-1 External Interrupt/Event Controller Block Diagram



8.2.3 Wakeup Event Management

AT32F403 can wake up the core (WFE) by managing external or internal events. Wakeup events can be generated by the following configuration:

- Enabling an interrupt in the peripheral control register but not in NVIC, and enabling the SEVONPEND bit in the Cortex®-M4F system control register. When CPU resumes from WFE, the corresponding peripheral interrupt pending bit and the peripheral NVIC IRQ channel pending bit (in the NVIC interrupt clear pending register) should be cleared.
- Configuring an external or internal EXTI line as event mode. When CPU resumes from WFE, it is not necessary to clear the corresponding peripheral interrupt pending bit or the NVIC IRQ channel pending bit since the pending bit corresponding to the event line is not set.

In connectivity line devices, Ethernet wakeup events also have the WFE wakeup capability.

To use an external I/O port as a wakeup event, please refer to [Section 8.2.4](#).

8.2.4 Function Overview

To generate interrupts, the interrupt line should be configured and enabled first. This is done by programming the two trigger registers with the desired edge detection and enabling the interrupt request by writing '1' to the corresponding bit in the interrupt mask register. When the selected edge occurs on the external interrupt line, an interrupt request is generated. The pending bit corresponding to the interrupt line is also set. This request is reset by writing '1' to the corresponding bit in pending register.

To generate events, the event line should be configured and enabled first. This is done by programming the two trigger registers with the desired edge detection and enabling the event request by writing '1' to the corresponding bit in the event mask register. When the selected edge occurs on the event line, an event request pulse is generated. The pending bit corresponding to the event line is not set.

An interrupt/event request can also be generated by software when writing '1' to the software interrupt/event register.

Hardware interrupt selection

To configure the 20 lines as interrupt sources, use the following procedure:

- Configure the mask bits of the 20 interrupt lines (EXTI_INTEN)
- Configure the trigger selection bits of the selected interrupt lines (EXTI_RTRSEL and EXTI_FTRSEL)
- Configure the enable and mask bits corresponding to the NVIC IRQ channel of external interrupt controller (EXTI), so that interrupts from the 20 lines can be correctly served.

Hardware event selection

To configure the 20 lines as event sources, use the following procedure:

- Configure the mask bits of the 20 event lines (EXTI_EVTEN)
- Configure the trigger selection bits of the event lines (EXTI_RTRSEL and EXTI_FTRSEL)

Software interrupt/event selection

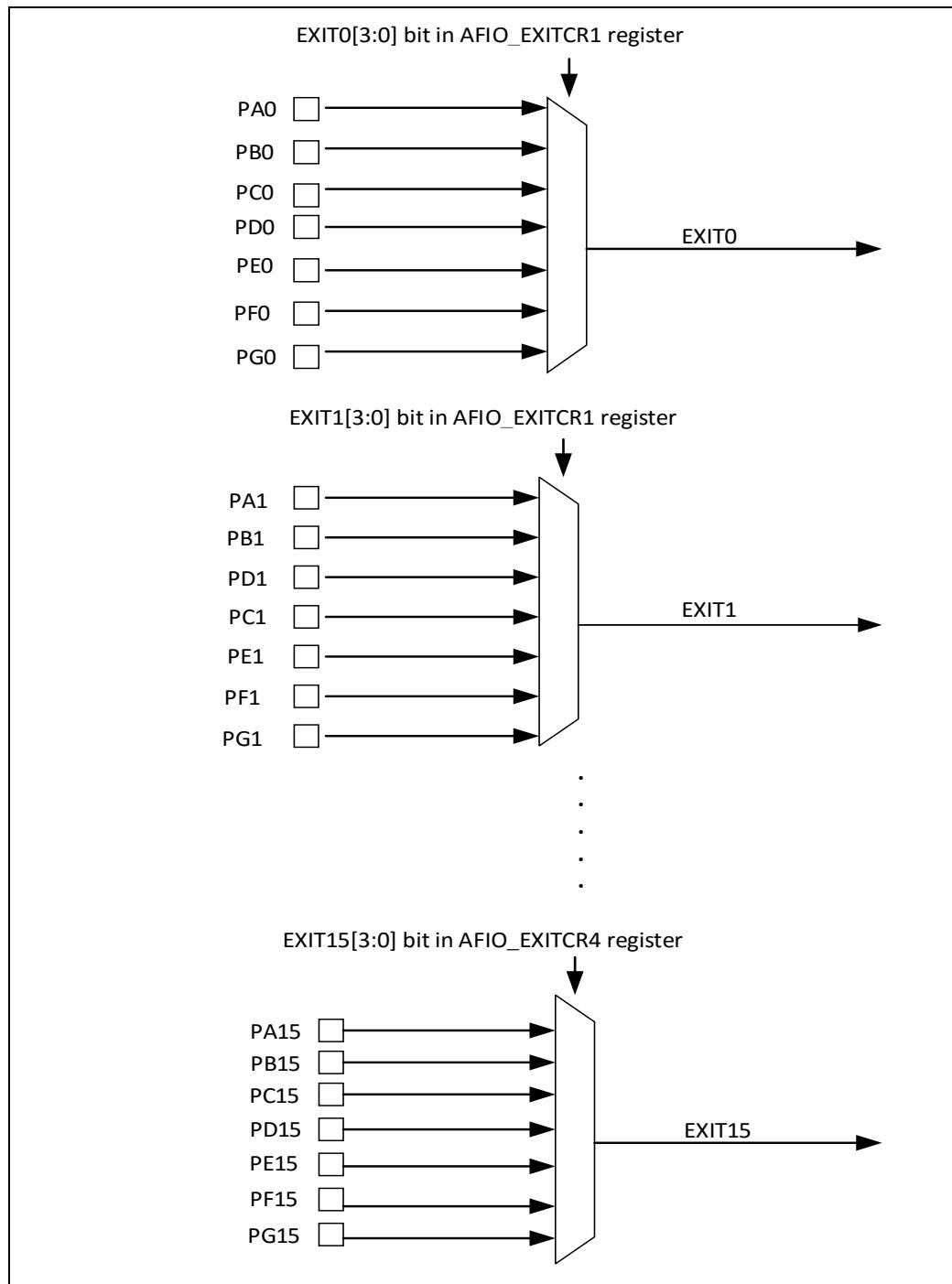
The 20 lines can be configured as software interrupt/event lines. The following is the procedure to generate a software interrupt:

- Configure the mask bits of the 20 interrupt/event lines (EXTI_INTEN, EXTI_EVTEN)
- Set the request bit in the software interrupt register (EXTI_SWIE)

8.2.5 External Interrupt/Event Line Mapping

The 112 GPIOs are connected to the 16 external interrupt/event lines in the following manners shown in Figure 8-2:

Figure 8-2 External Interrupt GPIO Mapping



To configure the external interrupt/event on GPIO lines with AFIO_EXITCRx, the AFIO clock should first be enabled. Please refer to [Section 3.3.7](#).

The other four EXTI lines are connected as follows:

- EXTI line 16 is connected to the PVD output.
- EXTI line 17 is connected to the RTC alarm event.
- EXTI line 18 is connected to the USB wakeup event.
- EXTI line 19 is connected to the Ethernet Wakeup event (available only in connectivity line devices)

8.3 EXTI Registers Description

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

Table 8-2 lists the EXTI register map and reset values. Bit 19 in all the registers only applies to

connectivity line devices, and it is reserved in other devices.

Table 8-2 External Interrupt/Event Controller Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	EXTI_INTEN	Reserved												MR[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04	EXTI_EVTE_N	Reserved												MR[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x08	EXTI_RTRS_EL	Reserved												TR[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0C	EXTI_FTRS_EL	Reserved												TR[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	EXTI_SWIE	Reserved												SWIE[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	EXTI_PR	Reserved												PR[19:0]																			
														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

8.3.1 Interrupt Mask Register (EXTI_INTEN)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																				
Reserved												MR19	MR18	MR17	MR16																				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0																				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			
Bit 31:20												Reserved. Must be kept at reset value '0'.																							
Bit 19:0												MRx: Interrupt Mask on line x 0: Interrupt request from Line x is masked. 1: Interrupt request from Line x is not masked. Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.																							

8.3.2 Event Mask Register (EXTI_EVTEN)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																					
Reserved												MR19	MR18	MR17	MR16																					

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0		

Bit 31:20	Reserved. Must be kept at reset value '0'.
Bit 19:0	<p>MRx: Event Mask on line x</p> <p>0: Event request from Line x is masked. 1: Event request from Line x is not masked.</p> <p>Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.</p>

8.3.3 Rising Edge Trigger Selection Register (EXTI_RTRSEL)

Address offset: 0x08

Reset value: 0x0000 0000

Bit 31:20	Reserved. Must be kept at reset value '0'.
Bit 19:0	TRx : Rising trigger event configuration bit of line x 0: Rising trigger (event and interrupt) on input line x is disabled. 1: Rising trigger (event and interrupt) on input line x is enabled. Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.

Note: The external wakeup lines are edge triggered, and no glitches are generated on these lines. When the EXTI_RTRSEL register is written, rising edge signal on external interrupt line cannot be identified, and the pending bit will not be set, either. Rising and falling edge triggers can be set at the same time for the same interrupt line. In this configuration, both edges can trigger interrupts.

8.3.4 Falling Edge Trigger Selection Register (EXTI_FTRSEL)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														TR19	TR18	TR17	TR16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0		

Bit 31:20	Reserved. Must be kept at reset value '0'.
Bit 19:0	TRx: Falling trigger event configuration bit of line x 0: Falling trigger (event and interrupt) on input line x is disabled. 1: Falling trigger (event and interrupt) on input line x is enabled. Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.

Note: The external wakeup lines are edge triggered, and no glitches are generated on these lines. When the `EXTI_RTRSEL` register is written, falling edge signal on external interrupt line cannot be identified, and the pending bit will not be set, either. Rising and falling edge triggers can be set at the same time for the same interrupt line. In this configuration, both edges can trigger interrupts.

8.3.5 Software Interrupt Event Register (EXTI_SWIE)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														SW IER19	SW IER18	SW IER17	SW IER16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
SW IER15	SW IER14	SW IER13	SW IER12	SW IER11	SW IER10	SW IER9	SW IER8	SW IER7	SW IERR6	SW IER5	SW IER4	SW IER3	SW IER2	SW IER1	SW IER0		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:20		Reserved. Must be kept at reset value '0'.															
Bit 19:0		SWIERx: Software interrupt on line x When this bit is '0', the corresponding pending bit in the <code>EXTI_PR</code> register will be set by writing '1'. If interrupt is enabled in the <code>EXTI_INTEN</code> and the <code>EXTI_EVTEN</code> registers, an interrupt will be generated at this time. Note: This bit is cleared by clearing the corresponding bit in the <code>EXTI_PR</code> register (by writing a '1' to the bit). Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.															

8.3.6 Pending Register (EXTI_PR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														PR19	PR18	PR17	PR16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:20		Reserved. Must be kept at reset value '0'.															
Bit 19:0		PRx: Pending bit 0: No trigger request occurred. 1: Selected trigger request occurred. This bit is set when the selected edge event occurs on the external interrupt line. This bit is cleared by writing '1' or by changing the polarity of edge detection. Note: Bit 19 is used in connectivity line devices only and is reserved otherwise.															

9 DMA Controller (DMA)

9.1 DMA Introduction

Direct memory access (DMA) provides high-speed data transfer between peripheral and memory, or between memory and memory. Without the intervention of CPU, data can be quickly transferred through DMA, keeping CPU resources free for other operations.

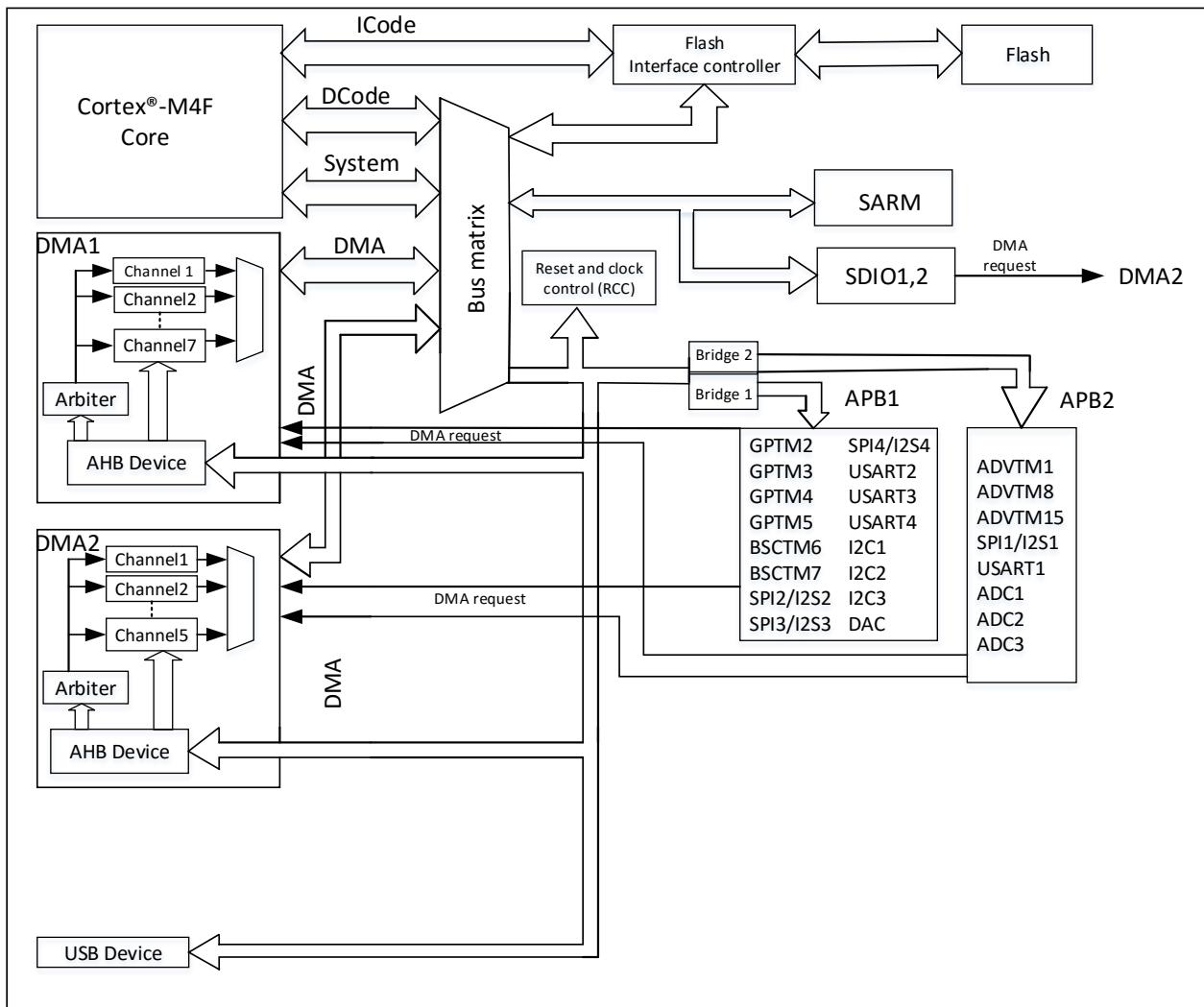
There are 12 channels in the two DMA controllers (7 in DMA1, 5 in DMA2). Each channel specifically manages memory access requests from one or more peripherals. There is one arbiter for coordinating the priority of each DMA request.

9.2 DMA Main Features

- 12 independently configurable channels (requests): 7 for DMA1 and 5 for DMA2
- Each of the 12 channels is connected to the dedicated hardware DMA requests and supports software trigger. This configuration is done by software.
- On the same DMA block, priorities between several requests are software programmable (4 levels including very high, high, medium, and low). In case of equality, priority is given by hardware (request 0 has priority over request 1, and so on).
- Independent source and destination transfer size (byte, half-word, and word), simulation of packing and unpacking. Source/Destination addresses must be aligned according to the data size.
- Support circular buffer management
- 3 event flags on each channel (DMA half transfer, DMA transfer complete, and DMA transfer error). The 3 events are logically ORed together in a single interrupt request.
- Memory-to-memory transfer
- Peripheral-to-memory, memory-to-peripheral, and peripheral-to-peripheral transfers
- Flash, peripheral SRAM, APB1, APB2, and AHB peripherals can be accessed as source and destination.
- Programmable amount of data to be transferred: Up to 65535

Figure 9-1 shows the function block diagram:

Figure 9-1 DMA Block Diagram



Note: The number of DMA peripherals in Figure 9-1 may decrease depending on different models.

9.3 Function Overview

DMA Controller and Cortex®-M4F core share the same system data bus, executing direct memory data transfer. When CPU and DMA access the same destination (RAM or peripherals), DMA request will stop CPU access to system bus for several cycles. The bus arbiter implements round-robin scheduling to ensure at least half of the system bus bandwidth (memory or peripheral) for CPU.

9.3.1 DMA Transaction

After an event, the peripheral asserts a request signal to the DMA controller, which serves the request according to the channel priorities. Afterwards, the DMA controller accesses the peripheral, and asserts an 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 deasserts the acknowledge signal. If there are more requests, the peripheral can initiate the next transaction.

To sum up, each DMA transfer involves the following procedure:

- Load data from the memory address programmed by peripheral data register or by current peripheral/memory address register. The start address used for the first transfer is the peripheral base address or memory unit programmed in the DMA_CPB_{Ax} or DMA_CMB_{Ax} register.
- Store the data loaded to the memory address programmed by peripheral data register or current peripheral/memory address register. The start address used for the first transfer is the peripheral base address or memory unit programmed in the

DMA_CPBAX or DMA_CMBAX register.

- Execute a decrementing operation of the DMA_TCNTx register, which contains the number of unfinished transactions.

9.3.2 Arbiter

The arbiter manages the channel requests based on their priorities to enable peripheral/memory access.

There are two stages in priority management:

- Software: The priority of each channel can be configured in the DMA_CHCTRLx register. There are four levels:
 - Very high priority
 - High priority
 - Medium priority
 - Low priority
- Hardware: If 2 requests have the same software priority level, the channel with lower number will get priority over the channel with higher number. For example, channel 2 gets priority over channel 4.

Note: The DMA1 controller has priority over the DMA2 controller.

9.3.3 DMA Channels

Each channel can handle DMA transfer between a peripheral register located at a fixed address and a memory address. The amount of data to be transferred is programmable (up to 65535). The register which contains the amount of data items to be transferred is decremented after each transaction.

Programmable data size

The transferred data size of peripheral and memory can be programmed through the PWIDHT and MWIDHT bits in the DMA_CHCTRLx register.

Pointer incrementation

By configuring the PINC and MINC flag bits in the DMA_CHCTRLx register, peripheral and memory pointers can be selectively and automatically post-incremented after each transaction. If incremented mode is enabled, the next transfer address will be the previous address incremented by 1, 2, or 4, depending on the chosen data size. The first transfer address is the one programmed in the DMA_CPBAX/DMA_CMBAX registers. During transfer process, these registers keep their initially programmed values. The current transfer addresses (in the current internal peripheral/memory address register) cannot be modified by software.

If the channel is configured in non-circular mode, no DMA transaction will be generated after a transfer is completed (that is, the number of data items to be transferred has reached zero). To begin a new DMA transfer, write a new number of data items to be transferred into the DMA_TCNTx register with the DMA channel disabled.

Note: If a DMA channel is disabled, the DMA registers will not be reset. The DMA channel registers (DMA_CHCTRLx, DMA_CPBAX, and DMA_CMBAX) remain the values programmed previously.

In the circular mode, after the last transfer is completed, the DMA_TCNTx register is automatically reloaded with the initially programmed value. The current internal peripheral/memory address register is also reloaded with the initial base address programmed by the DMA_CPBAX/DMA_CMBAX registers.

Channel configuration process

DMA channel x configuration procedure is as follows (where x is the channel number):

1. Set the peripheral register address in the DMA_CPBAX register. Data will be transferred from/to this address when there is peripheral data transfer request.
2. Set the memory address in the DMA_CMBAX register. Data will be read from/written to this address when there is peripheral data transfer request.
3. Configure the amount of data to be transferred in the DMA_TCNTx register. After each data transfer, this value will be decremented.
4. Configure the channel priority by using the CHPL[1:0] bits in the DMA_CHCTRLx

register.

5. Configure data transfer direction, circular mode, peripheral/memory incremented mode, peripheral/memory data size, and interrupt after half or full transfer in the DMA_CHCTRLx register.
6. Activate the channel by setting the ENABLE bit in the DMA_CHCTRLx register.

Once the DMA channel is enabled, it can serve any DMA request from the peripheral connected to the channel.

After half of the data is transferred, the half-transfer flag (HTIF) is set. An interrupt is generated if the Half-Transfer Interrupt Enable bit (HTIE) is set. After the transfer is completed, the Transfer Complete Flag (TCIF) is set. An interrupt is generated if the Transfer Complete Interrupt Enable bit (TCIE) is set.

Circular mode

Circular mode is used to handle circular buffers and continuous data transfer (e.g. ADC scan mode). The CIRM bit in the DMA_CHCTRLx register can be used to enable this function. Once the circular mode is activated, when the amount of data to be transferred becomes 0, it will automatically be reloaded with the initial value programmed at the channel configuration phase, and DMA transaction will continue.

Memory-to-memory mode

The DMA channels can also work without requests from peripherals. This mode is called memory-to-memory mode. If the MEMTOMEM bit in the DMA_CHCTRLx register is set, DMA transfer begins as soon as software activates the DMA channel by setting the CHEN bit in the DMA_CHCTRLx register.

DMA transfer stops once the DMA_TCNTx register reaches zero. Memory-to-memory mode cannot be used with the circular mode at the same time.

9.3.4 Programmable Data Transfer Width, Alignment, and Endian

When PWIDTH is not equal to MWIDTH, DMA block performs data alignment according to Table 9-1.

Table 9-1 Programmable Data Transfer Width and Endian Behavior (When PINC = MINC = 1)

Source Port Width	Destination Port Width	Number of Data Items to Transfer	Source Content: Address/Data	Transfer Operation	Destination Content: Address/Data
8	8	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0, then write B0[7:0] at 0x0 2: Read B1[7:0] at 0x1, then write B1[7:0] at 0x1 3: Read B2[7:0] at 0x2, then write B2[7:0] at 0x2 4: Read B3[7:0] at 0x3, then write B3[7:0] at 0x3	0x0/B0 0x1/B1 0x2/B2 0x3/B3
8	16	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0, then write B0[15:0] at 0x0 2: Read B1[7:0] at 0x1, then write B1[15:0] at 0x2 3: Read B2[7:0] at 0x2, then write B2[15:0] at 0x4 4: Read B3[7:0] at 0x3, then write B3[15:0] at 0x6	0x0/00B0 0x2/00B1 0x4/00B2 0x6/00B3
8	32	4	0x0/B0 0x1/B1 0x2/B2 0x3/B3	1: Read B0[7:0] at 0x0, then write 000000B0[31:0] at 0x0 2: Read B1[7:0] at 0x1, then write 000000B1[31:0] at 0x4 3: Read B2[7:0] at 0x2, then write 000000B2[31:0] at 0x8 4: Read B3[7:0] at 0x3, then write 000000B3[31:0] at 0xC	0x0/000000B0 0x4/000000B1 0x8/000000B2 0xC/000000B3
16	8	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[15:0] at 0x0, then write B0[7:0] at 0x0 2: Read B3B2[15:0] at 0x2, then write B2[7:0] at 0x1 3: Read B5B4[15:0] at 0x4, then write B4[7:0] at 0x2 4: Read B7B6[15:0] at 0x6, then write B6[7:0] at 0x3	0x0/B0 0x1/B2 0x2/B4 0x3/B6
16	16	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[15:0] at 0x0, then write B1B0[15:0] at 0x0 2: Read B3B2[15:0] at 0x2, then write B3B2[15:0] at 0x2 3: Read B5B4[15:0] at 0x4, then write B5B4[15:0] at 0x4 4: Read B7B6[15:0] at 0x6, then write B7B6[15:0] at 0x6	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6
16	32	4	0x0/B1B0 0x2/B3B2 0x4/B5B4 0x6/B7B6	1: Read B1B0[15:0] at 0x0, then write 0000B1B0[31:0] at 0x0 2: Read B3B2[15:0] at 0x2, then write 0000B3B2[31:0] at 0x4 3: Read B5B4[15:0] at 0x4, then write 0000B5B4[31:0] at 0x8 4: Read B7B6[15:0] at 0x6, then write 0000B7B6[31:0] at 0xC	0x0/0000B1B0 0x4/0000B3B2 0x8/0000B5B4 0xC/0000B7B6

32	8	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, then write B0[7:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4, then write B4[7:0] at 0x1 3: Read BBBAB9B8[31:0] at 0x8, then write B8[7:0] at 0x2 4: Read BFBEBDBC[31:0] at 0xC, then write BC[7:0] at 0x3	0x0/B0 0x1/B4 0x2/B8 0x3/BC
32	16	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, then write B1B0[15:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4, then write B5B4[15:0] at 0x2 3: Read BBBAB9B8[31:0] at 0x8, then write B9B8[15:0] at 0x4 4: Read BFBEBDBC[31:0] at 0xC, then write BDBC[15:0] at 0x6	0x0/B1B0 0x2/B5B4 0x4/B9B8 0x6/BDBC
32	32	4	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC	1: Read B3B2B1B0[31:0] at 0x0, then write B3B2B1B0[31:0] at 0x0 2: Read B7B6B5B4[31:0] at 0x4, then write B7B6B5B4[31:0] at 0x4 3: Read BBBAB9B8[31:0] at 0x8, then write BBBAB9B8[31:0] at 0x8 4: Read BFBEBDBC[31:0] at 0xC, then write BFBEBDBC[31:0] at 0xC	0x0/B3B2B1B0 0x4/B7B6B5B4 0x8/BBBAB9B8 0xC/BFBEBDBC

Addressing an AHB peripheral that does not support byte or half-word write operations

When the DMA block initiates an AHB byte or half-word write operation, the data are duplicated on the unused lanes of the HWDATA[31:0] bus. Therefore, when DMA writes byte or half-word to the AHB peripheral that does not support byte or half-word write operations (when HSIZE is not used by the peripheral), errors will not be generated. The DMA writes the 32-bit HWDATA in the manner shown in the following two examples:

- To write the half-word “0xABCD” when HSIZE = half-word, the DMA sets the HWDATA bus to “0xABCDABCD”.
- To write the byte “0xAB” when HSIZE = byte, the DMA sets the HWDATA bus to “0xABABABAB”.

If the AHB/APB bridge is an AHB 32-bit slave peripheral that does not take the HSIZE data into account, it will transform any AHB byte or half-word operation into 32-bit APB operation in the following manner:

- An AHB byte write operation of the data “0xB0” to 0x0 (or to 0x1, 0x2 or 0x3) will be converted to an APB word write operation of the data “0xB0B0B0B0” to 0x0.
- An AHB half-word write operation of the data “0xB1B0” to 0x0 (or to 0x2) will be converted to an APB word write operation of the data “0xB1B0B1B0” to 0x0.

For instance, to write the APB backup registers (16-bit registers aligned to a 32-bit address boundary), the memory data source width (MWIDTH) must be configured to “16 bits”, and the peripheral data destination width (PWIDTH) to “32 bits”.

9.3.5 Error Management

A DMA transfer error can be generated when reading from or writing to a reserved address space. When a DMA transfer error occurs during a DMA read or a write access, the CHEN bit of in the channel configuration register (DMA_CHCTRLx) corresponding to the faulty channel will be cleared automatically by hardware, and the operation on the channel is disabled. At this time, the transfer error interrupt flag bit (ERRIF) corresponding to the channel in the DMA_IFR register is set. An interrupt is generated if the transfer error interrupt enable bit in the DMA_CHCTRLx register is set.

9.3.6 Interrupts

An interrupt can be generated on a DMA half transfer, transfer complete, or transfer error form each DMA channel. For the purpose of flexibility, interrupts are enabled by configuring different bits in the registers.

Table 9-2 DMA Interrupt Request

Interrupt Event	Event Flag Bit	Enable Control Bit
Half transfer	HTIF	HTIE
Transfer completed	TCIF	TCIE
Transfer error	ERRIF	ERRIE

Note: DMA2 channel 4 and DMA2 channel 5 interrupts are mapped onto the same interrupt vector.

9.3.7 DMA Request Mapping

DMA1 Controller

The 7 requests from the peripherals (TMRx[1,2,3,4], ADC1, SPI1, SPI/I2S2, I²Cx[1,2], and USARTx[1,2,3]) are logically ORed before entering the DMA1 controller, and this means that only one request can be enabled at a time. Please refer to Figure 9-2 for DMA1 request mapping.

The peripheral DMA requests can be independently activated/de-activated by programming the control bit in the corresponding peripheral registers.

Figure 9-2 DMA1 Request Mapping

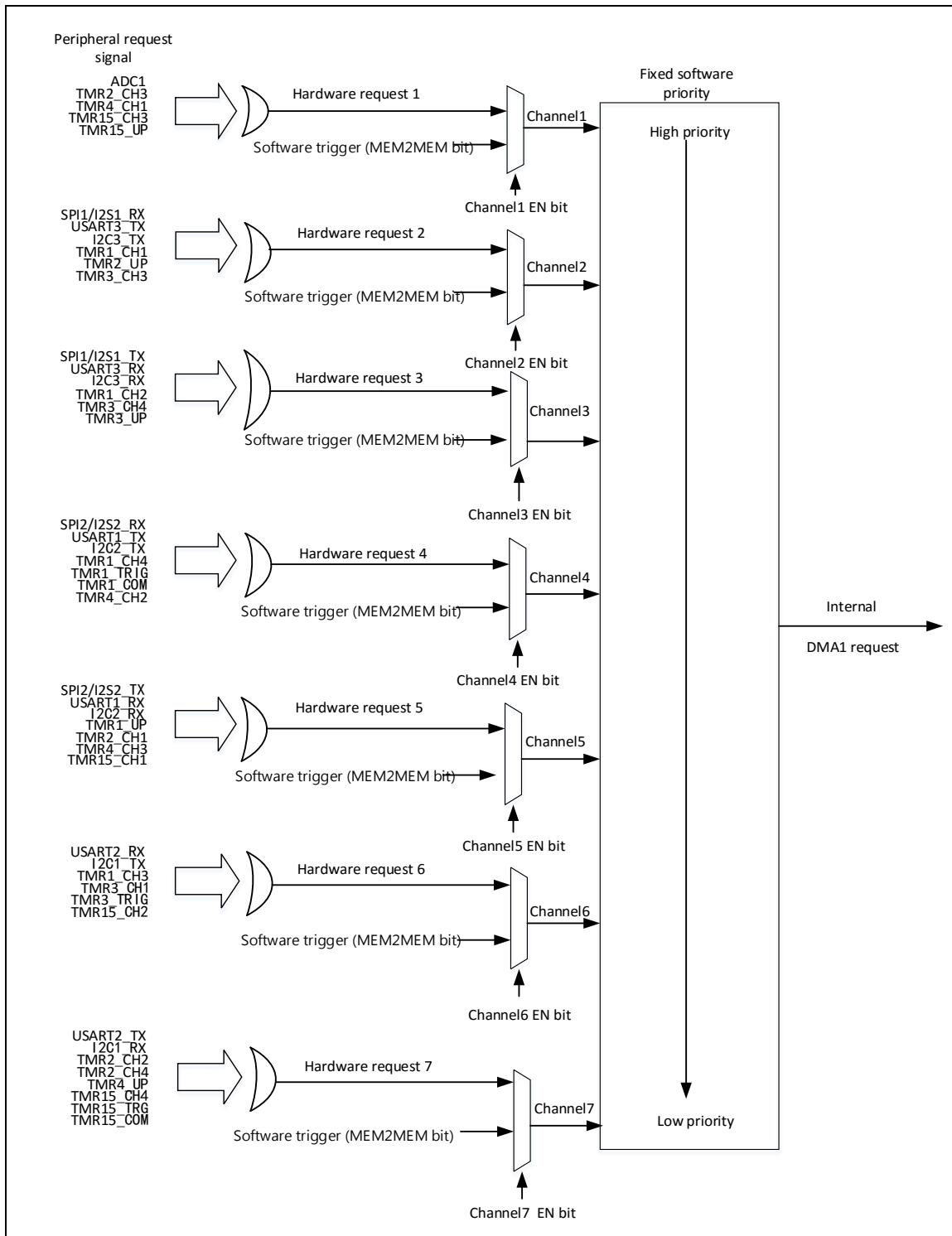


Table 9-3 Summary of DMA1 Requests for Each Channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5	Channel 6	Channel 7
ADC1	ADC1						
SPI/I ² S		SPI1/I2S1_RX	SPI1/I2S1_TX	SPI2/I2S2_RX	SPI2/I2S2_TX		
USART		USART3_TX	USART3_RX	USART1_TX	USART1_RX	USART2_RX	USART2_TX
I ² C		I2C3_TX	I2C3_RX	I2C2_TX	I2C2_RX	I2C1_TX	I2C1_RX
TMR1		TMR1_CH1	TMR1_CH2	TMR1_CH4 TMR1_TRIG TMR1_COM	TMR1_UP	TMR1_CH3	
TMR2	TMR2_CH3	TMR2_UP			TMR2_CH1		TMR2_CH2 TMR2_CH4
TMR3		TMR3_CH3	TMR3_CH4 TMR3_UP			TMR3_CH1 TMR3_TRIG	
TMR4	TMR4_CH1			TMR4_CH2	TMR4_CH3		TMR4_UP
TMR15	TMR15_CH3 TMR15_UP				TMR15_CH1	TMR15_CH2	TMR15_CH4 TMR15_TRG TMR15_COM

DMA2 Controller

The 5 requests from the peripherals (TMRx[5, 6, 7, 8], ADC3, SPI/I2S3, UART4, DAC_Channel 1, 2, and SDIO) are logically ORed and sent to the DMA2, and this means that only one request can be enabled at a time. Please refer to Figure 9-3 for DMA2 requests mapping.

The peripheral DMA requests can be independently activated/de-activated by programming the control bit in the corresponding peripheral registers.

Figure 9-3 DMA2 Request Mapping

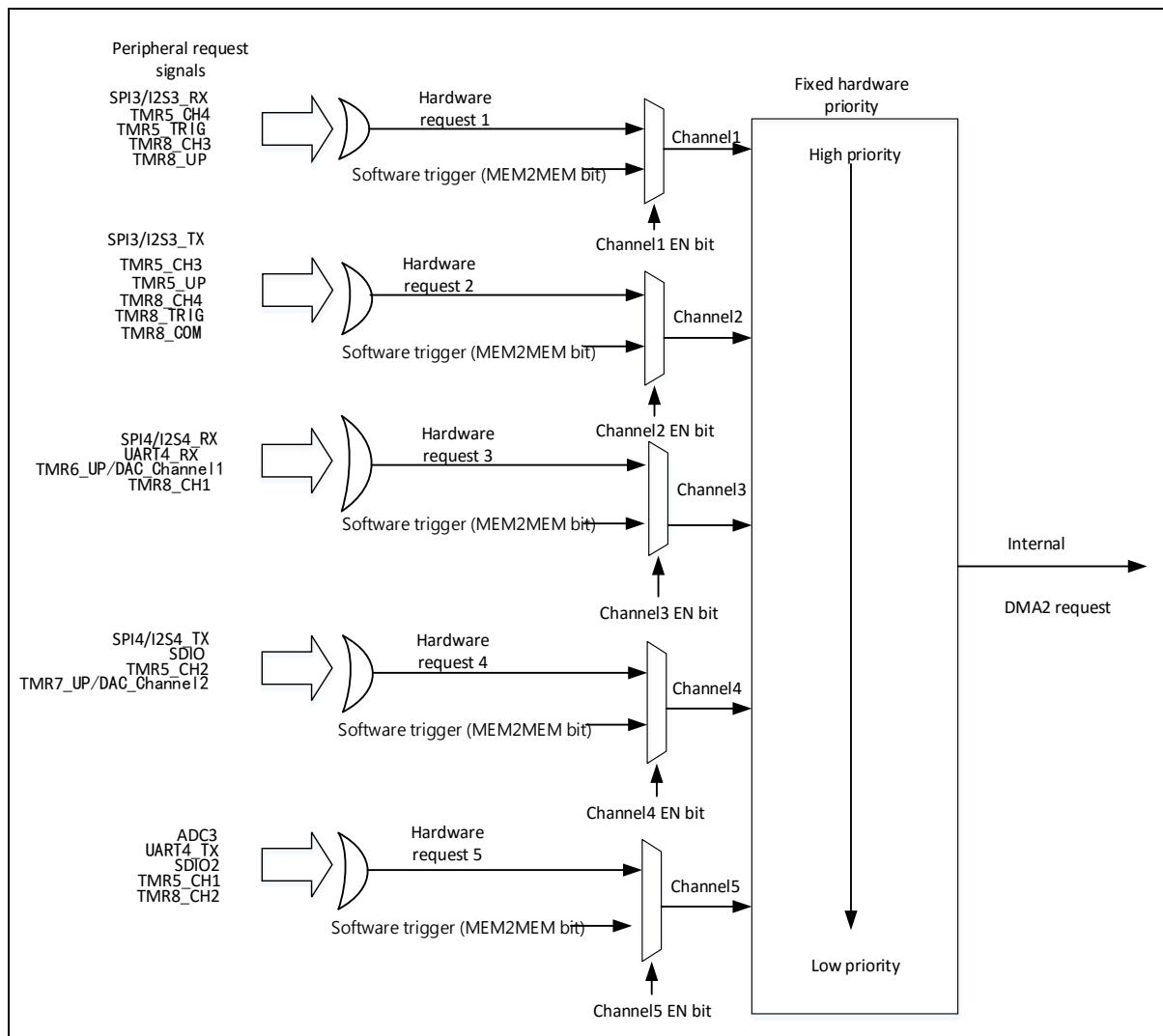


Table 9-4 Summary of DMA2 Requests for Each Channel

Peripheral	Channel 1	Channel 2	Channel 3	Channel 4	Channel 5
ADC3					ADC3
SPI/I2S	SPI3/I2S3_RX	SPI3/I2S3_TX	SPI4/I2S4_RX	SPI4/I2S4_TX	
UART4			UART4_RX		UART4_TX
SDIO				SDIO	
SDIO2					SDIO2
TMR5	TMR5_CH4 TMR5_TRIG	TMR5_CH3 TMR5_UP		TMR5_CH2	TMR5_CH1
TMR6/ DAC channel 1			TMR6_UP/ DAC channel 1		
TMR7/DAC channel 2				TMR7_UP/ DAC channel 2	
TMR8	TMR8_CH3 TMR8_UP	TMR8_CH4 TMR8_TRIG TMR8_COM	TMR8_CH1		TMR8_CH2

9.4 DMA Registers

The peripheral registers can be accessed by bytes (8-bit), half-words (16-bit) or words (32-bit).

Note: In the following registers, all bits related to channel6 and channel7 are not relevant for DMA2 since it has only 5 channels.

Table 9-5 DMA Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																	
000h	DMA_ISTS	Reserved	ERRIF7	27	HTIF7	26	TCIF7	25	GIF7	24	HTIF6	23	TCIF6	22	GIF5	21	TCIF5	20	GIF6	19	TCIF4	18	GIF4	17	TCIF3	16	GIF3	15	TCIF2	14	GIF2	13	TCIF1	12	GIF1	11	TCIF0	10	GIF0	9	TCIF8	8	GIF8	7	TCIF9	6	GIF9	5	TCIF10	4	GIF10	3	TCIF11	2	GIF11	1	TCIF12	0	GIF12	0	TCIF13	0	GIF13	0	TCIF14	0	GIF14	0	TCIF15	0	GIF15	0	TCIF16	0	GIF16	0	TCIF17	0	GIF17	0	TCIF18	0	GIF18	0	TCIF19	0	GIF19	0	TCIF20	0	GIF20	0	TCIF21	0	GIF21	0	TCIF22	0	GIF22	0	TCIF23	0	GIF23	0	TCIF24	0	GIF24	0	TCIF25	0	GIF25	0	TCIF26	0	GIF26	0	TCIF27	0	GIF27	0	TCIF28	0	GIF28	0	TCIF29	0	GIF29	0	TCIF30	0	GIF30	0	TCIF31	0	GIF31	0	TCIF32	0	GIF32	0	TCIF33	0	GIF33	0	TCIF34	0	GIF34	0	TCIF35	0	GIF35	0	TCIF36	0	GIF36	0	TCIF37	0	GIF37	0	TCIF38	0	GIF38	0	TCIF39	0	GIF39	0	TCIF40	0	GIF40	0	TCIF41	0	GIF41	0	TCIF42	0	GIF42	0	TCIF43	0	GIF43	0	TCIF44	0	GIF44	0	TCIF45	0	GIF45	0	TCIF46	0	GIF46	0	TCIF47	0	GIF47	0	TCIF48	0	GIF48	0	TCIF49	0	GIF49	0	TCIF50	0	GIF50	0	TCIF51	0	GIF51	0	TCIF52	0	GIF52	0	TCIF53	0	GIF53	0	TCIF54	0	GIF54	0	TCIF55	0	GIF55	0	TCIF56	0	GIF56	0	TCIF57	0	GIF57	0	TCIF58	0	GIF58	0	TCIF59	0	GIF59	0	TCIF60	0	GIF60	0	TCIF61	0	GIF61	0	TCIF62	0	GIF62	0	TCIF63	0	GIF63	0	TCIF64	0	GIF64	0	TCIF65	0	GIF65	0	TCIF66	0	GIF66	0	TCIF67	0	GIF67	0	TCIF68	0	GIF68	0	TCIF69	0	GIF69	0	TCIF70	0	GIF70	0	TCIF71	0	GIF71	0	TCIF72	0	GIF72	0	TCIF73	0	GIF73	0	TCIF74	0	GIF74	0	TCIF75	0	GIF75	0	TCIF76	0	GIF76	0	TCIF77	0	GIF77	0	TCIF78	0	GIF78	0	TCIF79	0	GIF79	0	TCIF80	0	GIF80	0	TCIF81	0	GIF81	0	TCIF82	0	GIF82	0	TCIF83	0	GIF83	0	TCIF84	0	GIF84	0	TCIF85	0	GIF85	0	TCIF86	0	GIF86	0	TCIF87	0	GIF87	0	TCIF88	0	GIF88	0	TCIF89	0	GIF89	0	TCIF90	0	GIF90	0	TCIF91	0	GIF91	0	TCIF92	0	GIF92	0	TCIF93	0	GIF93	0	TCIF94	0	GIF94	0	TCIF95	0	GIF95	0	TCIF96	0	GIF96	0	TCIF97	0	GIF97	0	TCIF98	0	GIF98	0	TCIF99	0	GIF99	0	TCIF100	0	GIF100	0	TCIF101	0	GIF101	0	TCIF102	0	GIF102	0	TCIF103	0	GIF103	0	TCIF104	0	GIF104	0	TCIF105	0	GIF105	0	TCIF106	0	GIF106	0	TCIF107	0	GIF107	0	TCIF108	0	GIF108	0	TCIF109	0	GIF109	0	TCIF110	0	GIF109	0	TCIF111	0	GIF111	0	TCIF112	0	GIF112	0	TCIF113	0	GIF113	0	TCIF114	0	GIF114	0	TCIF115	0	GIF115	0	TCIF116	0	GIF116	0	TCIF117	0	GIF117	0	TCIF118	0	GIF118	0	TCIF119	0	GIF119	0	TCIF120	0	GIF120	0	TCIF121	0	GIF121	0	TCIF122	0	GIF122	0	TCIF123	0	GIF123	0	TCIF124	0	GIF124	0	TCIF125	0	GIF125	0	TCIF126	0	GIF126	0	TCIF127	0	GIF127	0	TCIF128	0	GIF128	0	TCIF129	0	GIF129	0	TCIF130	0	GIF130	0	TCIF131	0	GIF131	0	TCIF132	0	GIF132	0	TCIF133	0	GIF133	0	TCIF134	0	GIF134	0	TCIF135	0	GIF135	0	TCIF136	0	GIF136	0	TCIF137	0	GIF137	0	TCIF138	0	GIF138	0	TCIF139	0	GIF139	0	TCIF140	0	GIF140	0	TCIF141	0	GIF141	0	TCIF142	0

038h	DMA_CPBAA3	PA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
03Ch	DMA_CMBAA3	MA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
040h		Reserved									
044h	DMA_CHCTRL4	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
048h	DMA_TCNT 4	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
04Ch	DMA_CPBAA4	PA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
050h	DMA_CMBAA4	MA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
054h		Reserved									
058h	DMA_CHCTRL5	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
05Ch	DMA_TCNT 5	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
060h	DMA_CPBAA5	PA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
064h	DMA_CMBAA5	MA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
068h		Reserved									
06Ch	DMA_CHCTRL6	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
070h	DMA_TCNT 6	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
074h	DMA_CPBAA6	PA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
078h	DMA_CMBAA6	MA[31:0]									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
07Ch		Reserved									
080h	DMA_CHCTRL7	Reserved									
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0									
084h	DMA_TCNT 7	Reserved									
		CNT[15:0]									

	Reset Value	0 0
088h	DMA_CPB _{A7}	PA[31:0]
	Reset Value	0 0
08Ch	DMA_CMBA ₇	MA[31:0]
	Reset Value	0 0
090h		Reserved

9.4.1 DMA Interrupt Status Register (DMA_ISTS)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
					ERR IF7	HTI F7	TCI F7	GI F7	ERR IF6	HT IF6	TC IF6	GI F6	ERR IF5	HT IF5	TCI F5	GI F5
15	14	13	12	r	r	r	r	r	r	r	r	r	r	r	r	r
				11	10	9	8	7	6	5	4	3	2	1	0	
ERR IF4	HT IF4	TC IF4	GI F4	ER RIF3	HT IF3	TC IF3	GI F3	ERR IF2	HT IF2	TC IF2	GI F2	ER RIF1	HT IF1	TC IF1	GI F1	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bit 31:28	Reserved. Always read as 0.
Bit 27, 23, 19, 15, 11, 7, 3	ERRIF_x: Channel x transfer error flag ($x = 1 \dots 7$) These bits are set by hardware. Writing '1' to the corresponding bit of DMA_ICLR register can clear the corresponding flag bit here. 0: No transfer error (TE) occurred on channel x. 1: Transfer error (TE) occurred on channel x.
Bit 26, 22, 18, 14, 10, 6, 2	HTIF_x: Channel x half transfer flag ($x = 1 \dots 7$) These bits are set by hardware. Writing '1' to the corresponding bit of DMA_ICLR register can clear the corresponding flag bit here. 0: No half transfer (HT) event occurred on channel x. 1: Half transfer (HT) event occurred on channel x.
Bit 25, 21, 17, 13, 9, 5, 1	TCIF_x: Channel x transfer complete flag ($x = 1 \dots 7$) These bits are set by hardware. Writing '1' to the corresponding bit of DMA_ICLR register can clear the corresponding flag bit here. 0: No transfer complete (TC) event occurred on channel x. 1: Transfer complete (TC) event occurred on channel x.
Bit 24, 20, 16, 12, 8, 4, 0	GIF_x: Channel x global interrupt flag ($x = 1 \dots 7$) These bits are set by hardware. Writing '1' to the corresponding bit of DMA_ICLR register can clear the corresponding flag bit here. 0: No TE, HT, or TC events occurred on channel x. 1: TE, HT, or TC events occurred on channel x.

9.4.2 DMA Interrupt Flag Clear Register (DMA_ICLR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
					CERRI F7	CHTI F7	CTCI F7	CGI F7	CERRI F6	CHTI F6	CTCI F6	CGI F6	CERRI F5	CHTI F5	CTCI F5	CGI F5
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CERRI F4	CHTI F4	CTCI F4	CGI F4	CERRI F3	CHTI F3	CTCI F3	CGI F3	CERRI F2	CHTI F2	CTCI F2	CGI F2	CERRI F1	CHTI F1	CTCI F1	CGI F1	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 31:28	Reserved. Always read as 0.
-----------	-----------------------------

Bit 27, 23, 19, 15, 11, 7, 3	CERRIF_x : Channel x transfer error clear ($x = 1 \dots 7$) These bits are set and cleared by software. 0: No effect 1: Clear the corresponding ERRIF flag in the DMA_ISTS register
Bit 26, 22, 18, 14, 10, 6, 2	CHTIF_x : Channel x half transfer clear ($x = 1 \dots 7$) These bits are set and cleared by software. 0: No effect 1: Clear the corresponding HTIF flag in the DMA_ISTS register
Bit 25, 21, 17, 13, 9, 5, 1	CTCIF_x : Channel x transfer complete clear ($x = 1 \dots 7$) These bits are set and cleared by software. 0: No effect 1: Clear the corresponding TCIF flag in the DMA_ISTS register
Bit 24, 20, 16, 12, 8, 4, 0	CGIF_x : Channel x global interrupt clear ($x = 1 \dots 7$) These bits are set and cleared by software. 0: No effect 1: Clear the corresponding GIF, ERRIF, HTIF, and TCIF flags in the DMA_ISTS register

9.4.3 DMA Channel x Configuration Register (DMA_CHCTRL_x) ($x = 1 \dots 7$)

Address offset: 0x08 + 20 x (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	MEM TO MEM	CHPL[1:00]	MWIDTH [1:0]	PWIDTH [1:0]	MIN C	PINC	CIR M	DIR	ERRI E	HTIE	TCIE	CHE N			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:15	Reserved. Always read as 0.
Bit 14	MEMTOMEM : Memory-to-memory mode This bit is set and cleared by software. 0: Memory-to-memory mode is disabled. 1: Memory-to-memory mode is enabled.
Bit 13:12	CHPL[1:0] : Channel priority level These bits are set and cleared by software. 00: Low 01: Medium 10: High 11: Very high
Bit 11:10	MWIDTH[1:0] : Memory size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved
Bit 9:8	PWIDTH[1:0] : Peripheral size These bits are set and cleared by software. 00: 8 bits 01: 16 bits 10: 32 bits 11: Reserved

Bit 7	MINC: Memory increment mode This bit is set and cleared by software. 0: Memory increment mode is disabled. 1: Memory increment mode is enabled.
Bit 6	PINC: Peripheral increment mode This bit is set and cleared by software. 0: Peripheral increment mode is disabled. 1: Peripheral increment mode is enabled.
Bit 5	CIRM: Circular mode This bit is set and cleared by software. 0: Circular mode is disabled. 1: Circular mode is enabled.
Bit 4	DIR: Data transfer direction This bit is set and cleared by software. 0: Read from peripheral 1: Read from memory
Bit 3	ERRIE: Transfer error interrupt enable This bit is set and cleared by software. 0: TE interrupt is disabled. 1: TE interrupt is enabled.
Bit 2	HTIE: Half transfer interrupt enable This bit is set and cleared by software. 0: HT interrupt is disabled. 1: HT interrupts is enabled.
Bit 1	TCIE: Transfer complete interrupt enable This bit is set and cleared by software. 0: TC interrupt is disabled. 1: TC interrupt is enabled.
Bit 0	CHEN: Channel enable This bit is set and cleared by software. 0: Channel is disabled. 1: Channe is enabled.

9.4.4 DMA Channel x Number of Data Register (DMA_TCNTx) (x = 1 ... 7)

Address offset: 0x0C + 20 x (channel number – 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CNT[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:16	Reserved. Always read as 0.															
Bit 15:0	CNT[15:0]: Number of data to transfer The number of data to be transferred is from 0 up to 65535. This register can only be written when the channel is disabled (CHEN = 0 in the DMA_CHCTRLx register). Once the channel is enabled, this register is read-only, indicating how many transfers remain. This register decrements after each DMA transfer. After the transfer is completed, this register becomes '0'; if the channel is configured in auto-reload mode, the register will be reloaded automatically by the value previously programmed. If this register is '0', no data transaction will be proceeded no matter the channel is enabled or not.															

9.4.5 DMA Channel x Peripheral Address Register (DMA_CPB_{Ax}) ($x = 1 \dots 7$)

Address offset: 0x10 + 20 x (channel number – 1)

Reset value: 0x0000 0000

When the channel is enabled (CHEN = 1 in the DMA_CHCTRL_x register), this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PA[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PA[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:0		PA[31:0]: Peripheral address Base address of peripheral data register is the source or destination for data transfer. When PWIDTH = '01' (16 bits), the PA[0] bit is ignored. Access is automatically aligned to a half-word address. When PWIDTH = '10' (32 bits), PA[1:0] are ignored. Access is automatically aligned to a word address.													

9.4.6 DMA Channel x Memory Address Register (DMA_CMB_{Ax}) ($x = 1 \dots 7$)

Address offset: 0x14 + 20 x (channel number – 1)

Reset value: 0x0000 0000

When the channel is enabled (CHEN = 1 in the DMA_CHCTRL_x register), this register cannot be written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MA[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MA[31:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:0		MA[31:0]: Memory address Memory address is the source or destination for data transfer. When MWIDTH = '01' (16 bits), the MA[0] bit is ignored. Access is automatically aligned to a half-word address. When MWIDTH = '10' (32 bits), MA[1:0] are ignored. Access is automatically aligned to a word address.													

10 Timer

AT32F403 timers include basic timers, general-purpose timers, and advanced-control timers. Please refer to Section 10.1 ~ Section 10.4 for the detailed function modes. All functions of different timers are shown in the following tables.

Timer Type	Timer	Counter Bit	Count mode	Repetition	Prescaler Ratio	DMA Requests	Capture/compare Channel	PWM Input Mode	ETR Input	Break Input
Advanced-control timer	TMR1 TMR8 TMR15	16	Up Down Up/Down	8-bit	1 ~ 65536	O	4	O	O	O
General-purpose timer	TMR2 TMR5	16/32	Up Down Up/Down	X	1 ~ 65536	O	4	O	O	X
	TMR3 TMR4	16	Up Down Up/Down	X	1 ~ 65536	O	4	O	O	X
	TMR9 TMR12	16	Up	X	1 ~ 65536	X	2	O	X	X
	TMR10 TMR11 TMR13 TMR14	16	Up	X	1 ~ 65536	X	1	X	X	X
Basic timer	TMR6 TMR7	16	Up	X	1 ~ 65536	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 TMR15	16	Up Down Up/Down	O	O	O	O	O	O	Timer synchronization
General-purpose timer	TMR2 TMR5	16/32	Up Down Up/Down	O	O	X	X	O	O	Timer synchronization
	TMR3 TMR4	16	Up Down Up/Down	O	O	X	X	O	O	Timer synchronization
	TMR9 TMR12	16	Up	O	O	X	X	X	X	Timer synchronization ADC/DAC
	TMR10 TMR11 TMR13 TMR14	16	Up	O	O	X	X	X	X	X
Basic timer	TMR6 TMR7	16	Up	X	X	X	X	X	X	DAC

10.1 Basic Timer (TMR6 and TMR7)

10.1.1 TMR6 and TMR7 Introduction

Basic timers TMR6 and TMR7 consist of a 16-bit auto-reload counter driven by their own programmable prescaler, respectively. They can be used as general-purpose timers to provide time base, and are especially used to provide clocks for the digital-to-analog converter (DAC). In fact, the timers are internally connected to DAC and can drive DAC directly through the trigger outputs.

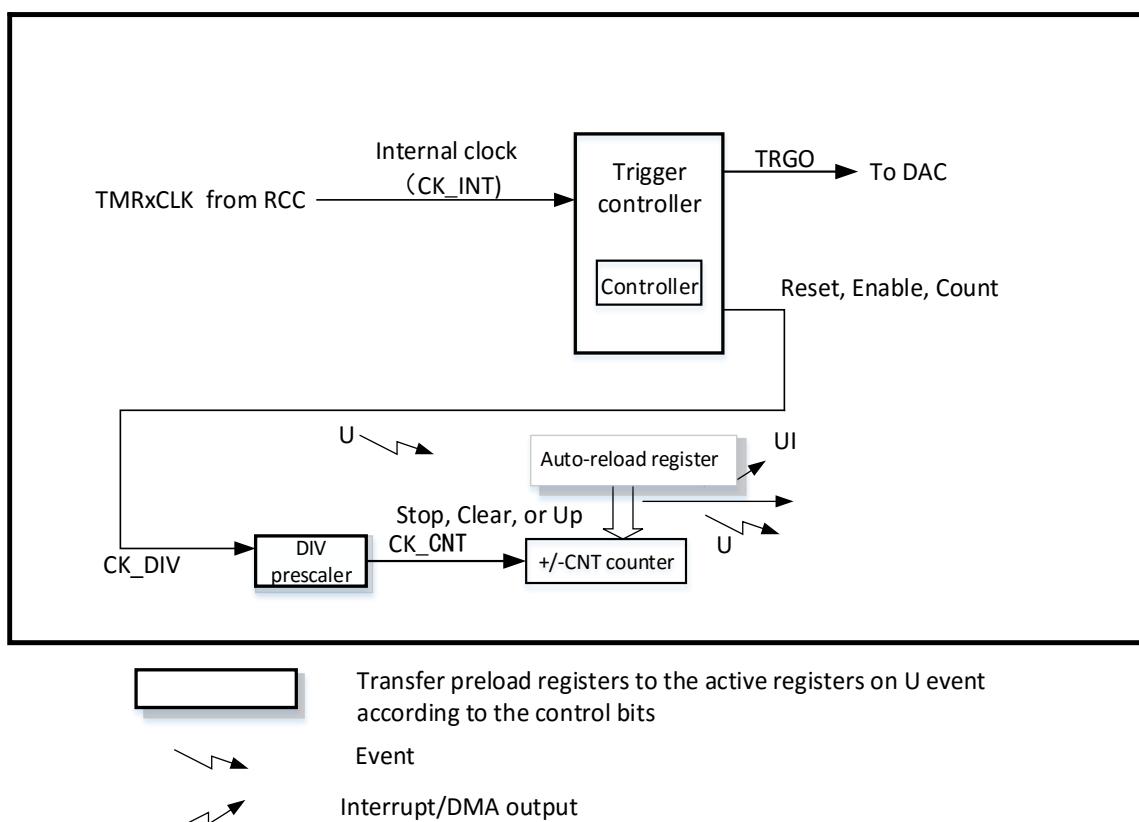
The two timers are completely independent with each other and do not share any resources.

10.1.2 TMR6 and TMR7 Main Features

The main features of TMR6 and TMR7 include:

- 16-bit auto-reload up counter
- 16-bit programmable prescaler (can be modified on the fly) used to divide the input clock frequency by any factor between 1 ~ 65536
- Synchronization circuit to trigger DAC
- Interrupt/DMA request on update event (counter flow)

Figure 10-1 Basic Timer Block Diagram



10.1.3 TMR6 and TMR7 Function Overview

10.1.3.1 Time-base Unit

This programmable timer mainly consists of a 16-bit up counter with auto-reload function. The counter clock is obtained through a prescaler.

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

The time-base unit includes:

- Counter register (TMRx_CNT)
- Prescaler register (TMRx_DIV)

- Auto-reload register (TMRx_AR)

The auto-reload register is preloaded. Each read or write to the auto-reload register is achieved by writing or reading the preload register. The contents written to the preload register are transferred into its shadow register immediately or at each update event according to the auto-reload preload enable bit (ARPEN) in the TMRx_CTRL1 register. When the UEVDIS bit in the TMRx_CTRL1 register is '0', update events will be generated by hardware once the counter reaches the overflow value. Update events can also be generated by software; please refer to the following sections for the detailed introduction on generation of the update events.

The counter is driven by the prescaler output, CK_CNT, and it is enabled when the counter enable bit (CNTEN) in the TMRx_CTRL1 register is set.

Note: The actual counter enable signal, CNT_EN, is set 1 clock cycle after CNTEN is set.

10.1.3.2 Prescaler

The prescaler can divide the counter clock frequency by any factor between 1 ~ 65536, and this is achieved with the counter of a 16-bit register (TMRx_DIV). Its value can be changed on the fly since the TMRx_DIV control register is buffered. The new prescaler ratio takes effect at the next update event.

Figure 10-2 and Figure 10-3 give examples of on-the-fly prescaler ratio change.

Figure 10-2 Counter Timing Diagram with Prescaler Division Changing from 1 to 2

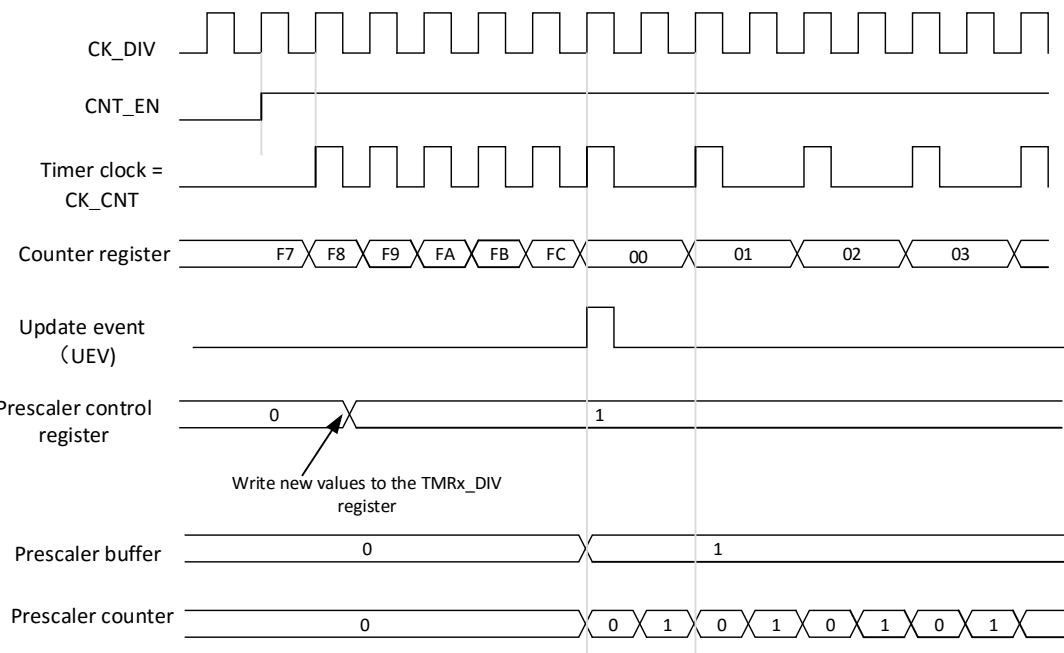
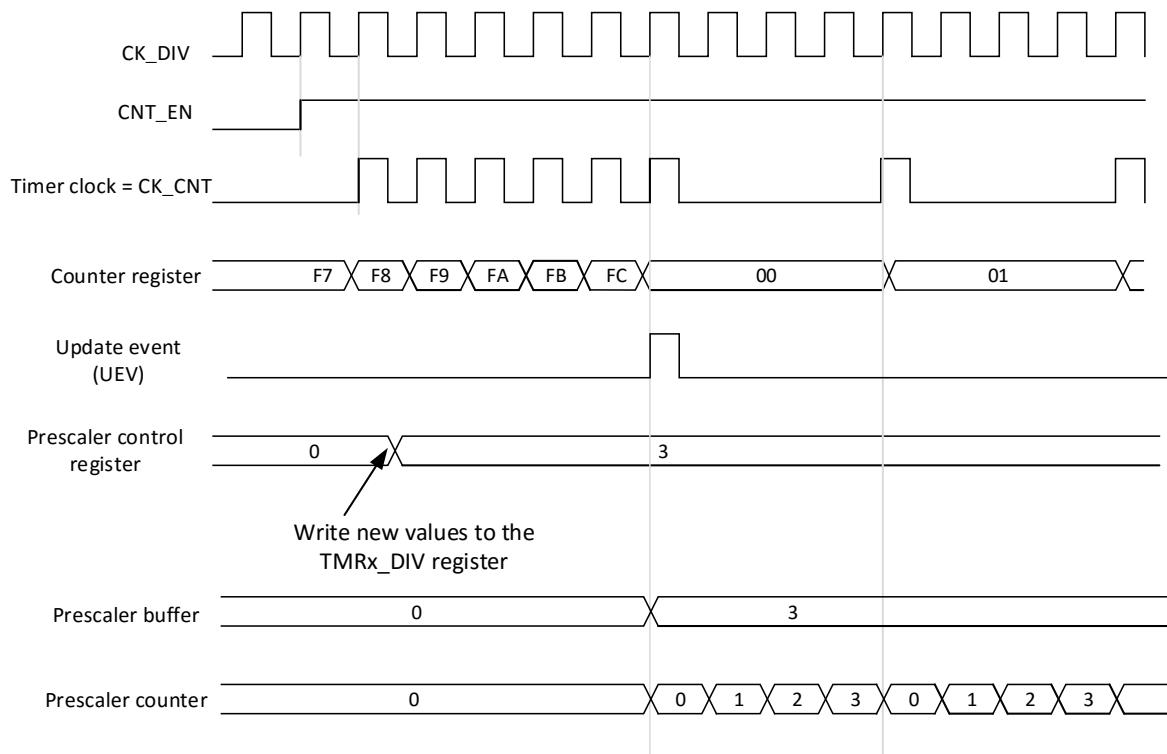


Figure 10-3 Counter Timing Diagram with Prescaler Division Changing from 1 to 4



10.1.3.3 Counting Mode

The counter counts from 0 to the auto-reload value (TMRx_AR register), and restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow; setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller) also generates the update events.

UEV events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing to the preload registers. No update event occurs until the UEVDIS bit is cleared. However, the counter and the prescaler still restart counting from 0 (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set (that is, no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated (according to the UEVRS bit), and the update flag is set (the UEVIF bit in the TMRx_STS register):

- The buffer of the prescaler is reloaded with the preload value (contents of the TMRx_DIV register).
- The auto-reload shadow register is updated with the preload value (TMRx_AR).

Figure 10-4 ~ 10-9 show examples of the counter behavior at different clock frequencies when TMRx_AR = 0x36.

Figure 10-4 Counter Timing Diagram with Internal Clock Divided by 1

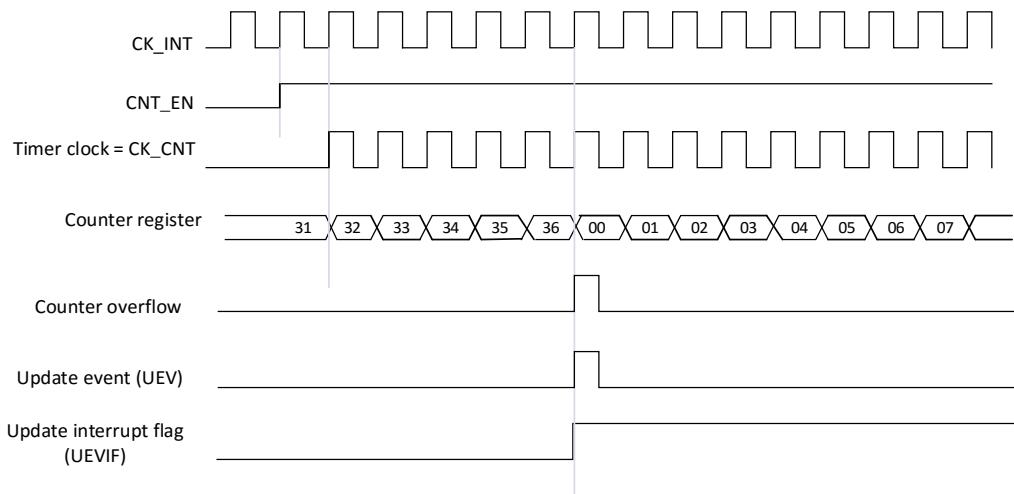


Figure 10-5 Counter Timing Diagram with Internal Clock Divided by 2

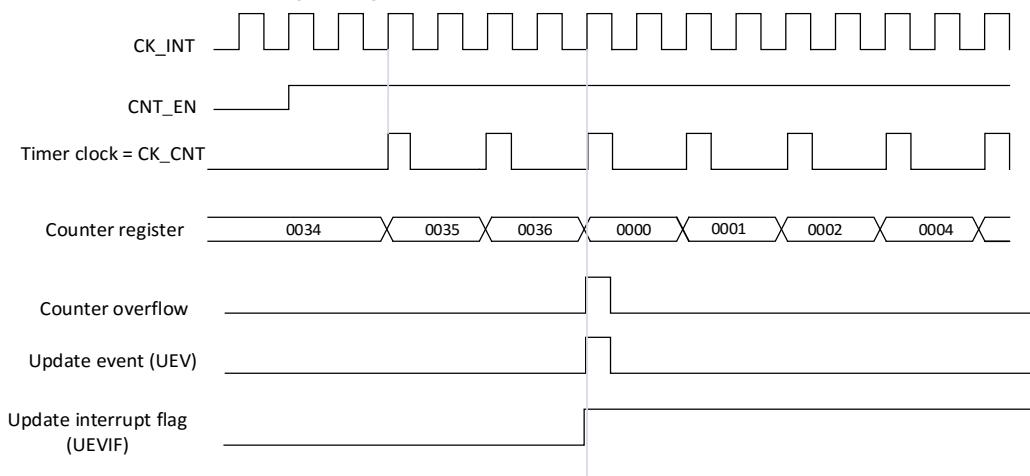


Figure 10-6 Counter Timing Diagram with Internal Clock Divided by 4

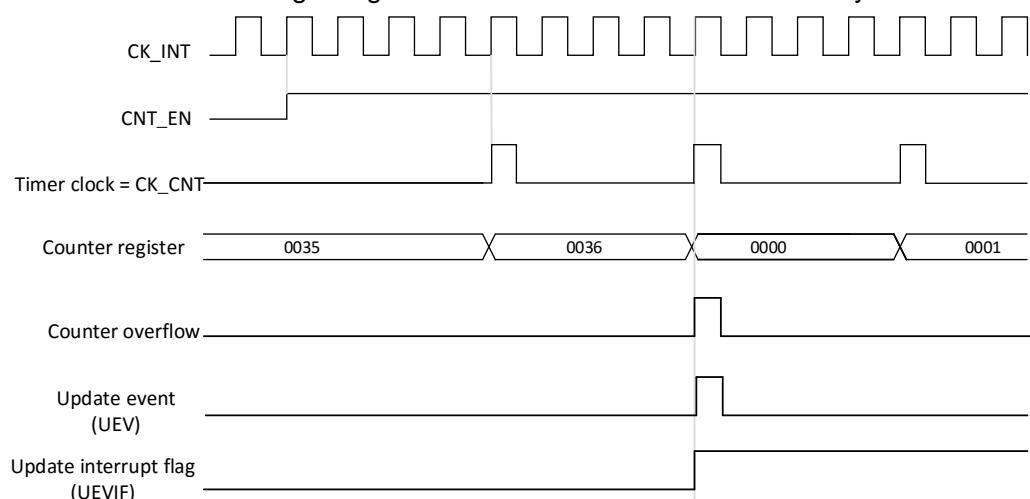


Figure 10-7 Counter Timing Diagram with Internal Clock Divided by N

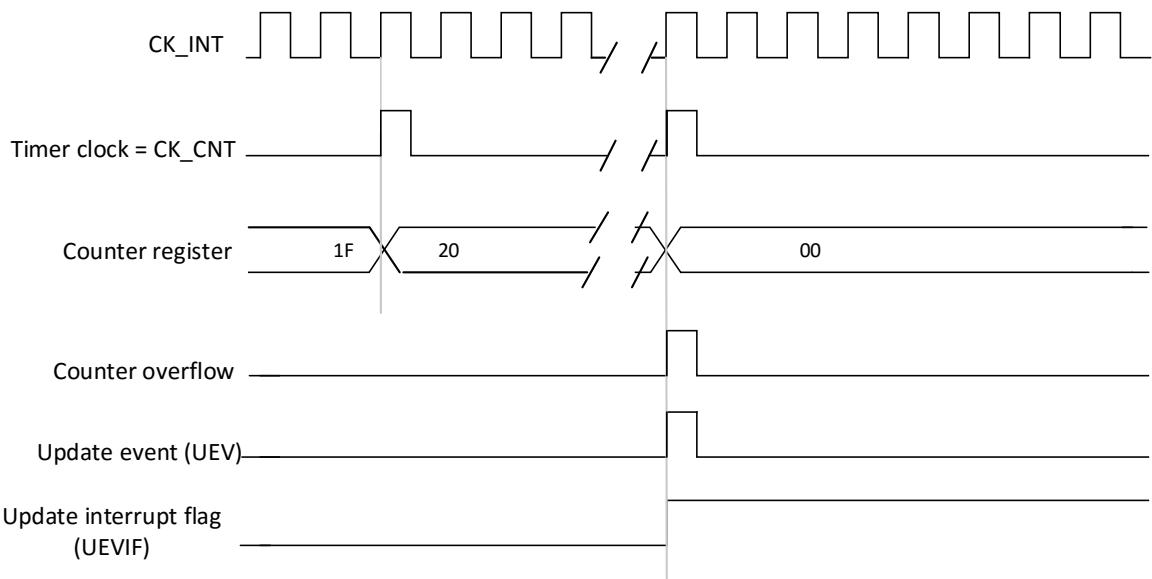


Figure 10-8 Counter Timing Diagram with Update Event When ARPEN = 0 (TMRx_AR Not Preloaded)

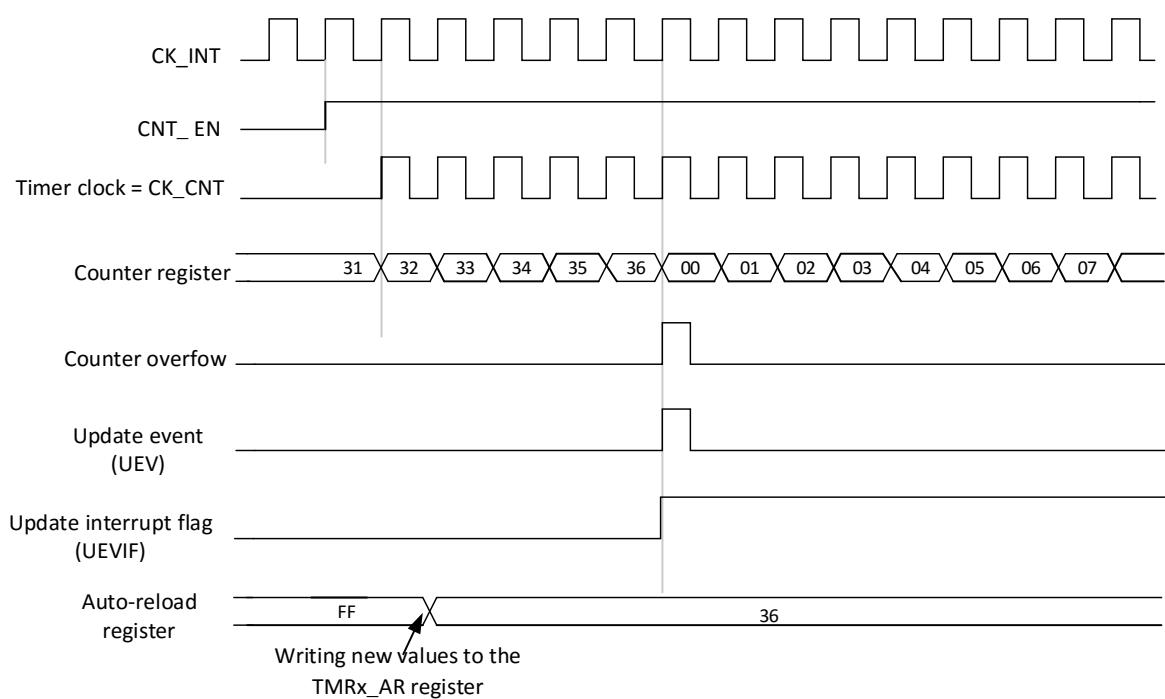
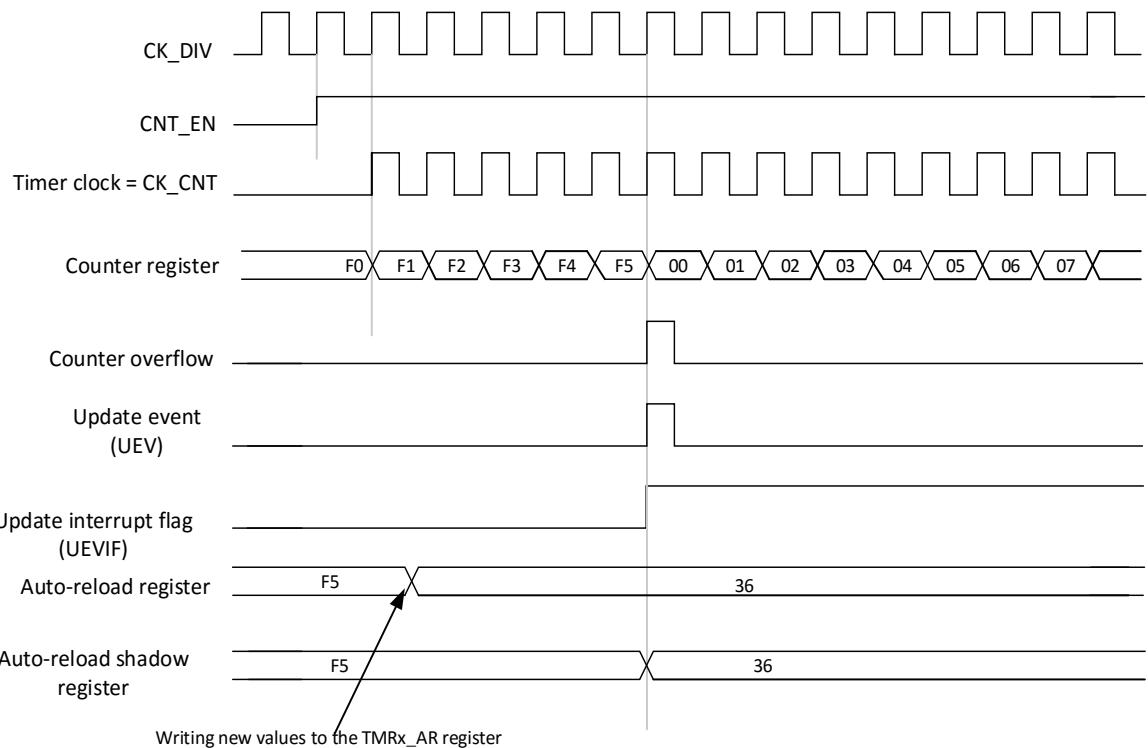


Figure 10-9 Counter Timing Diagram with Update Event When ARPEN = 1 (TMRx_AR is Preloaded)



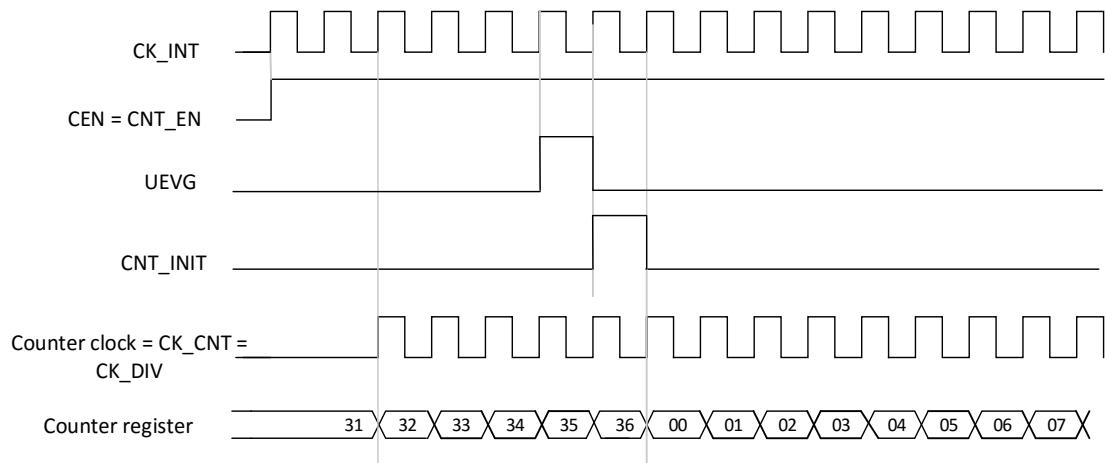
10.1.3.4 Clock Source

The counter clock is provided by the internal clock (CK_INT) source.

The CNTEN bit in the TMRx_CTRL1 register and the UEVG bit in the TMRx_EVEG register are the actual control bits, which can be changed only by software (except for the UEVG bit, which is cleared automatically). Once the CNTEN bit is written '1', the prescaler is clocked by the internal clock.

Figure 10-10 shows the behavior of the control circuit and the up counter in normal mode without prescaler.

Figure 10-10 Normal Mode Timing Diagram with Internal Clock Divided by 1



10.1.3.5 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter either continues to work or stops, depending on the configuration bit, DBG_TMRx_STOP, in the DBG module. For more details, please refer to [Section 22.2.2](#).

10.1.4 TMR6 and TMR7 Registers

These peripheral registers can be accessed by half-words (16-bit) or words (32-bit). In Table 10-1, all the TMRx registers are mapped to a 16-bit addressable space.

Table 10-1 TMR6 and TMR7 – Register Table and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TMRx_CTRL1	Reserved																				ARPEN	Reserved			UEVRS	UEVDIS	CNTEN	0				
	Reset Value																																
0x04	TMRx_CTRL2	Reserved																				MMSEL[2:0]	Reserved			UEVDE	UEVIE	0					
	Reset Value																																
0x0C	TMRx_DIE	Reserved																				UEVDE	Reserved			UEVIF	UEVIFG	0					
	Reset Value																																
0x10	TMRx_STS	Reserved																				UEVDE	Reserved			UEVIF	UEVIFG	0					
	Reset Value																																
0x14	TMRx_EVEG	Reserved																				UEVDE	Reserved			UEVIF	UEVIFG	0					
	Reset Value																																
0x24	TMRx_CNT	Reserved										CNT[15:0]										0	CNT[15:0]			UEVIF	UEVIFG	0					
	Reset Value																																
0x28	TMRx_DIV	Reserved										DIV[15:0]										0	DIV[15:0]			UEVIF	UEVIFG	0					
	Reset Value																																
0x2C	TMRx_AR	Reserved										AR[15:0]										0	AR[15:0]			UEVIF	UEVIFG	0					
	Reset Value																																

10.1.4.1 TMR6 and TMR7 Control Register 1 (TMRx_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved										ARPEN	Reserved			OPMODE	UEVRS	UEVDIS	CNTEN
										res	rw			rw	rw	rw	rw

Bit 15:8	Reserved. Always read as 0.
Bit 7	ARPEN: Auto-reload preload enable 0: TMRx_AR register is not buffered. 1: TMRx_AR register is buffered.
Bit 6:4	Reserved. Always read as 0.

Bit 3	OPMODE: One-pulse mode 0: The counter is not stopped at update event. 1: The counter stops counting at the next update event (clearing the CNTEN bit).
Bit 2	UEVRS: Update request source This bit is set and cleared by software to select the UEV event sources. 0: If interrupt/DMA is enabled, any of the following events generates an update interrupt or DMA request: - Counter overflow - Setting the UEVG bit - Updates generated from the slave mode controller 1: If interrupt/DMA is enabled, only counter overflow/underflow generates an update interrupt or DMA request.
Bit 1	UEVDIS: Update disable This bit is set and cleared by software to enable or disable UEV event generation. 0: UEV is enabled. An update event (UEV) is generated by one of the following events: - Counter overflow - Setting the UEVG bit - Updates generated from the slave mode controller After update events are generated, buffered registers are loaded with their preload values. 1: UEV is disabled. An update event is not generated, and the shadow registers keep their values (AR, DIV). However, the counter and the prescaler are reinitialized if the UEVG bit is set, or a hardware reset is received from the slave mode controller.
Bit 0	CNTEN: Counter enable 0: Counter is disabled. 1: Counter is enabled. <i>Note: Gated mode can work only if the CNTEN bit is set by software. In trigger mode, the CNTEN bit is automatically set by hardware. In one-pulse mode, the CNTEN bit is cleared automatically when an update event occurs.</i>

10.1.4.2 TMR6 and TMR7 Control Register 2 (TMRx_CTRL2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						MMSEL			Reserved						
res						rw	rw	rw							res
Bit 15:7															
Bit 6:4															
Bit 3:0															
Reserved. Always read as 0.															
MMSEL: Master mode selection These bits are used to select the information in master mode to be sent to slave timers for synchronization (TRGO). The combinations are as follows: 000: Reset - The UEVG bit in the TMRx_EVEG register is used as a trigger output (TRGO). If reset is generated by the trigger input (slave mode controller is configured as reset mode), then the signal on TRGO is delayed compared to the actual reset. 001: Enable - The counter enable signal, CNT_EN, is used as a trigger output (TRGO). It can start several timers at the same time or control the time to enable a slave timer. The counter enable signal is generated by a logic OR between the CNTEN control bit and the trigger input when configured as gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, unless the master/slave mode is selected (see the MSMODE bit in the TMRx_SMC register). 010: Update - The update event is used as a trigger output (TRGO). For instance, a master timer can be used as a prescaler for a slave timer.															

10.1.4.3 TMR6 and TMR7 DMA/Interrupt Enable Register (TMRx_DIE)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								UEVD E	Reserved							
res								rw	res							
Bit 15:9															UEV IE	
Bit 8															rw	
Bit 7:1																
Bit 0																
Bit 15:9																
Bit 8																
Bit 7:1																
Bit 0																

10.1.4.4 TMR6 and TMR7 Status Register (TMRx_STS)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								res	rw							
Bit 15:1															UEV IF	
Bit 0															rw	
Bit 15:1																
Bit 0																

10.1.4.5 TMR6 and TMR7 Event Generation Register (TMRx_EVEG)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								res	rw							
Bit 15:1															UEV G	
Bit 0															rw	
Bit 15:1																
Bit 0																

10.1.4.6 TMR6 and TMR7 Counter (TMRx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

10.1.4.7 TMR6 and TMR7 Prescaler (TMRx_DIV)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0	DIV[15:0]: Prescaler value The counter clock frequency CK_CNT equals fCK_DIV/(DIV[15:0]+1). At each update event, DIV value is sent to the actual prescaler register.														

10.1.4.8 TMR6 and TMR7 Auto-reload Register (TMRx_AR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0	AR[15:0]: Auto-reload value AR value will be sent to the actual auto-reload register. Please refer to Section 10.1.3.1 for more details about AR update and behavior. The counter stops when the auto-reload value is 0.														

10.2 General-purpose Timer (TMR2 to TMR5)

10.2.1 TMRx Introduction

The general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

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

The timers are completely independent, and they do not share any resources and can be synchronized. Please refer to [Section 10.2.3.15](#).

10.2.2 TMRx Main Function

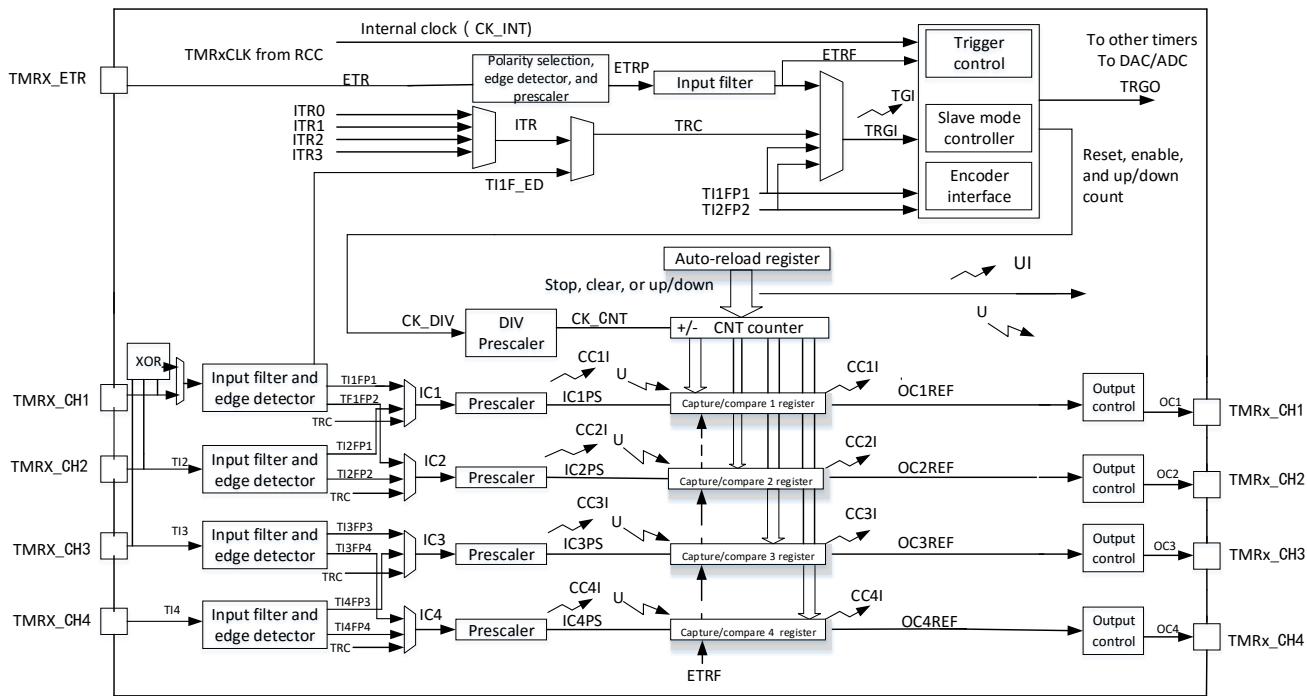
The main functions of general-purpose TMRx (TMR2, TMR3, TMR4, and TMR5) include:

- 16-bit up, down, and up/down auto-reload counter

Note: 32-bit counter can be selected in TMR2 and TMR5. Please refer to [Section 10.2.4.1](#) TMRx_CTRL1 register for more details.

- 16-bit programmable prescaler (can be modified on the fly) used to divide the counter clock frequency by any factor between 1 ~ 65536
- 4 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (Edge-aligned or Center-aligned modes)
 - One-pulse mode output
- Timers and the interconnected synchronization circuit are controlled by external signals.
- Interrupt/DMA is generated at the following events:
 - Update: Counter overflow/underflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter starts, stops, initialized, or counts by internal/external trigger)
 - Input capture
 - Output compare
- Support incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 10-11 General-purpose Timer Block Diagram



Note: transfers preload registers to active registers on U event according to the control bits.

Event

Interrupt and DMA output

10.2.3 TMRx Function Overview

10.2.3.1 Time-base Unit

This programmable timer mainly consists of a 16-bit counter and relevant auto-reload registers. The counter can count up, down, or both up and down. The counter clock is obtained through a prescaler.

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

The time-base unit includes:

- Counter register (TMRx_CNT)
- Prescaler register (TMRx_DIV)
- Auto-reload register (TMRx_AR)

The auto-reload register is preloaded. Each read or write to the auto-reload register will access the preload register. The contents written to the preload register are transferred into its shadow register immediately or at each update event (UEV) according to the auto-reload preload enable bit (ARPEN) in the TMRx_CTRL1 register. When the UEVDIS bit in the TMRx_CTRL1 register is '0', update events will be generated once the counter reaches the overflow value (or underflow when downcounting). Update events can also be generated by software. Please refer to the following sections for the detailed introduction on generation of the update events.

The counter is driven by the prescaler clock output CK_CNT, and it is enabled when the counter enable bit (CNTEN) in the TMRx_CTRL1 register is set. (Please refer to the controller slave mode descriptions for more details on counter enable.)

Note: The actual counter enable signal, CNT_EN, is set 1 clock cycle after CNTEN is set.

Prescaler

The prescaler can divide the counter clock frequency by any factor between 1 ~ 65536, and this is achieved with a 16-bit counter controlled by a 16-bit register (in the TMRx_DIV register). Its value can be changed on the fly since the control register is buffered. The new prescaler ratio takes effect at the next update event.

[Figure 10-12](#) and [Figure 10-13](#) are examples of on-the-fly prescaler ratio change.

Figure 10-12 Counter Timing Diagram with Prescaler Division Changing from 1 to 2

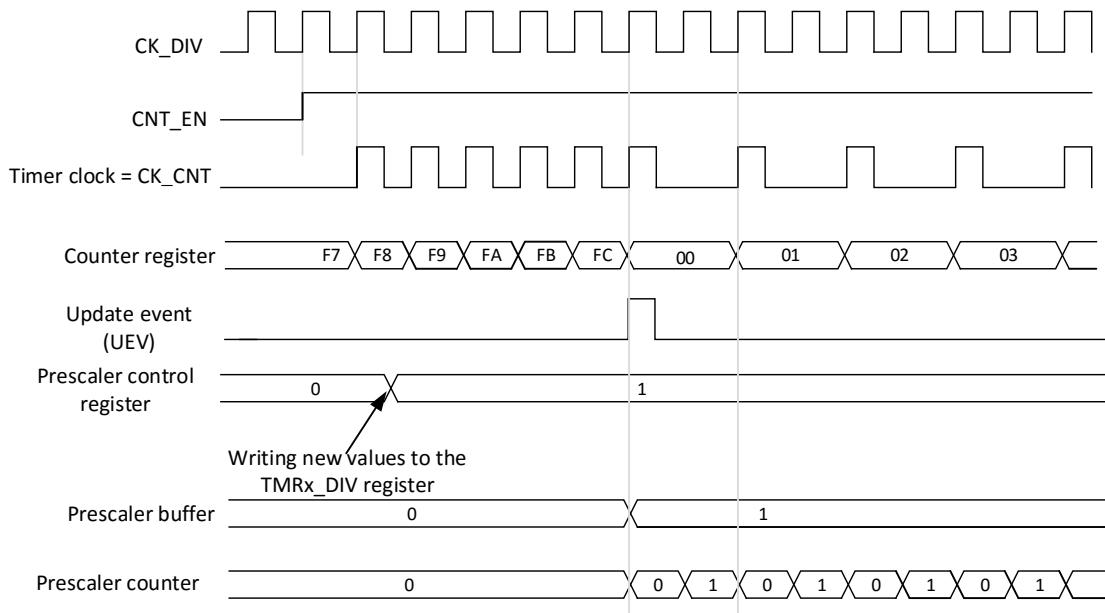
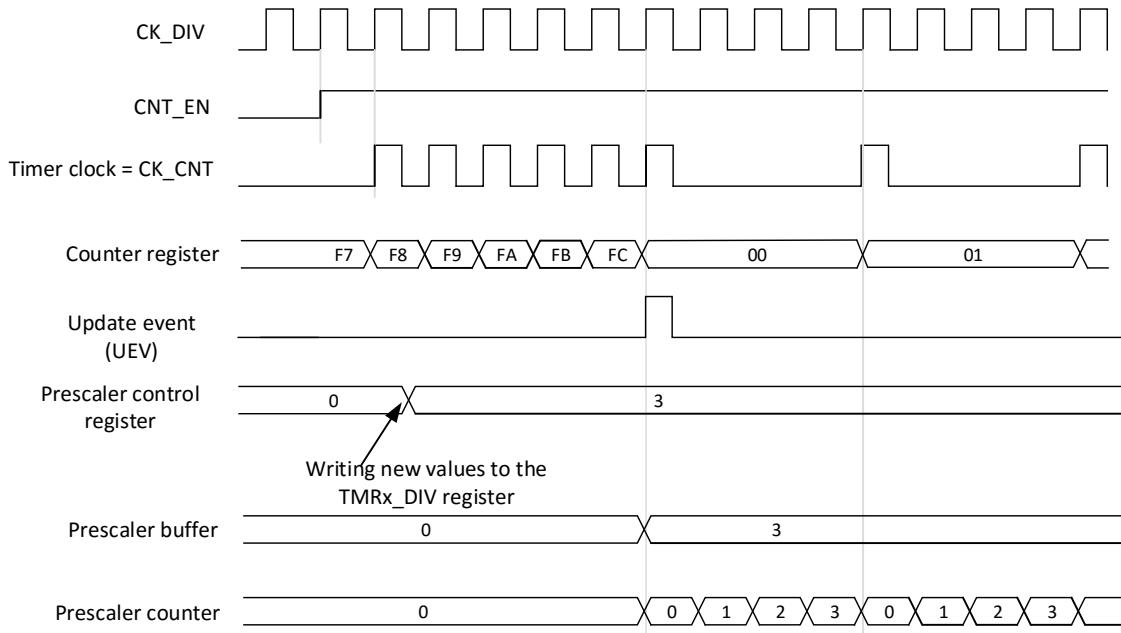


Figure 10-13 Counter Timing Diagram with Prescaler Division Changing from 1 to 4



10.2.3.2 Counter Mode

Upcounting mode

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

An update event can be generated at each counter overflow; setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller) also generates the update events.

Update events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing new values to the preload registers. No update event occurs

until the UEVDIS bit is cleared. However, when the update events are generated, the counter and the prescaler are still cleared (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set by hardware (that is, no interrupt or DMA request is sent). This can avoid update and capture interrupt generation when the counter is cleared in capture mode.

When an update event occurs, all the registers are updated, and the update flag (the UEVIF bit in the TMRx_STS register) is set by hardware (according to the UEVRS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).
- The auto-reload shadow register is updated with the preload value (TMRx_AR).

Figure 10-14 ~ Figure 10-19 show the examples of the counter behavior for different clock frequencies when TMRx_AR = 0x36:

Figure 10-14 Counter Timing Diagram with Internal Clock Divided by 1

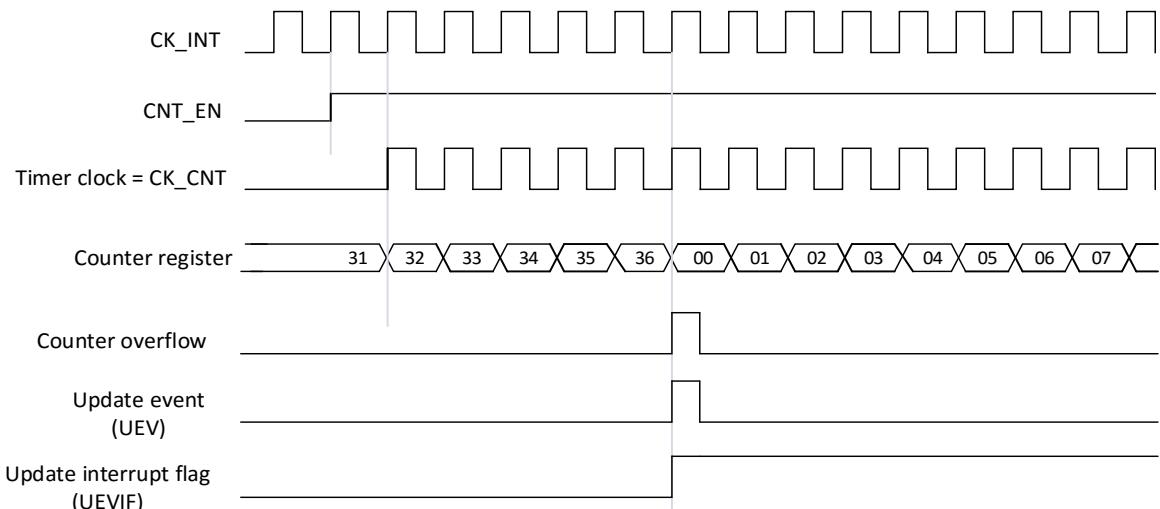


Figure 10-15 Counter Timing Diagram with Internal Clock Divided by 2

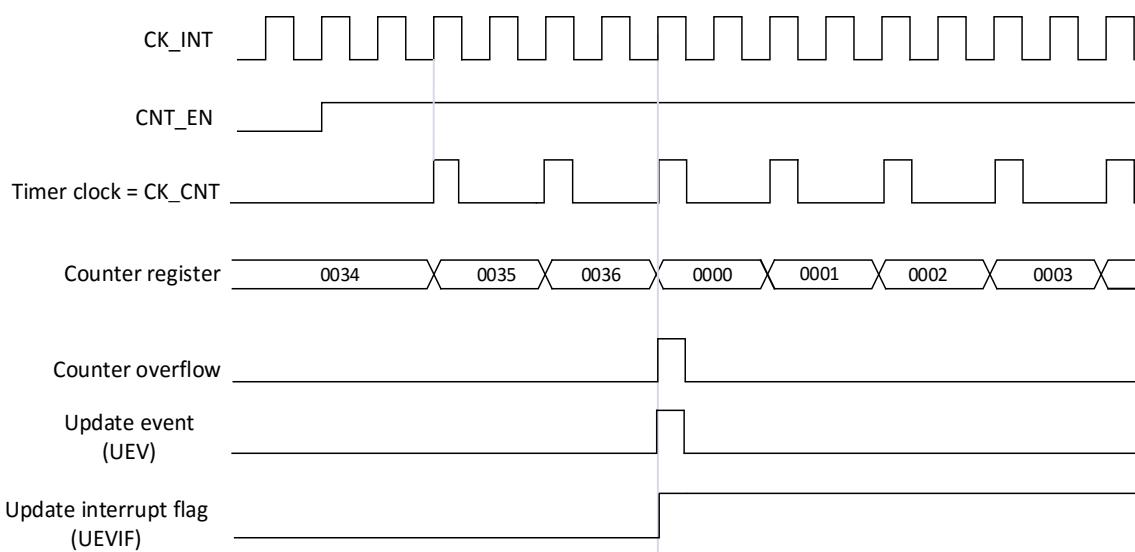


Figure 10-16 Counter Timing Diagram with Internal Clock Divided by 4

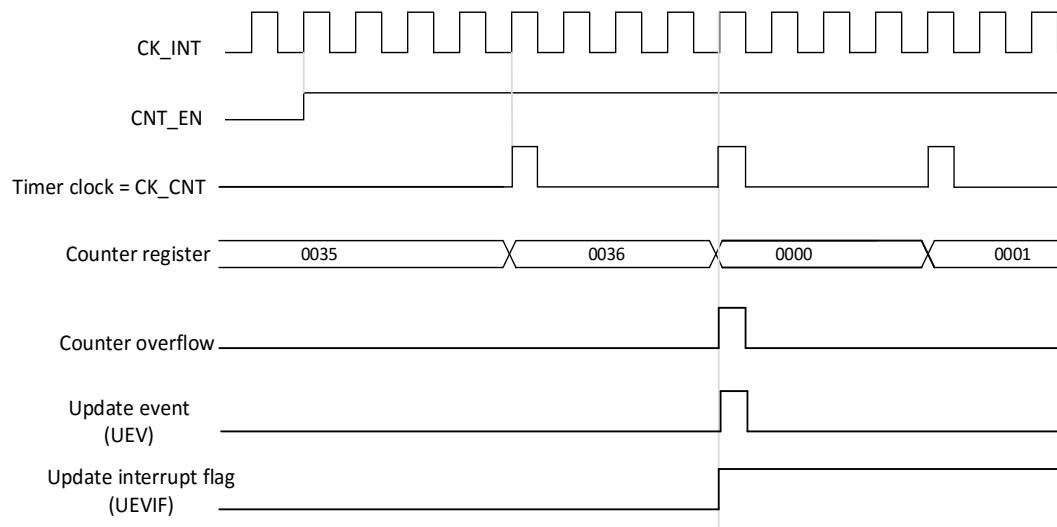


Figure 10-17 Counter Timing Diagram with Internal Clock Divided by N

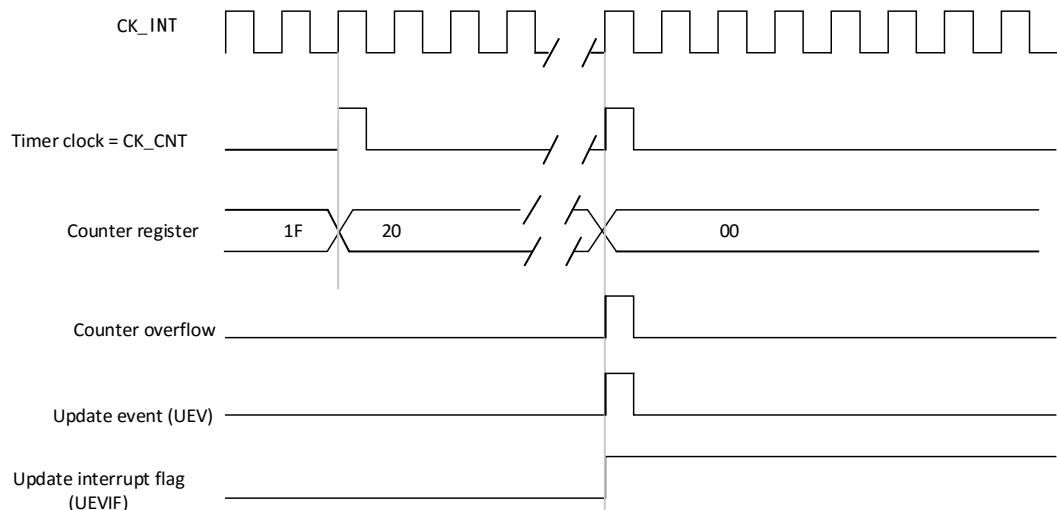


Figure 10-18 Counter Timing Diagram with Update Event When ARPEN = 0 (TMRx_AR Not Preloaded)

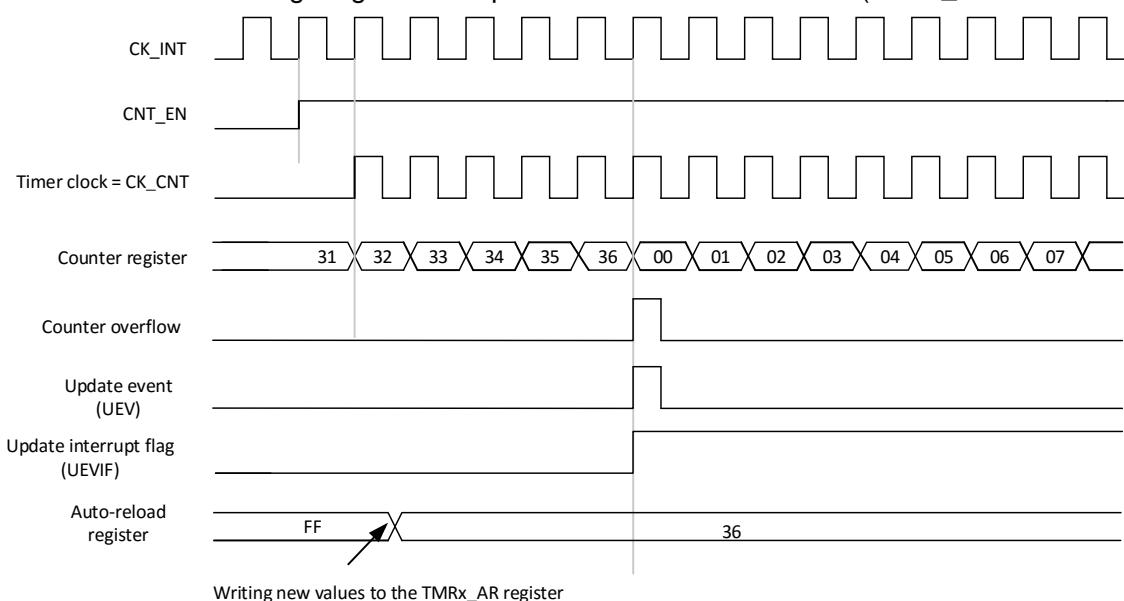
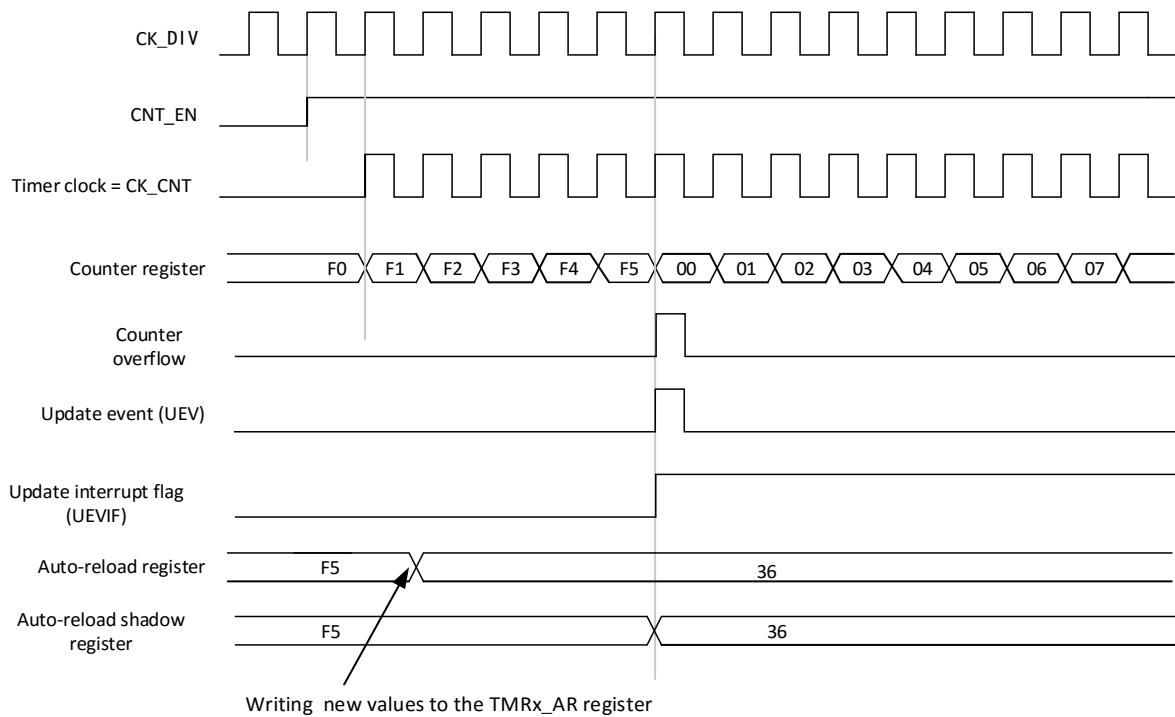


Figure 10-19 Counter Timing Diagram with Update Event When ARPEN = 1 (TMRx_AR is Preloaded)



Downcounting mode

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

An update event can be generated at each counter underflow; setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller) also generates an update event.

Update events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing new values to the preload registers. Thus, no update event occurs until the UEVDIS bit is cleared. However, the counter still counts from the current auto-reload value, and the prescaler counter restarts from 0 (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set (so no interrupt or DMA request is generated). This can avoid update and capture interrupts generation when a capture event occurs and the counter is cleared.

When an update event occurs, all the registers are updated and the update flag bit (the UEVIF bit in the TMRx_STS register) is set (according to the UEVRS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).
- The current auto-reload register is updated with the preload value (content of the TMRx_AR register).

Note: Auto-reload is updated before the counter is reloaded, so the next cycle will be the expected value.

Figure 10-20 ~ Figure 10-24 show the examples of the counter behavior for different clock frequencies when TMRx_AR = 0x36:

Figure 10-20 Counter Timing Diagram with Internal Clock Divided by 1

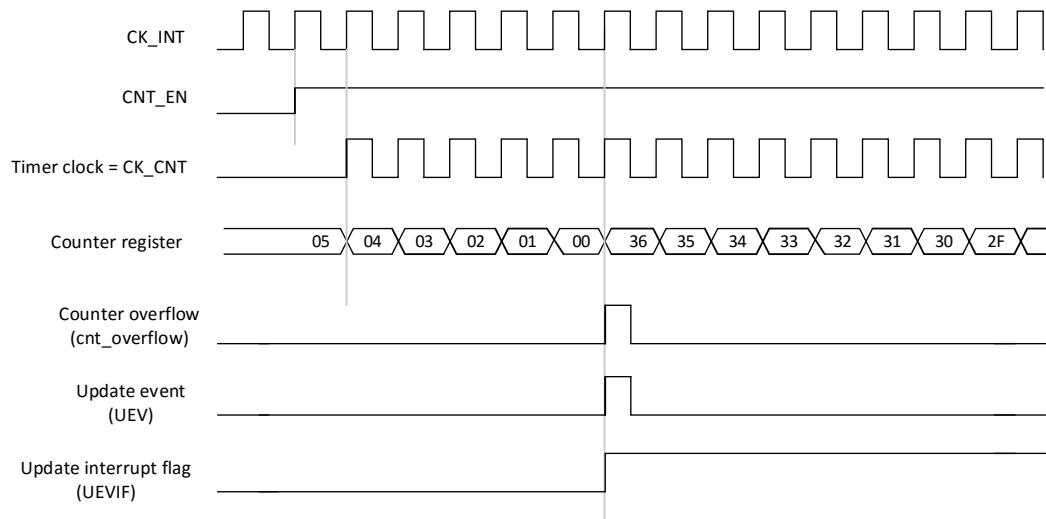


Figure 10-21 Counter Timing Diagram with Internal Clock Divided by 2

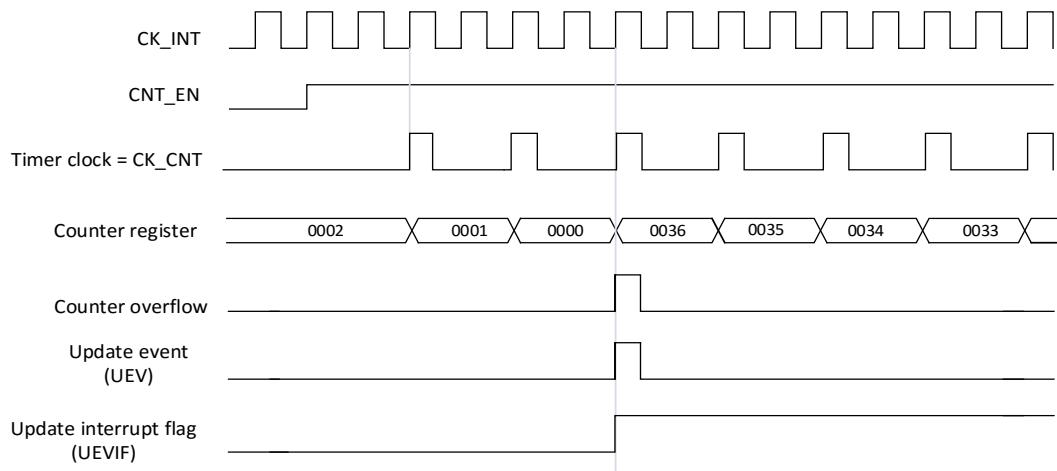


Figure 10-22 Counter Timing Diagram with Internal Clock Divided by 4

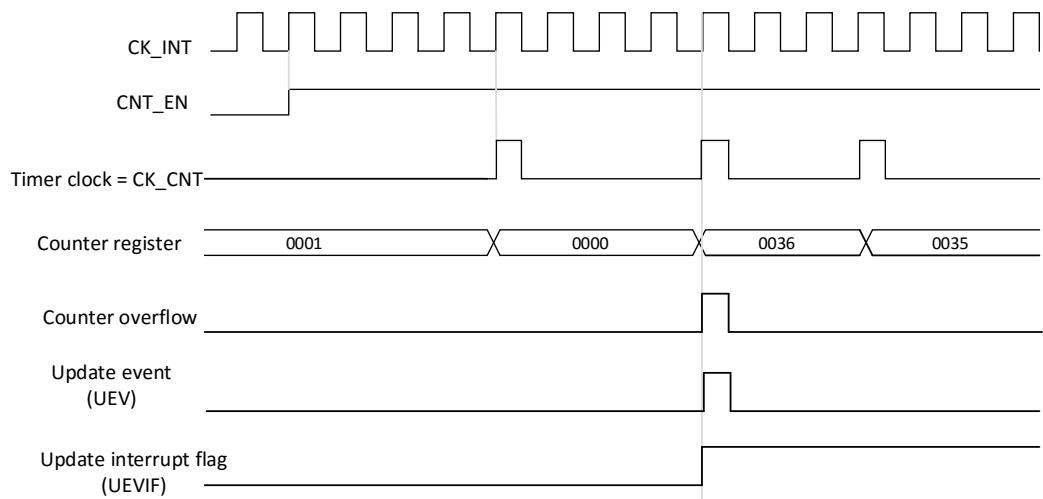


Figure 10-23 Counter Timing Diagram with Internal Clock Divided by N

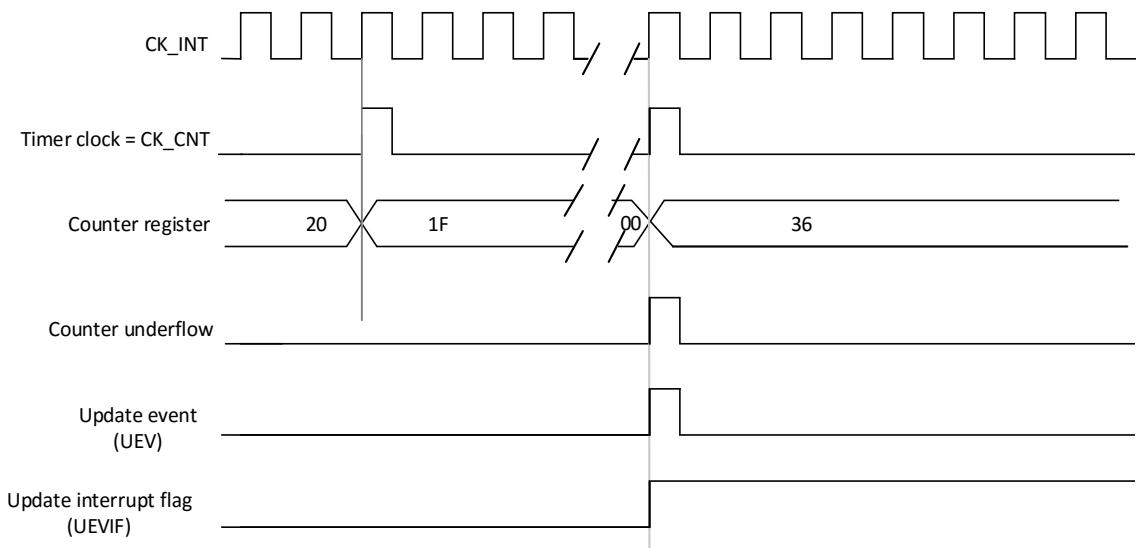
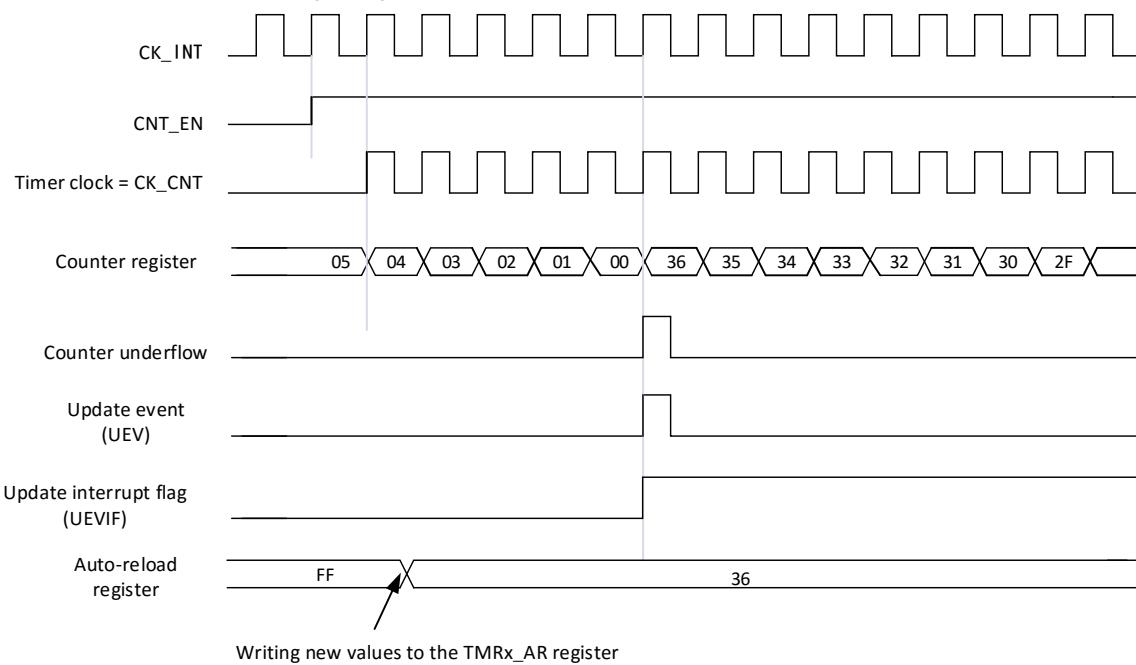


Figure 10-24 Counter Timing Diagram with Update Event When ARPEN = 0



Center-aligned mode (Up/Down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TMRx_AR register) – 1, generates a counter overflow event, and counts from the auto-reload value down to 1 and generates a counter underflow event. Then, it restarts counting from 0.

In this mode, the DIR direction bit in the TMRx_CTRL1 register cannot be written. It is updated by hardware and indicates the current counting direction.

The update event can be generated at each counter overflow and at each counter underflow, or by setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller). Afterwards, the counter as well as the prescaler restarts counting from 0.

The UEV update event can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This can avoid updating the shadow registers while writing new values to the preload registers. Therefore, no update event occurs until the UEVDIS bit is cleared. However, the counter continues counting up or down based on the current auto-reload value. In addition, if the UEVRS bit (update request selection) in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV without setting the UEVIF flag (so no interrupt or DMA request is generated). This can avoid generating both update and capture interrupts when the capture event occurs and the counter is cleared.

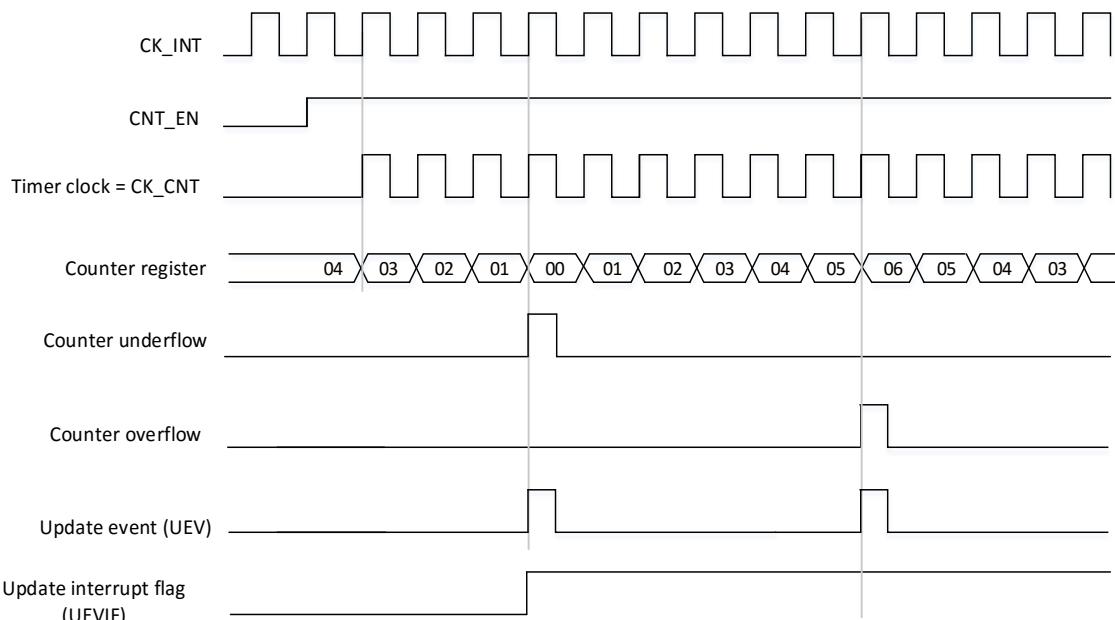
When an update event occurs, all the registers are updated, and the update flag bit (the UEVIF bit in the TMRx_STS register) is set (according to the UEVRS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).
- The current auto-reload register is updated with the preload value (content of the TMRx_AR register).

Note: *If an update is generated because of counter overflow, auto-reload is updated before the counter is reloaded, so the next cycle will be the expected value (the counter is reloaded with the new value).*

Figure 10-25 ~ Figure 10-30 show the examples of the counter behavior for different clock frequencies:

Figure 10-25 Counter Timing Diagram with Internal Clock Divided by 1, TMRx_AR = 0x6



Note: Center-aligned mode 1 is used here (Please refer to [Section 10.2.4.1](#) for the register configuration).

Figure 10-26 Counter Timing Diagram with Internal Clock Divided by 2

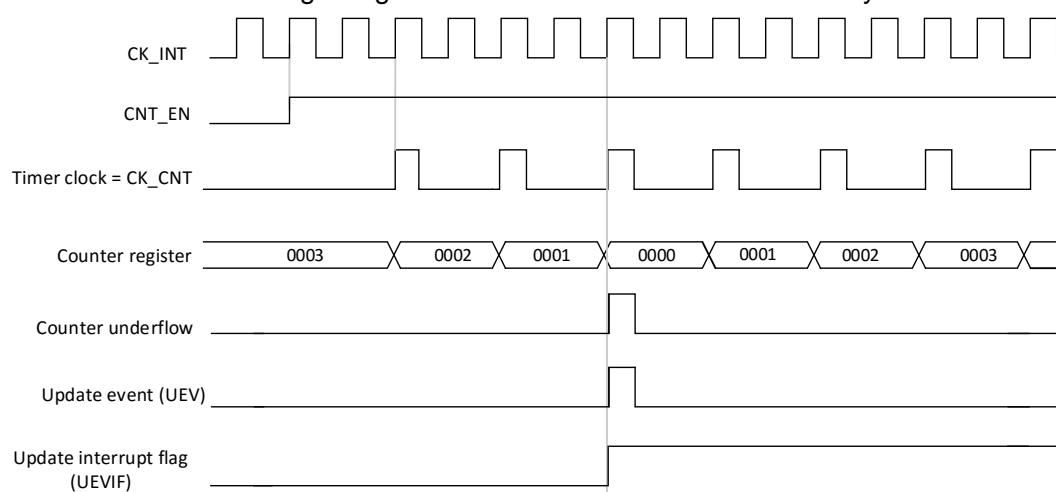


Figure 10-27 Counter Timing Diagram with Internal Clock Divided by 4, TMRx_AR = 0x36

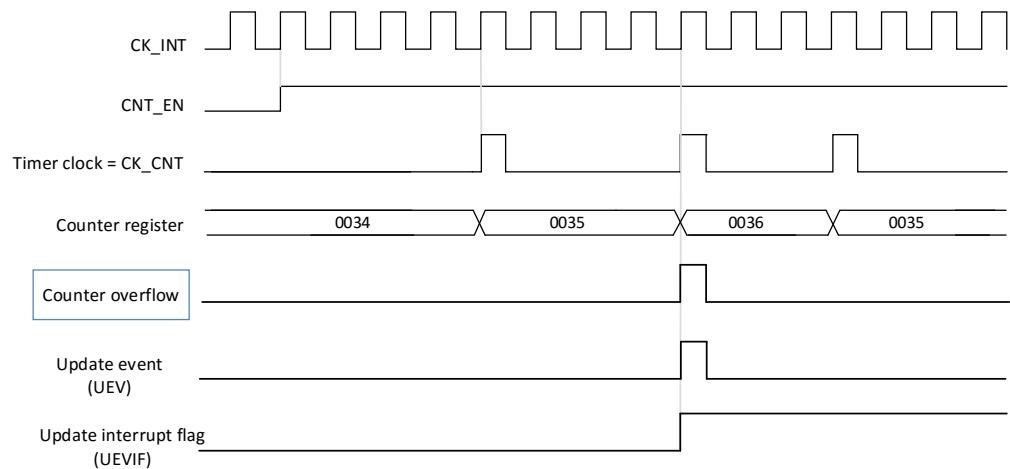


Figure 10-28 Counter Timing Diagram with Internal Clock Divided by N

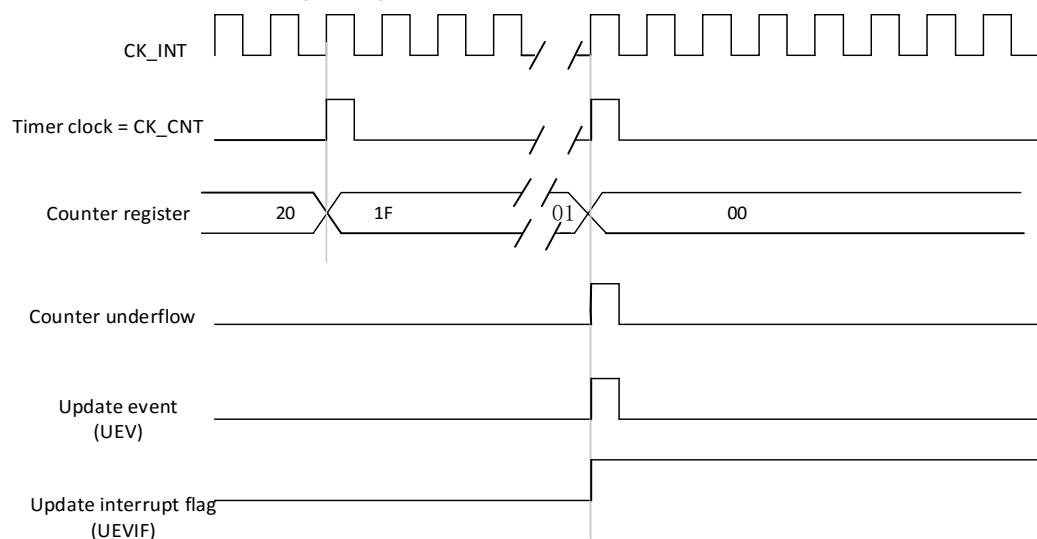


Figure 10-29 Counter Timing Diagram with Update Event When ARPEN = 1 (Counter underflow)

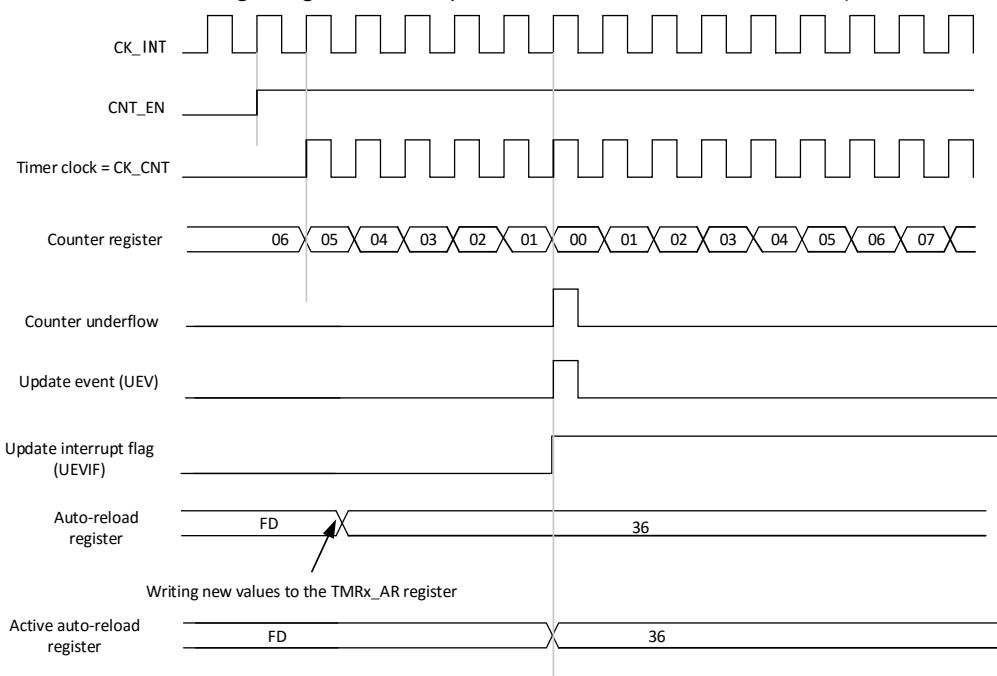
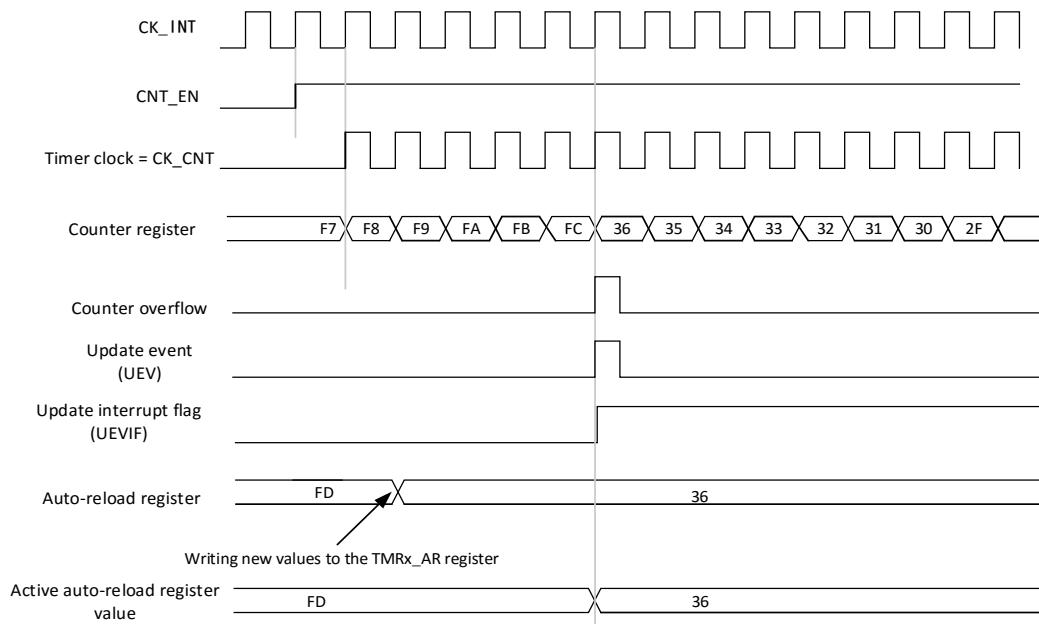


Figure 10-30 Counter Timing Diagram with Update Event When ARPEN = 1 (Counter overflow)



10.2.3.3 Clock Selection

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

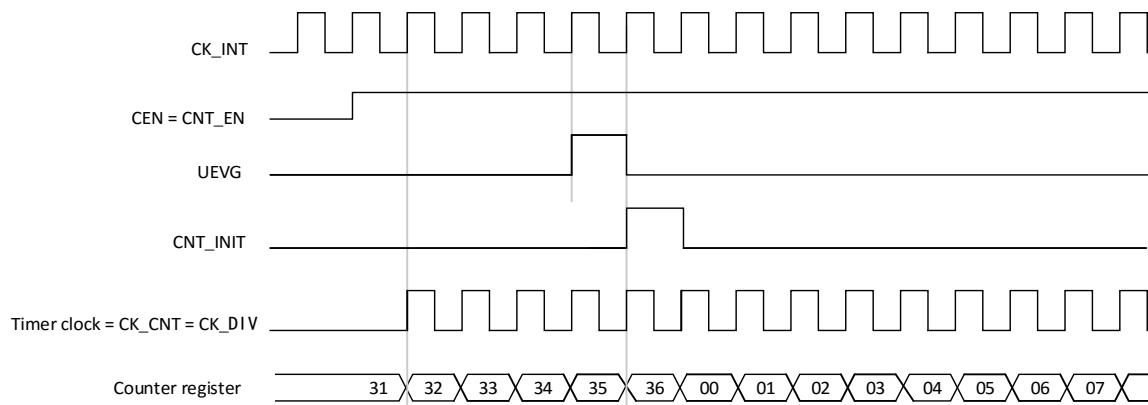
- Internal clock (CK_INT)
- External clock mode 1: External input pin (TIx)
- External clock mode 2: External trigger input (ETR)
- Internal trigger input (ITRx): Use one timer as the prescaler for another timer. For example, the user can configure Timer 1 to act as the prescaler for Timer 2. Please refer to [Section 10.2.3.15](#).

Internal clock source (CK_INT)

If the slave mode controller is disabled (SMSEL = 000 in the TMRx_SMC register), the CNTEN, DIR (in the TMRx_CTRL1 register) and UEVG bits (in the TMRx_EVEG register) are the actual control bits, and they can be changed only by software (except for the UEVG bit, which will be cleared automatically). As soon as the CNTEN bit is written '1', the prescaler is clocked by the internal clock CK_INT.

Figure 10-31 shows the behavior of the control circuit and the up counter in normal mode, without prescaler.

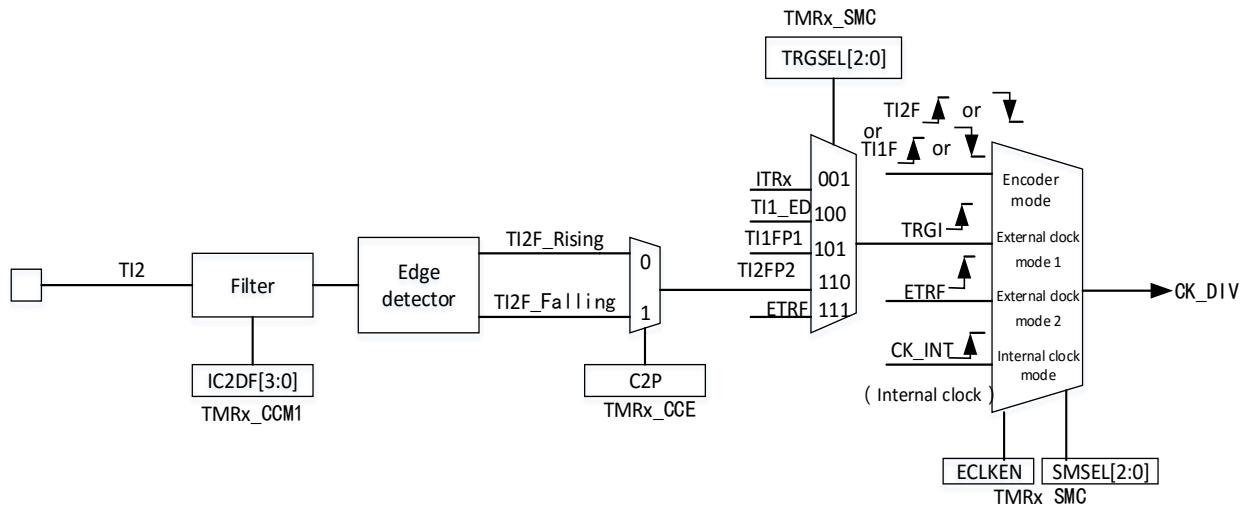
Figure 10-31 Control Circuit in Normal Mode with Internal Clock Divided by 1



External clock mode 1

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

Figure 10-32 TI2 External Clock Connection Example



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

1. Configure channel 2 to detect the rising edges on the TI2 input by writing C2SEL = '01' in the TMRx_CCM1 register.
2. Configure the input filter duration by writing the IC2DF[3:0] bits in the TMRx_CCM1 register (if no filter is needed, keep IC2DF = 0000).

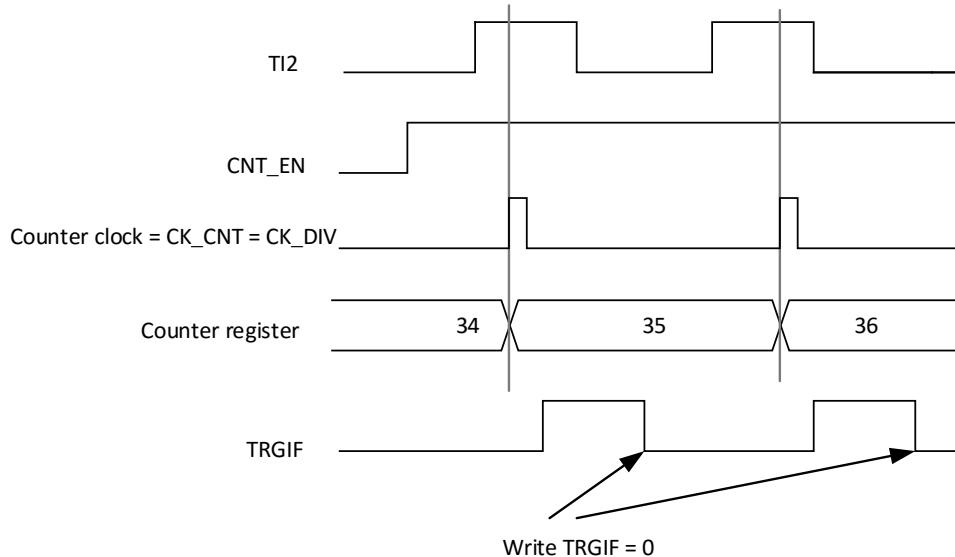
Note: The capture prescaler is not used as trigger, so it does not need to be configured.

3. Select the rising edge polarity by writing C2P = '0' in the TMRx_CCE register.
4. Select the timer external clock mode 1 by writing SMSEL = '111' in the TMRx_SMC register.
5. Select TI2 as the trigger input source by writing TRGSEL = '110' in the TMRx_SMC register.
6. Enable the counter by writing CNTEN = '1' in the TMRx_CTRL1 register.

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

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

Figure 10-33 Control Circuit in External Clock Mode 1



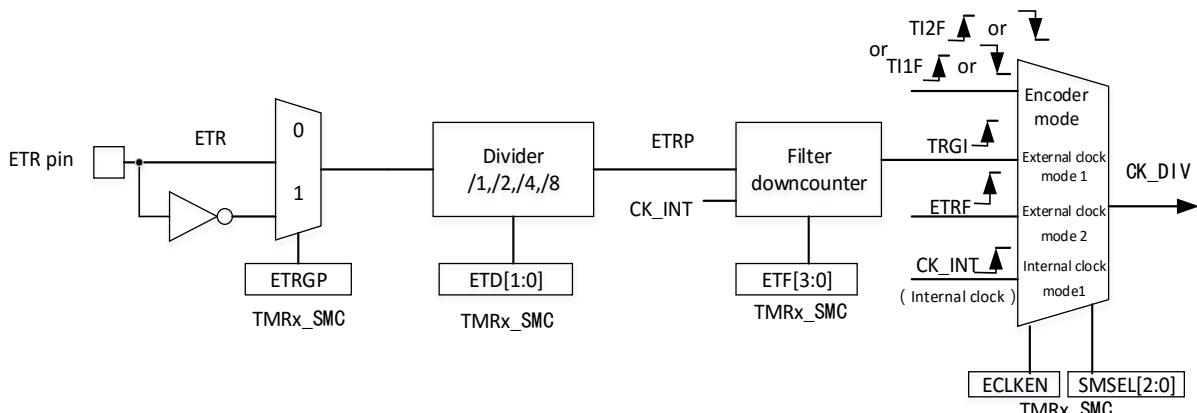
External clock mode 2

This mode is selected by writing ECLKEN = 1 in the TMRx_SMC register.

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

Figure 10-34 shows the block diagram of the external trigger input.

Figure 10-34 Block Diagram of External Trigger Input



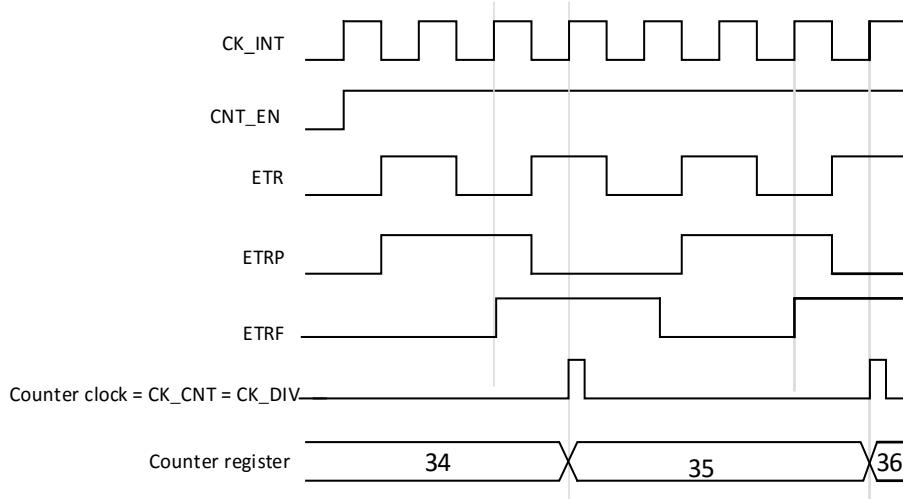
For example, to configure the up counter to count each 2 rising edges on ETR, use the following procedure:

1. Since no filter is needed in this example, write ETDF[3:0] = 0000 in the TMRx_SMC register.
2. Set the prescaler by writing ETD[1:0] = 01 in the TMRx_SMC register.
3. Select the rising edge detection on ETR by writing ETRGP = 0 in the TMRx_SMC register.
4. Enable the external clock mode 2 by writing ECLKEN = 1 in the TMRx_SMC register.
5. Enable the counter by writing CNTEN = 1 in the TMRx_CTRL1 register.

The counter counts once each 2 ETR rising edges.

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

Figure 10-35 Control Circuit in External Clock Mode 2

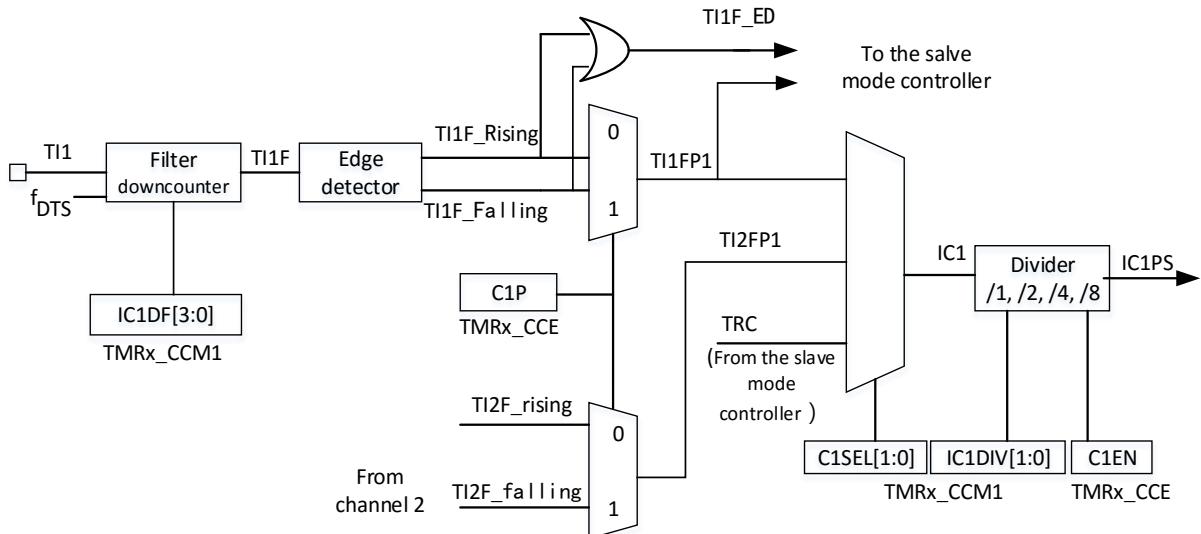


10.2.3.4 Capture/Compare Channel

Each capture/compare channel is built around a capture/compare register (including a shadow register), also an input stage for capture (digital filter, multiplexing, and prescaler) and an output stage (comparator and output control). Figure 10-36 ~ Figure 10-38 provide an overview of one capture/compare channel.

The input stage samples the corresponding TIx input signal to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be the trigger input for the slave mode controller, or be the capture command. This signal enters the capture register through the prescaler.

Figure 10-36 Capture/Compare Channel (e.g. Channel 1 Input Stage)



The output stage generates an intermediate waveform OCxRef (active high) which serves as reference. The polarity of final output signal is determined by the end of the chain.

Figure 10-37 Capture/Compare Channel 1 Main Circuit

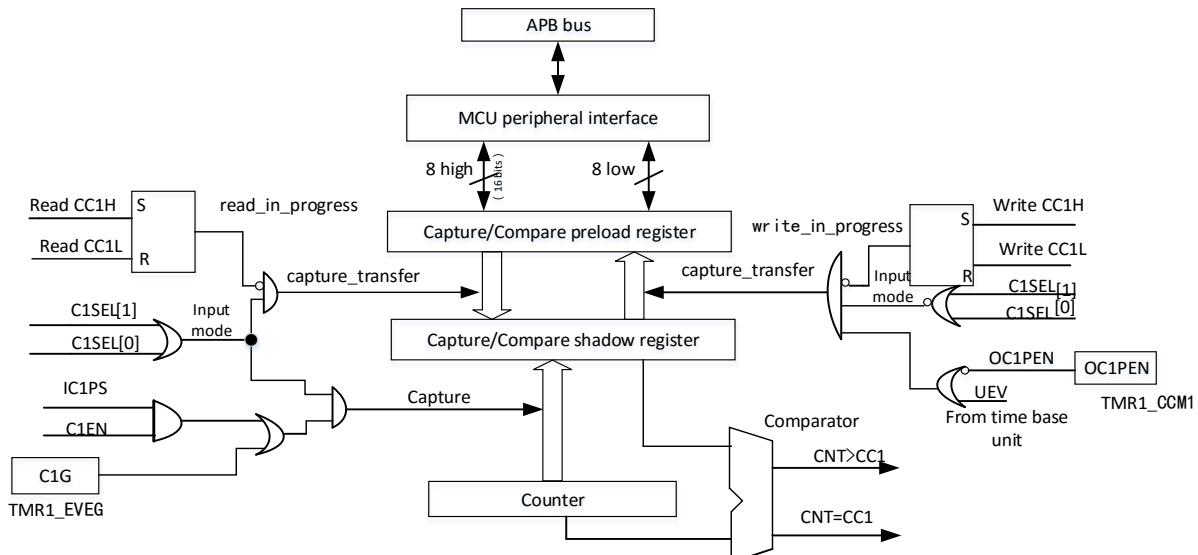
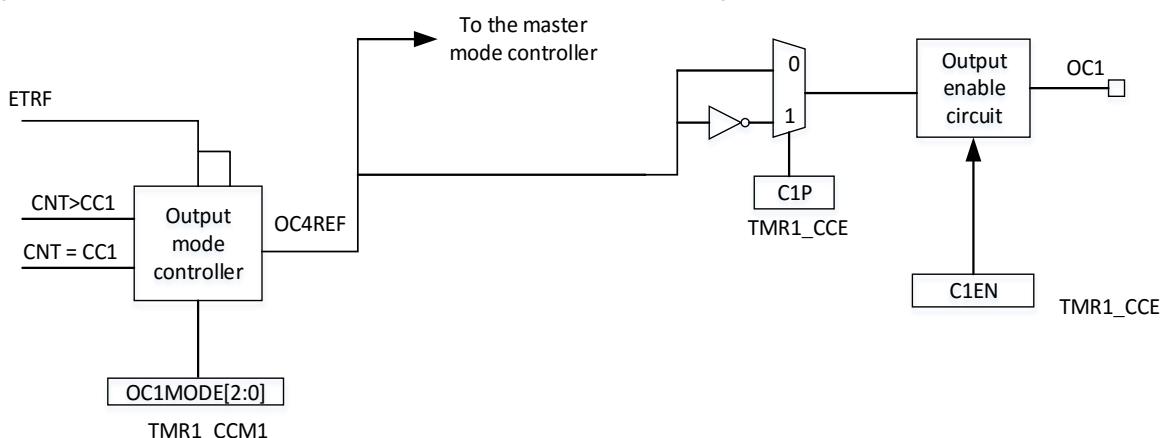


Figure 10-38 Capture/Compare Channel Output Stage (Channel 1)



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

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

In compare mode, the content of the preload register is copied into the shadow register, and then the content of the shadow register is compared with the counter.

10.2.3.5 Input Capture Mode

In input capture mode, the capture/compare registers (TMRx_CC x) latch the current counter value after a corresponding edge on the IC x signal is detected. When a capture occurs, the corresponding CxIF flag (the TMRx_STS register) is set. An interrupt or a DMA request can be sent if enabled. If a capture occurs when the CxIF flag is already high, the over-capture flag CxOF (the TMRx_STS register) is set. CxIF can be cleared by writing CxIF = 0 or by reading the captured data stored in the TMRx_CC x register. CxOF = 0 is cleared when writing CxOF = 0.

The following example shows how to capture the counter value to the TMRx_CC1 register during TI1 input rising edge:

- Select the active input: Write C1SEL = 01 in the TMRx_CCM1 register since the TMRx_CC1 register must be linked to the TI1 input. The channel is configured as input as long as the C1SEL is not '00'. The TM1_CC1 register becomes read-only.
- Program the desired input filter duration according to the input signal (if the input is TI x , the control bit of the input filter is the ICxDF bit in the TMRx_CCM x register). If the input signal is not stable during five internal clock cycles at most, the filter duration should be configured longer than five clock cycles. Therefore, an actual edged transition on TI1 can be validated by 8 consecutive samples (at f_{DTs} frequency). That is, write IC1DF = 0011 in the TMRx_CCM1 register.
- Select the active transition edge on the TI1 channel by writing C1P = 0 (rising edge) in the TMRx_CCE register.
- Program the input prescaler. In this example, each valid transition is expected to be captured, so the prescaler is disabled (write IC1DIV = 00 in the TMRx_CCM1 register).
- Set C1EN = 1 in the TMRx_CCE register to allow the counter value to be captured into the capture register
- If needed, enable the relevant interrupt request by setting the C1IE bit in the TMRx_DIE register, and the DMA request by setting the C1DE bit in the TMRx_DIE register.

When an input capture occurs:

- The counter value is sent to the TMRx_CC1 register on active transition.
- C1IF flag is set (interrupt flag). When at least two consecutive captures occur and the C1IF flag is not cleared, C1OF is also set.
- An interrupt is generated if the C1IE bit is set.
- A DMA request is generated if the C1DE bit is set.

In case of overcapture, it is recommended that the data be read before the overcapture flag. This is to avoid missing an overcapture information that is generated after reading the overcapture flag and before reading the data.

Note: Input capture interrupt and/or DMA requests can be generated by setting the corresponding CxG bit in the TMRx_EVEG register with software.

10.2.3.6 PWM Input Mode

This mode is a special case of input capture mode. The procedure is the same as input capture mode except for the following conditions:

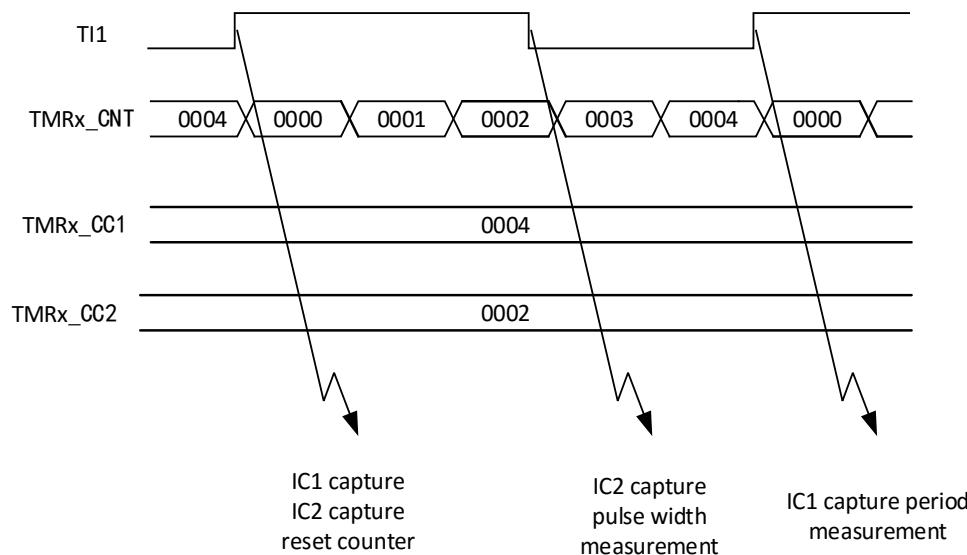
- Two IC x signals are mapped on the same TI x input.
- The two IC x signals are active on edges but with opposite polarities.
- One of the two TI x FP signals is used as trigger input signal and the slave mode

controller is configured as reset mode.

For example, if users need to measure the period (the TMRx_CC1 register) and the duty cycle (the TMRx_CC2 register) of the PWM signal input on TI1, the procedure is as follows (depending on CK_INT frequency and prescaler value):

- Select the active input for TMRx_CC1: Write C1SEL = 01 in the TMRx_CCM1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used to capture data into TMRx_CC1 and clear the counter): Write C1P = 0 (active on rising edge).
- Select the active input for TMRx_CC2: Write C2SEL = 10 in the TMRx_CCM1 register (TI1 selected).
- Select the active polarity for TI1FP2 (capture data to TMRx_CC2): Write C2P = 1 (active on falling edge).
- Select the valid trigger input signal: Write TRGSEL = 101 in the TMRx_SMC register (TI1FP1 selected).
- Configure the slave mode controller as reset mode: Write SMSEL = 100 in the TMRx_SMC register.
- Enable the captures: Write C1EN = 1 and C2EN = 1 in the TMRx_CCE register.

Figure 10-39 PWM Input Mode Timing



Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can be used only with the TMRx_CH1/TMRx_CH2 signals.

10.2.3.7 Forced Output Mode

In output mode (CxSEL = 00 in the TMRx_CCMx register), output compare signal (OCxREF and the corresponding OCx) can be forced to be active or inactive level directly by software, independent of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, users can write the corresponding OCxMODE = 101 in the TMRx_CCMx register. In this way, OCxREF is forced high (OCxREF is always active high), and at the same time OCx gets the opposite value of the CxP polarity bit.

For example: CxP = 0 (OCx active high), then OCx is forced to be high level.

The OCxREF signal can be forced low by writing OCxMODE = 100 in the TMRx_CCx register.

In this mode, the comparison between the TMRx_CCx shadow register and the counter is still ongoing, and the corresponding flag is also modified. Accordingly, the corresponding interrupt and DMA requests are still generated. This will be described in the “Output Compare Mode” section.

10.2.3.8 Output Compare Mode

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

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

- Output the value defined by the output compare mode (the OCxMODE bit in the TMRx_CCMx register) and the output polarity (the CxP bit in the TMRx_CCE register) to the corresponding pin. When being compared and matched, the output pin can keep its level (OCxMODE = 000), be set active (OCxMODE = 001), be set inactive (OCxMODE = 010), or toggle (OCxMODE = 011).
- Set the flag bit in the interrupt status register (the CxIF bit in the TMRx_STS register).
- An interrupt will be generated if the corresponding interrupt mask is set (the CxIE bit in the TMRx_DIE register).
- A DMA request will be generated if the corresponding enable bit is set (the CxDE bit in the TMRx_DIE register, the CDSEL bit in the TMRx_CTRL2 register for the DMA request selection).

The OCxPEN bit in the TMRx_CCMx register can be configured to choose whether the TMRx_CCx register uses the preload register.

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

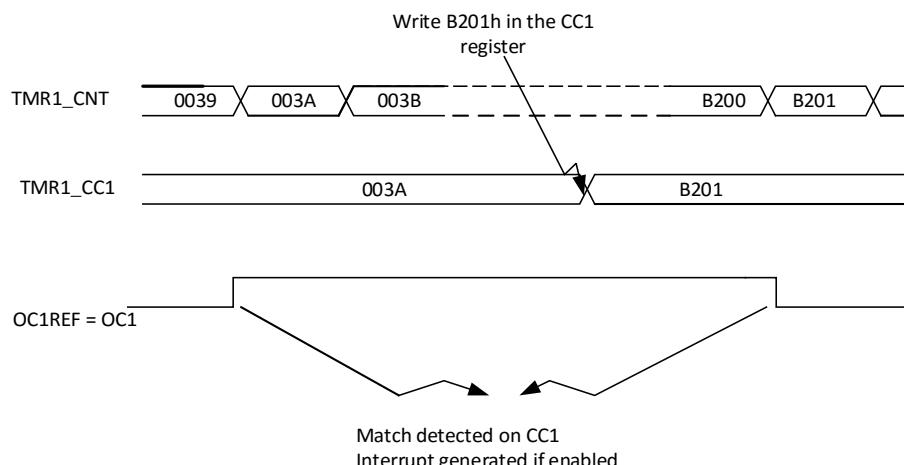
The timing resolution can reach one count of the counter. Output compare mode can also be used to output a single pulse (in one-pulse mode).

Output compare mode configuration procedure is as follows:

1. Select the counter clock (internal, external, and prescaler).
2. Write the corresponding data to the TMRx_AR and the TMRx_CCx registers.
3. Set the CxIE and/or CxDE bits if an interrupt and/or DMA request is to be generated.
4. Select the output mode. For example, when the counter CNT and the CCx bit are matched, toggle the output pin of OCx, CCx preload is not disabled, and the OCx output is enabled with active high, users must configure OCxMODE = '011', OCxPEN = '0', CxP = '0', and CxEN = '1'.
5. Enable the counter by setting the CNTEN bit in the TMRx_CTRL1 register.

The TMRx_CCx register can be updated at any time by software to control the output waveform, under the condition that the preload register is not enabled (OCxPEN = '0', else the TMRx_CCx shadow register can only be updated at the next update event). An example is shown in Figure 10-40.

Figure 10-40 Output Compare Mode, Toggle on OC1



10.2.3.9 PWM Mode

Pulse width modulation mode can generate a signal with its frequency determined by the TMRx_AR register and its duty cycle determined by TMRx_CCx register.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxMODE bit in the TMRx_CCMx register can independently configure each OCx output channel and generate single-channel PWM. The OCxPEN bit in the TMRx_CCMx register must be configured to enable the corresponding preload register. Finally, the ARPEN bit in the TMRx_CTRL1 register (in upcounting or center-aligned modes) must be set to enable the auto-reload preload register.

Since the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, users must initialize all the registers by setting the UEVG bit in the TMRx_EVEG register. OCx polarity is software programmable by setting the CxP bit in the TMRx_CCE register. It can be programmed as active high or active low. OCx output enable is controlled by the CxEN bit in the TMRx_CCE register. Please refer to the description of the [TMRx_CCE Register](#) for more details.

In PWM mode (1 or 2), TMRx_CNT and TMRx_CCx are always compared (according to the direction of the counter) to confirm that $TMRx_CCx \leq TMRx_CNT$ or $TMRx_CNT \leq TMRx_CCx$. However, to ensure the consistency with OCREF_CLR function (before the next PWM cycle, OCxREF can be cleared by an external event on the ETR signal), OCxREF is generated only when:

- The result of comparison changes.
- The output compare mode (the OCxMODE bit in the TMRx_CCMx register) is switched from "frozen" (No comparison, OCxMODE = '000') to a certain PWM mode (OCxMODE = '110' or '111').

In this way, software can force PWM output while running.

The timer can generate the edge-aligned or center-aligned PWM signal according to the status of CMSEL bit in the TMRx_CTRL1 register.

PWM edge-aligned mode

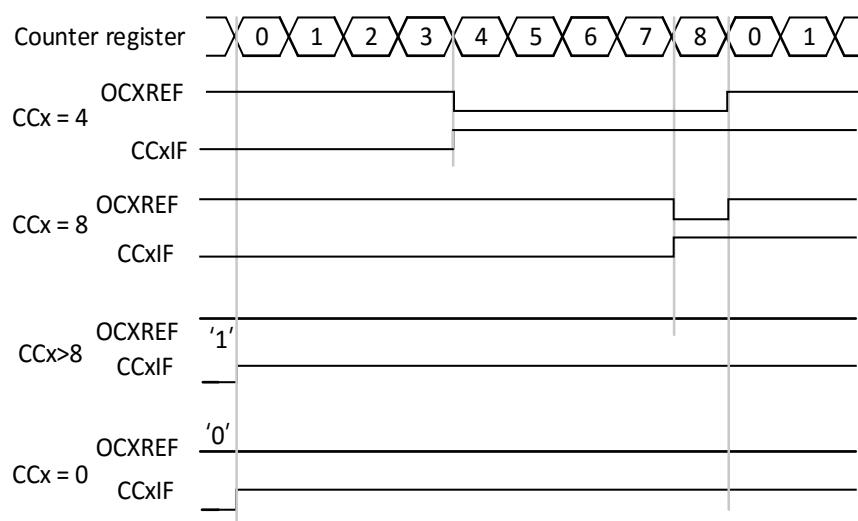
Upcounting configuration

Upcounting is active when the DIR bit in the TMRx_CTRL1 register is low. Please refer to [Section 10.2.3.2](#).

The following is an example of PWM mode 1. The reference PWM signal, OCxREF, is high once $TMRx_CNT < TMRx_CCx$; else, it becomes low. If the compare value in TMRx_CCx is greater than the auto-reload value (in TMRx_AR), OCxREF is held at '1'.

If the compare value is 0, then OCxREF is held at '0'. Figure 10-41 shows an example of edge-aligned PWM waveforms with TMRx_AR = 8.

Figure 10-41 Edge-aligned PWM Waveforms (AR = 8)



Downcounting configuration

Downcounting is active when the DIR bit in the TMRx_CTRL1 register is high. Please refer to [Section 10.2.3.2](#). In PWM mode 1, the reference signal, OCxREF, is low once $TMRx_CNT > TMRx_CCx$; else, it becomes high. If the compare value in TMRx_CCx is greater than the auto-reload value in TMRx_AR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

PWM center-aligned mode

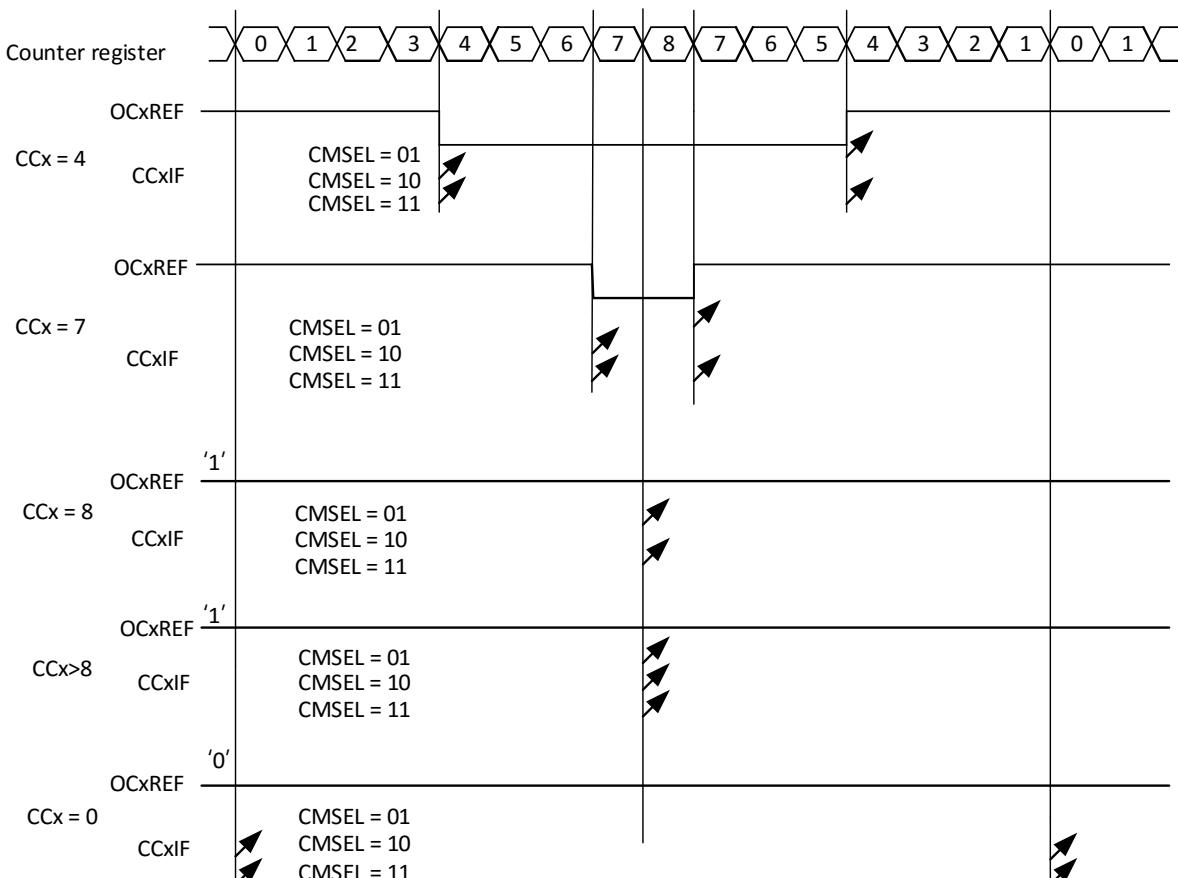
Center-aligned mode is active when the CMSEL bit in the TMRx_CTRL1 register is not '00' (all the other

configurations have the same effect on the OCxREF/OCx signals). According to different CMSEL bit configurations, the compare flag is set when the counter counts up, counts down, or both counts up and down. The direction bit (DIR) in the TMRx_CTRL1 register is updated by hardware and must not be changed by software. Please refer to the center-aligned mode in [Section 10.2.3.2](#).

Figure 10-42 shows several center-aligned PWM waveforms in an example where:

- TMRx_AR = 8
- PWM mode 1
- The compare flag is set when the counter counts down, center-aligned mode 1 is selected, and CMSEL = 01 in the TMRx_CTRL1 register.

Figure 10-42 Center-aligned PWM Waveforms (AP = 8)



Hints on using center-aligned mode:

- When entering the center-aligned mode, the current up/down configuration is used. This means whether the counter counts up or down depends on the current value of the DIR bit in the TMRx_CTRL1 register. In addition, the DIR and CMSEL bits cannot be changed by software at the same time.
- In center-aligned mode, writing to the counter while running is not recommended as it may lead to unexpected results. In particular:
 - If the value written to the counter is greater than the auto-reload value (TMRx_CNT > TMRx_AR), the direction will not be updated. For example, if the counter is counting up, it will keep counting up.
 - The direction is updated if 0 or the TMRx_AR value is written to the counter, but no update event UEV will be generated.
- The safest way to use the center-aligned mode is to generate an update by software (setting the UEVG bit in the TMRx_EVEG register) before enabling the counter, and do not write the counter while it is running.

10.2.3.10 One-pulse Mode

One-pulse mode (OPM) is quite different from the above-mentioned modes. It allows the counter to respond to a stimulus and to generate a pulse with a programmable length after a programmable delay.

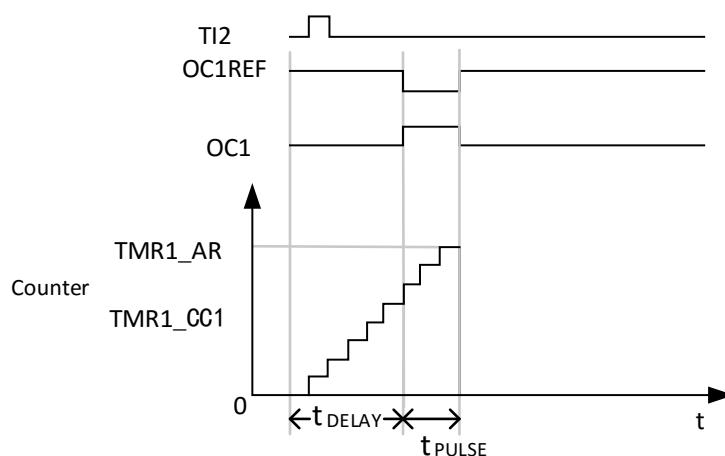
The counter can be enabled through the slave mode controller, generating waveforms in output compare mode or PWM mode. One-pulse mode is selected by setting the OPMODE bit in the TMRx_CTRL1 register, and this can stop the counter automatically at the next update event UEV.

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

In upcounting: $CNT < CCx \leq AR$ (especially, $0 < CCx$)

In downcounting: $CNT > CCx$

Figure 10-43 Example of One-pulse Mode



For example, if users want to generate a positive pulse with a length of t_{PULSE} on OC1 once a rising edge is detected on the TI2 input pin and after a delay t_{DELAY} .

If TI2FP2 is used as trigger 1:

- Map TI2FP2 to TI2 by writing C2SEL = '01' in the TMRx_CCM1 register.
- Write C2P = '0' in the TMRx_CCE register to enable TI2FP2 to detect a rising edge.
- Write TRGSEL = '110' in the TMRx_SMC register to make TI2FP2 the trigger of the slave mode controller (TRGI).
- Write SMSEL = '110' in the TMRx_SMC register (trigger mode), and TI2FP2 is used to enable the counter.

The OPM waveform is defined by the value written to the compare registers (the clock frequency and the counter prescaler should be taken into account).

- t_{DELAY} is defined by the value written to the TMRx_CC1 register.
- t_{PULSE} is defined by the comparison result of auto-reload value and the compare value (TMRx_AR - TMRx_CC1).
- Assume that users want to generate a waveform with a transition from '0' to '1' when a compare match occurs, and a transition from '1' to '0' when the counter reaches the preload value. First, set OC1MODE = '111' in the TMRx_CCM1 register to enter PWM mode 2. Enable the desired preload registers by setting OC1PEN = '1' in the TMRx_CCM1 register and the ARPEN bit in the TMRx_CTRL1 register. Afterwards, write compare value in the TMRx_CC1 register, write auto-reload value in the TMRx_AR register, modify the UEVG bit to generate an update event, and wait for an external trigger event on TI2. In this example, C1P = '0'.

In this example, the DIR and CMSEL bits in the TMRx_CTRL1 register should be low. Since only one pulse is needed, '1' must be written to the OPMODE bit in the TMRx_CTRL1 register to stop the counter at the next update event (when the counter rolls over to 0 from the auto-reload value).

Special case: OCx fast enable

In one-pulse mode, the counter is enabled by setting the CNTEN bit on the edge detection logic of TIx input pin. Then, the comparison between the counter and the compare value makes output toggle. However, since several clock cycles are needed for these operations, it limits the minimum delay tDELAY users can get.

The OCxFEN bit in the TMRx_CCMx register can be set to output a waveform with the minimum delay. In this case, OCxREF (and OCx) are forced to respond to the stimulus, instead of relying on the comparison result. The waveform outputted is the same as the one being compared and matched. OCxFEN acts only if the channel is configured as PWM1 or PWM2 mode.

10.2.3.11 Clearing OCxREF Signal on an External Event

For a given channel, a high level of the ETRF input can lower the OCxREF signal if the OCxDIS bit in the TMRx_CCMx register is set. The OCxREF signal remains low until the next update event, UEV, occurs.

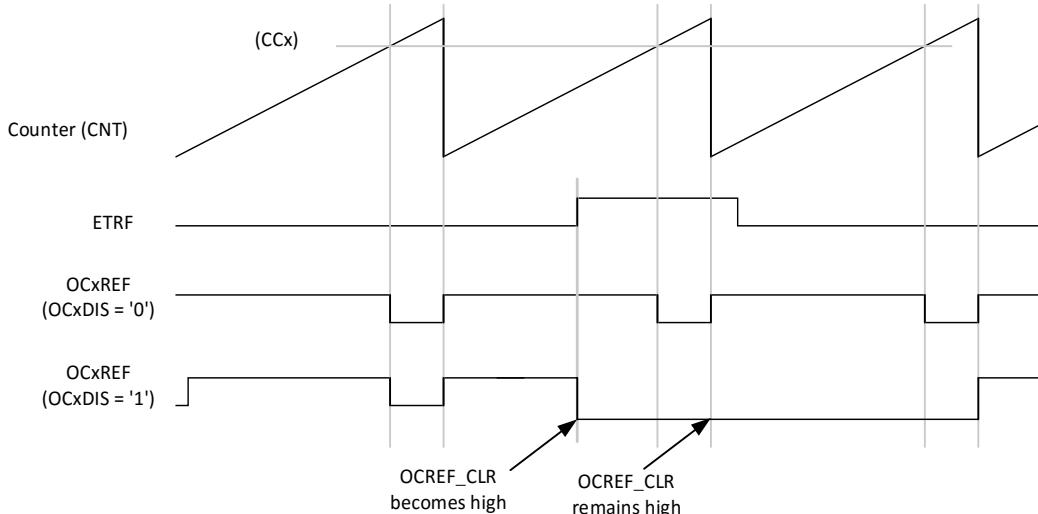
This function can only be used in output compare and PWM modes, and does not work in the forced output mode.

For example, the OCxREF signal can be connected to a comparator output for current control. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept OFF: ETD[1:0] = '00' in the TMRx_SMC register.
2. The external clock mode 2 must be disabled: ECLKEN = '0' in the TMRx_SMC register.
3. The external trigger polarity (ETRGP) and the external trigger filter (ETDF) can be configured according to users' needs.

Figure 10-44 shows how OCxREF signal behaves to respond to different OCxDIS values when the ETRF input becomes high. In this example, the timer TMRx is programmed in PWM mode.

Figure 10-44 Clearing OCxREF in TMRx



10.2.3.12 Encoder Interface Mode

To select the encoder interface mode, write SMSEL = 001 in the TMRx_SMC register if the counter is counting on TI2 edges only; write SMSEL = 010 if it is counting on TI1 edges only; write SMSEL = 011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the C1P and C2P bits in the TMRx_CCE register. Users can program the input filter as well when it is needed.

The two inputs, TI1 and TI2, are used as the interface of incremental encoder. Please refer to [Table 10-2](#). If the counter is enabled (CNTEN = '1' in the TMRx_CTRL1 register), it is clocked by each valid transition on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after input filter and polarity selection. If not filtered and not inverted, TI1FP1 = TI1 and TI2FP2 = TI2. Count pulses and direction

signal are generated according to the transition sequence of the two inputs. The counter counts up or down according to the transition sequence of the two inputs, and at the same time the DIR bit in the TMRx_CTRL1 register is set accordingly by hardware.

The DIR bit is re-calculated at each transition on any input (TI1 or TI2), no matter the counter is counting on TI1 only, TI2 only, or both TI1 and TI2.

Encoder interface mode basically acts like an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TMRx_AR register (0 to AR or ARR down to 0, depending on the direction). Hence, the TMRx_AR register must be configured before the counting starts. Likewise, features of the capture, comparator, prescaler, and trigger output work as usual.

In this mode, the counter is modified automatically according to the speed and the direction of the incremental encoder. Its content, therefore, always indicates the position of the encoder. The counting direction corresponds to the rotation direction of the connected sensor. Table 10-2 lists all the possible combinations, assuming that TI1 and TI2 do not switch at the same time.

Table 10-2 Counting Direction and Encoder Signals

Active Edge	Level on Opposite Signal (TI1FP1 to TI2, TI2FP2 to TI1)	TI1FP1 Signal		TI2FP2 Signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
Counting on TI2 only	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output signal, which indicates the mechanical zero position, can be connected to an external interrupt input to trigger a counter reset.

Figure 10-45 gives an example of counter operation, showing counting signal generation and direction control. It also shows how input jitter is compensated when both edges are selected. Jitters might occur if the sensor is positioned near to one of the switching points. For example, assume that the configuration is the following:

- C1SEL = '01' (TMRx_CCM1 register, IC1FP1 is mapped on TI1.)
- C2SEL = '01' (TMRx_CCM1 register, IC2FP2 is mapped on TI2.)
- C1P = '0' (TMRx_CCE register, IC1FP1 not inverted, IC1FP1 = TI1)
- C2P = '0' (TMRx_CCE register, IC2FP2 not inverted, IC2FP2 = TI2)
- SMSEL = '011' (TMRx_SMC register, all inputs are active on both rising and falling edges.)
- CNTEN = '1' (TMRx_CTRL1 register, counter enabled)

Figure 10-45 Example of Counter Operation in Encoder Interface Mode

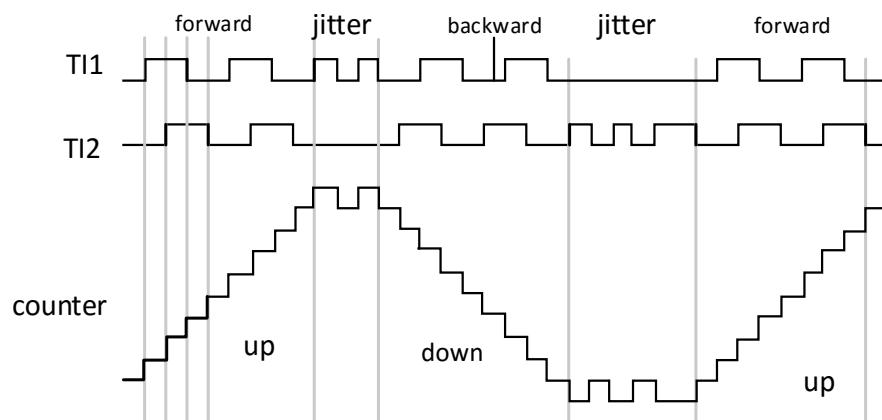
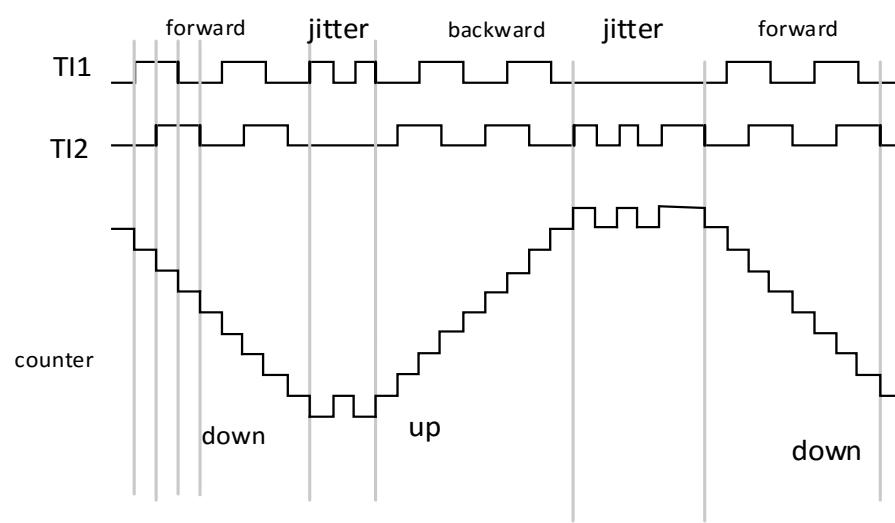


Figure 10-46 is an example of counter behavior when IC1FP1 polarity is inverted (the same configuration as the above example, except for C1P = '1').

Figure 10-46 Example of Encoder Interface Mode with IC1FP1 Polarity Inverted



The timer, when configured as encoder interface mode, provides information on the sensor's current position. Using the second timer configured in capture mode can measure the interval period between two encoder events to obtain the dynamic information (speed, acceleration, and deceleration). The encoder output which indicates the mechanical zero can be used for this purpose. Based on the interval between two events, the counter can also be read regularly. If possible, users can latch the counter value into the third input capture register (the capture signal must be periodic and can be generated by another timer). It is also possible to read its value through a DMA request generated by a real-time clock.

10.2.3.13 Timer Input XOR Function

The TI1SEL bit in the TMRx_CTRL2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, which includes three input pins, TMRx_CH1, TMRx_CH2, and TMRx_CH3.

The XOR output can be used with all the timer input functions, such as trigger or input capture. An example of this feature used to interface Hall sensors is listed in [Section 10.4.3.18](#).

10.2.3.14 Timer and External Trigger Synchronization

The TMRx timer can be synchronized with an external trigger in several modes: reset mode, gated mode, and trigger mode.

Slave mode: Reset mode

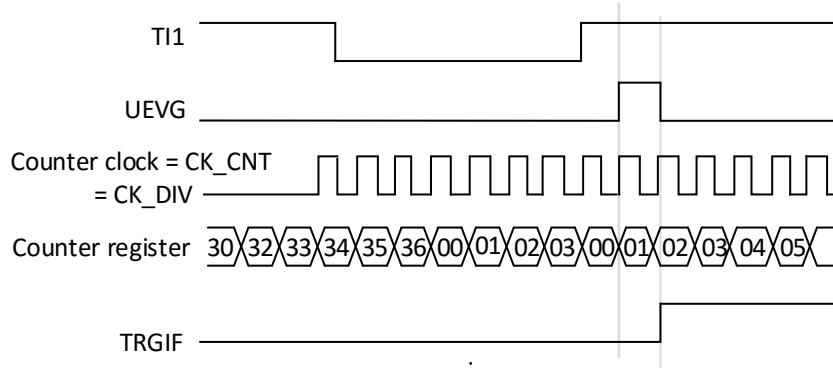
The counter and its prescaler can be reinitialized when a trigger input event occurs. In this case, if the UEVRS bit of the TMRx_CTRL1 register is low, an update event UEV is generated. Then, all the preloaded registers (TMRx_AR, TMRx_CCx) will be updated.

In the following example, the up counter is cleared due to a rising edge on TI1 input:

- Configure channel 1 to detect the rising edges on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bit selects the input capture source only, that is, C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 0 in the TMRx_CCE register to confirm the polarity (and detect rising edges only).
- Configure the timer as reset mode by writing SMSEL = 100 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Start the counter by writing CNTEN = 1 in the TMRx_CTRL1 register.

The counter starts counting on the internal clock, and runs normally until TI1 rising edge occurs; at this time, the counter is cleared and restarts from 0. In the meantime, the trigger flag (the TRGIF bit in the TMRx_STS register) is set, and an interrupt request, or a DMA request can be sent depending on the configuration of the TRGIE bit in the TMRx_DIE register (interrupt enable) and the TRGDE bits in TMRx_DIE register (DMA enable). Figure 10-47 shows the behavior when the auto-reload register TMRx_AR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is determined by the resynchronization circuit on TI1 input.

Figure 10-47 Control Circuit in Reset Mode



Slave mode: Gated mode

The counter can be enabled according to the selected level of input.

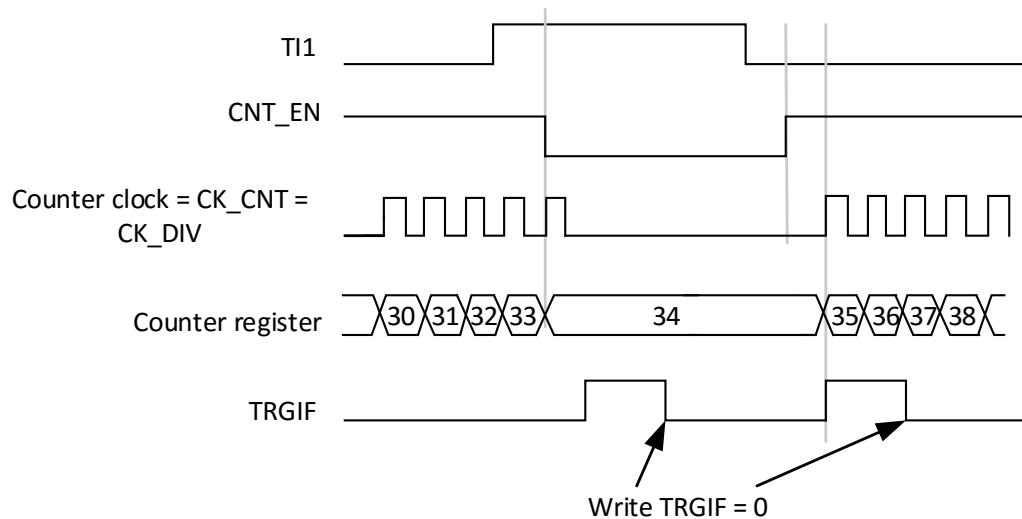
In the following example, the counter counts up only when TI1 input is low:

- Configure channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bits select the input capture source. Set C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 1 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as gated mode by writing SMSEL = 101 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Enable the counter by writing CNTEN = 1 in the TMRx_CTRL1 register. In gated mode, the counter cannot be enabled if CNTEN = 0, no matter what the trigger input level is.

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TRGIF flag in the TMRx_STS register is set both when the counter starts and stops.

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

Figure 10-48 Control Circuit in Gated Mode



Slave mode: Trigger mode

The counter can be enabled according to the selected level of input.

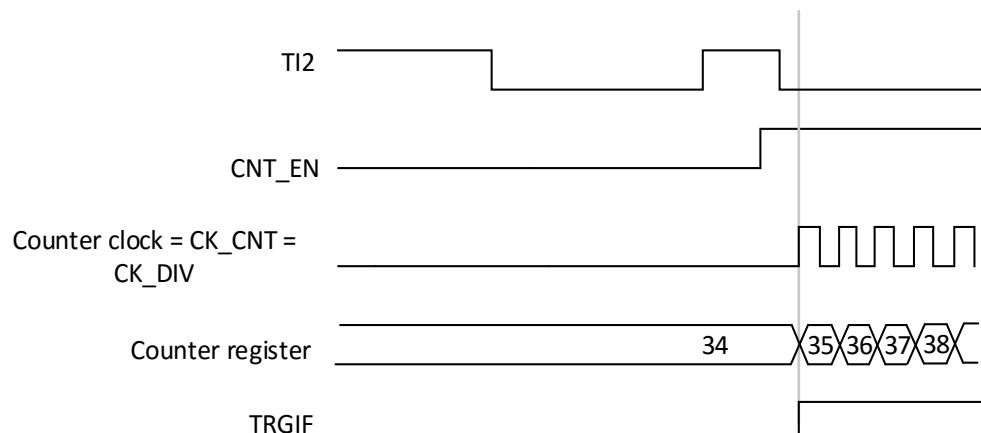
In the following example, the counter starts to count up at rising edge on TI2 input:

- Configure channel 2 to detect the rising edge on TI2. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C2SEL bits are only used to select the input capture source. Set C2SEL = 01 in the TMRx_CCM1 register. Write C2P = 0 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as trigger mode by writing SMSEL = 110 in the TMRx_SMC register. Select TI2 as the input source by writing TRGSEL = 110 in the TMRx_SMC register.

The counter starts counting on the internal clock as long as a rising edge occurs on TI2, and at the same time the TRGIF flag is set.

The delay between the rising edge on TI2 and enabling counting is determined by the resynchronization circuit on TI2 input.

Figure 10-49 Control Circuit in Trigger Mode



Slave mode: External Clock Mode 2 + Trigger Mode

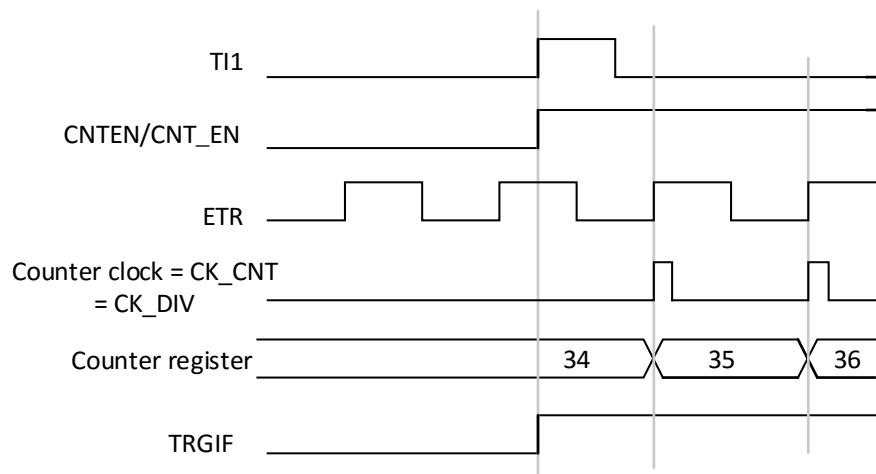
The external clock mode 2 can be used with another slave mode (except for external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be trigger input in reset mode, gated mode, or trigger mode. Selecting ETR as TRGI through the TRGSEL bits in the TMRx_SMC register is not recommended.

In the following example, as soon as a rising edge of TI1 occurs, the counter counts up once at each rising edge of the ETR signal:

1. Configure the external trigger input circuit through the TMRx_SMC register:
 - ETDF = 0000: No filter
 - ETD = 00: Prescaler is disabled.
 - ETRGP = 0: Detect the rising edges on ETR and set ECLKEN = 1 to enable the external clock mode 2.
2. Configure channel 1 as follows to detect the rising edges on TI:
 - IC1DF = 0000: No filter
 - The capture prescaler is not used for triggering and does not need to be configured.
 - Set C1SEL = 01 in the TMRx_CCM1 register to select the input capture source
 - Set C1P = 0 in the TMRx_CCE register to confirm polarity (and detect rising edge only)
3. Configure the timer as trigger mode by writing SMSEL = 110 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.

A rising edge on TI1 enables the counter and sets the TRGIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is determined by the resynchronization circuit on ETRP input.

Figure 10-50 Control Circuit in External Clock Mode 2 + Trigger Mode



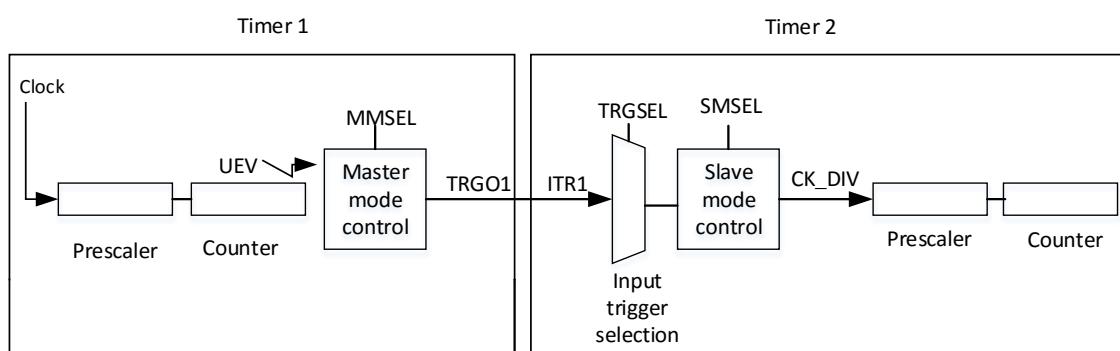
10.2.3.15 Timer Synchronization

All TMRx timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can reset, start, stop, or clock the counter of another timer in slave mode.

Figure 10-51 shows an overview of the trigger selection and the master mode selection blocks.

Using one timer as the prescaler for another timer

Figure 10-51 Master/Slave Timer Example



For example, the user can configure Timer 1 to act as a prescaler for Timer 2. Please refer to [Figure 10-51](#), and use the following procedure:

- Configure Timer 1 as master mode so that it can output a periodic trigger signal on each update event UEV. If MMSEL = '010' in the TMR1_CTRL2 register, a rising edge is outputted on TRGO1 each time when an update event is generated.
- To connect the TRGO1 output of Timer 1 to Timer 2, set TRGSEL = '000' in the TMR2_SMC register, and Timer 2 must be configured in slave mode using ITR1 as internal trigger.
- Then, set the slave mode controller in external clock mode 1 (SMSEL = 111 in the TMR2_SMC register). In this way, Timer 2 can be clocked by the periodic rising edge (Timer 1 counter overflow) trigger signal of Timer 1.
- Finally, both timers must be enabled by setting their corresponding CNTEN bits respectively (TMRx_CTRL1 register).

Note: If OCx is selected on Timer 1 as trigger output (MMSEL = 1xx), its rising edge is used to clock the counter of Timer 2.

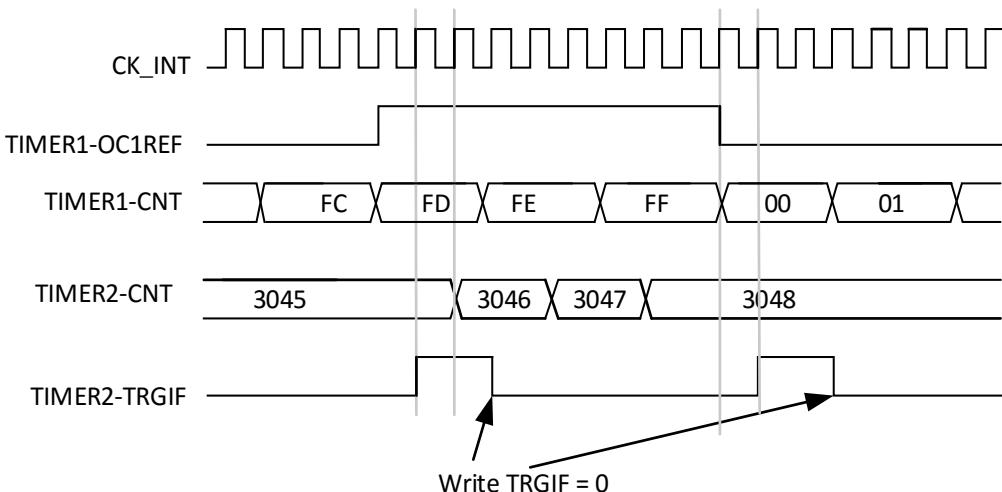
Using one timer to enable another timer

In this example, the enable of Timer 2 is controlled by the output compare of Timer 1. Please refer to [Figure 10-51](#) for the connections. Timer 2 counts on the divided internal clock only when OC1REF of Timer 1 is high. Both counter clock frequencies are obtained through the prescaler which divides CK_INT by 3 ($f_{CK_CNT} = f_{CK_INT}/3$).

- Configure Timer 1 as master mode to send its output compare reference signal (OC1REF) as trigger output (MMSEL = 100 in the TMR1_CTRL2 register).
- Configure the Timer 1 OC1REF waveform (TMR1_CCM1 register)
- Configure Timer 2 to get the input trigger from Timer 1 (TRGSEL = 000 in the TMR2_SMC register).
- Configure Timer 2 as gated mode (SMSEL = 101 in the TMR2_SMC register).
- Set CNTEN = 1 in the TMR2_CTRL1 register to enable Timer 2.
- Set CNTEN = 1 in the TMR1_CTRL1 register to start Timer 1.

Note: The Timer 2 clock is not synchronized with Timer 1 clock. This mode only affects the Timer 2 counter enable signal.

Figure 10-52 Timer 1 OC1REF Controls Timer 2

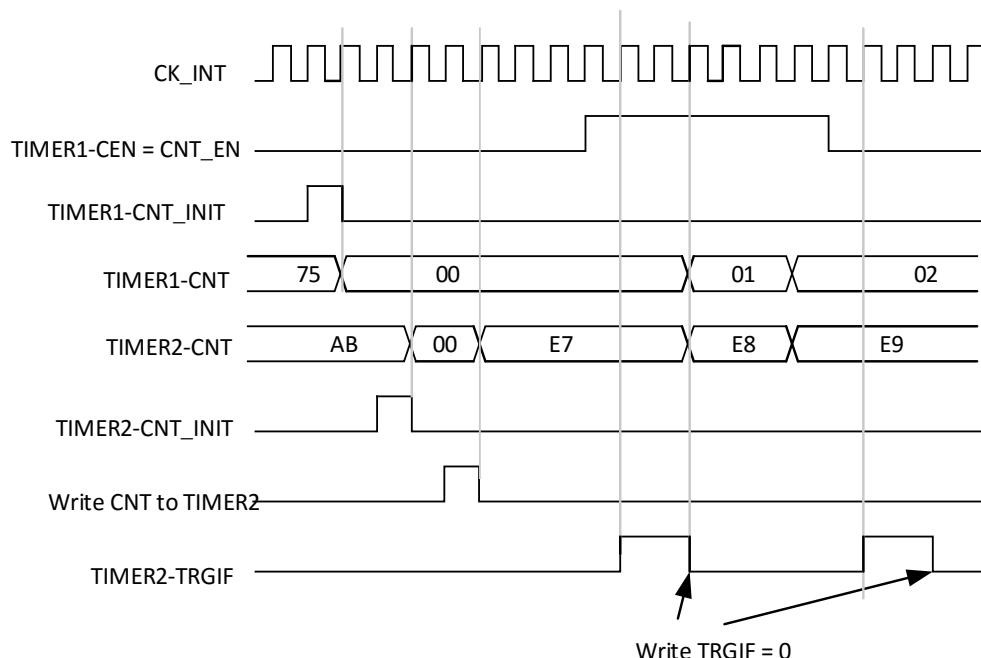


In [Figure 10-52](#), Timer 2 counter and prescaler are not initialized before Timer 2 is started, so they start counting from their current values. It is possible to start from a given value by resetting both timers before starting Timer 1. Then write any value desired in the timer counters. The timers can be reset just by writing the UEVG bit in the TMRx_EVEG registers.

In the following example, Timer 1 and Timer 2 need to be synchronized. Timer 1 is the master and starts from 0. Timer 2 is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. Timer 2 stops when Timer 1 is disabled by writing '0' to the CNTEN bit in the TMR1_CTRL1 register:

- Configure Timer 1 as master mode to send its output compare 1 reference signal (OC1REF) as trigger output (MMSEL = 100 in the TMR1_CTRL2 register).
- Configure the Timer 1 OC1REF waveform (TMR1_CCM1 register).
- Configure Timer 2 to get the input trigger from Timer 1 (TRGSEL = 000 in the TMR2_SMC register).
- Configure Timer 2 as gated mode (SMSEL = 101 in TMR2_SMC register).
- Set UEVG = '1' in the TMR1_EVEG register to reset Timer 1.
- Set UEVG = '1' in the TMR2_EVEG register to reset Timer 2.
- Initialize Timer 2 to 0xE7 by writing '0xE7' to the timer 2 counter (TMR2_CNT).
- Set CNTEN = '1' in the TMR2_CTRL1 register to enable Timer 2.
- Set CNTEN = '1' in the TMR1_CTRL1 register to start Timer 1.
- Set CNTEN = '0' in the TMR1_CTRL1 register to stop Timer 1.

Figure 10-53 Control Timer 2 by Enabling Timer 1

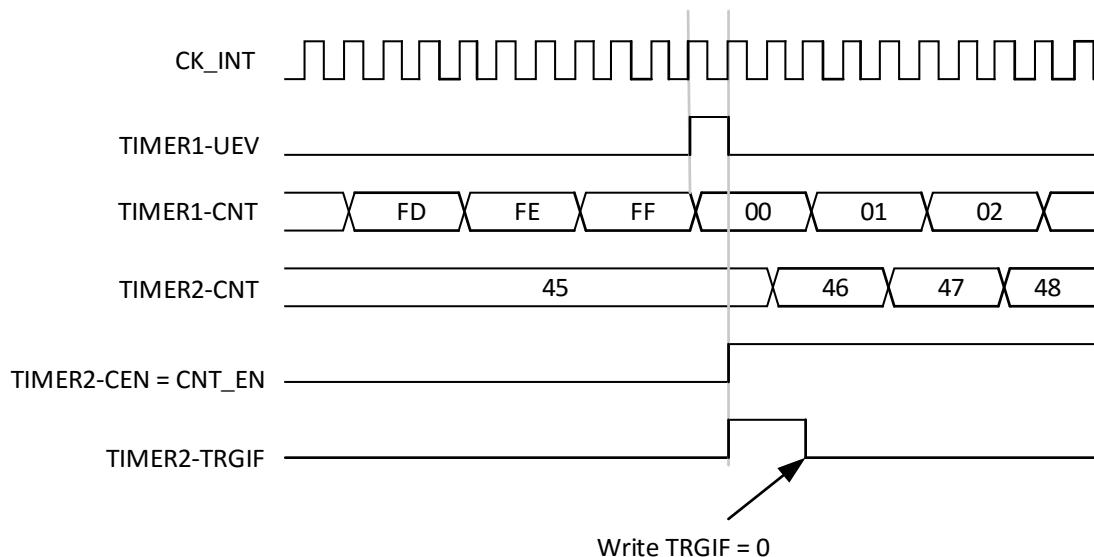


Using one timer to start another timer

In this example, Timer 1 update event is used to enable Timer 2. Please refer to [Figure 10-51](#) for the connections. Timer 2 starts counting from its current value (can be nonzero) on the divided internal clock once an update event is generated by Timer 1. When Timer 2 receives the trigger signal, its CNTEN bit is automatically set, and the counter starts counting until writing '0' to the CNTEN bit in the TMR2_CTRL1 register. Both counter clock frequencies are divided by the prescaler which divides CK_INT by 3 ($f_{CK_CNT} = f_{CK_INT}/3$).

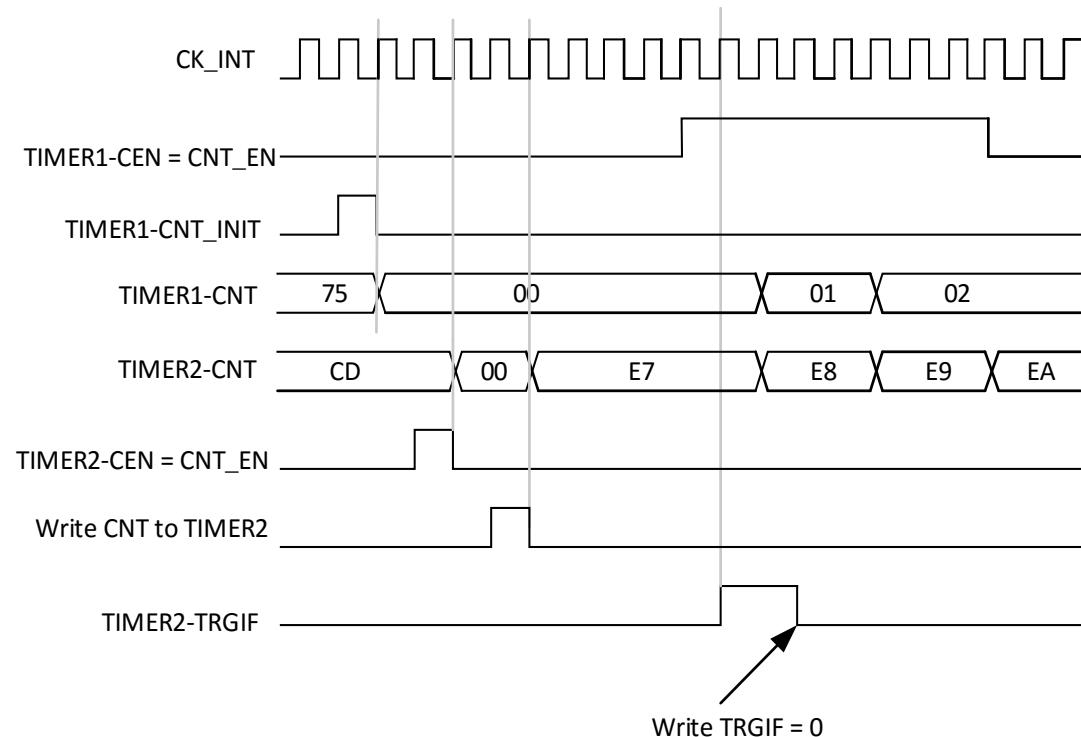
- Configure Timer 1 as master mode to send its update event (UEV) as trigger output (MMSEL = 010 in the TMR1_CTRL2 register).
- Configure the Timer 1 period (TMR1_AR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TRGSEL = 000 in the TMR2_SMC register).
- Configure Timer 2 as trigger mode (SMSEL = 110 in the TMR2_SMC register)
- Set CNTEN = 1 in the TMR1_CTRL1 register to start Timer 1.

Figure 10-54 Using Timer 1 Update to Trigger Timer 2



As in the above example, both counters can be initialized before starting counting. [Figure 10-55](#) shows the behavior with the same configuration as the above example, but using trigger mode instead of gated mode (SMSEL = 110 in the TMR2_SMC register).

Figure 10-55 Using Timer 1 Enable to Trigger Timer 2



Using one timer as the prescaler for another timer

In this example, Timer 1 is used as the prescaler of Timer 2. Please refer to [Figure 10-51](#) for the connections, and the configuration is as follows:

- Configure Timer 1 as master mode to send its update event (UEV) as trigger output (MMSEL = 010 in the TMR1_CTRL2 register).
- Configure the Timer 1 period (TMR1_AR register).
- Configure Timer 2 to get the input trigger from Timer 1 (TRGSEL = 000 in the TMR2_SMC register).
- Configure Timer 2 to use external clock mode (SMSEL = 111 in the TMR2_SMC register)

- Set CNTEN = 1 in the TMR1_CTRL2 register to start Timer 2.
- Set CNTEN = 1 in the TMR1_CTRL1 register to start Timer 1.

Starting two timers synchronously by an external trigger

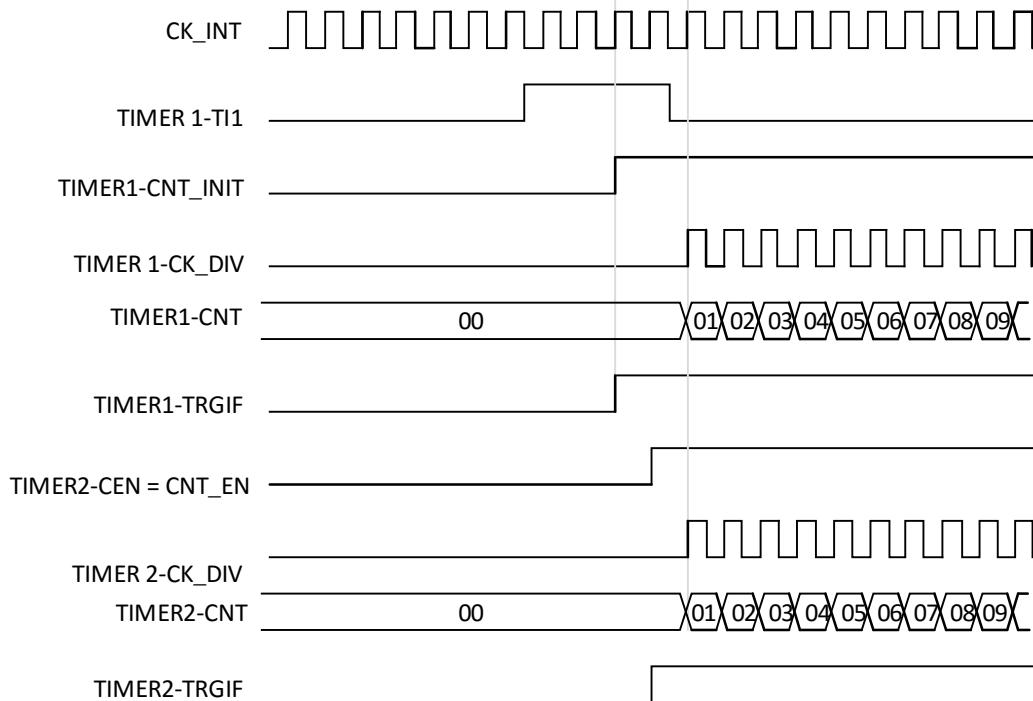
In this example, when Timer 1 TI1 input rises, enable Timer 1, and at the same time Timer 2 is enabled. Please refer to [Figure 10-51](#). To ensure that the counters are aligned, Timer 1 must be configured in master/slave mode (slave to TI1, master to Timer 2):

- Configure Timer 1 as master mode to send its enable as trigger output (MMSEL = '001' in the TMR1_CTRL2 register).
- Configure Timer 1 as slave mode to get the input trigger from TI1 (TRGSEL = '100' in the TMR1_SMC register).
- Configure Timer 1 as trigger mode (SMSEL = '110' in the TMR1_SMC register).
- Configure Timer 1 as master/slave mode, MSMODE = '1' in the TMR1_SMC register.
- Configure Timer 2 to get the input trigger from Timer 1 (TRGSEL = 000 in the TMR2_SMC register).
- Configure Timer 2 as trigger mode (SMSEL = '110' in the TMR2_SMC register). When a rising edge occurs on TI1 of Timer 1, both timers start counting synchronously on the internal clock, and both TRGIF flags are set.

Note: In this example, both timers are initialized before starting (by setting the corresponding UEVG bits). Both counters start from 0, but an offset can be inserted between them by writing any of the counter registers (TMRx_CNT).

In [Figure 10-56](#), there is a delay between CNT_EN and CK_DIV on Timer 1 in master/slave mode.

Figure 10-56 Using Timer 1 TI1 Input to Trigger Timer 1 and Timer 2



10.2.3.16 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter either continues to work normally or stops, depending on DBG_TMRx_STOP configuration in DBG module. For more details, please refer to [Section 22.2.2](#).

10.2.4 TMRx Registers Description

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

In Table 10-3, all the TMRx registers are mapped to a 16-bit addressable space.

Table 10-3 TMRx – Register Table and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0x00	TMRx_CTRL1	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x04	TMRx_CTRL2	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x08	TMRx_SMC	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x0C	TMRx_DIE	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x10	TMRx_STS	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x14	TMRx_EVEG	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x18	TMRx_CCM1 Output Compare mode	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0
0x1C	TMRx_CCM1 Input Capture mode	Reserved																			0	0	0	0	0
	Reset Value																				0	0	0	0	0

	TMRx_CCM2 Input Capture mode	Reserved								IC4DF[3:0]				IC4DIV[1:0]		C4SI[1:0]				IC3DF[3:0]				IC3DIV[1:0]		C3SEL[1:0]						
	Reset Value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x20	TMRx_CCE	Reserved																C4P	C4EN	Reser	C3P	C3EN	Reser	C2P	C2EN	Reser	C1P	C1EN				
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0				
0x24	TMRx_CNT	CNT[31:16]																CNT[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x28	TMRx_DIV	Reserved																DIV[15:0]														
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0				
0x2C	TMRx_AR	CNT[31:16]																AR[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x34	TMRx_CC1	CC1[31:16]																CC1[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x38	TMRx_CC2	CC2[31:16]																CC2[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x3C	TMRx_CC3	CC3[31:16]																CC3[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x40	TMRx_CC4	CC4[31:16]																CC4[15:0]														
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x48	TMRx_DMAC	Reserved																DBLEN[4:0]				Reserved		ADDR[4:0]				0	0	0	0	0
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x4C	TMRx_DMABA	Reserved																DMABA[15:0]														
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.2.4.1 Control Register 1 (TMRx_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				PMEN	CLKDIV [1:0]	ARPE_N	CMSEL[1:0]	DIR	OPM_ODE	UEVR_S	UEVD_IS	CNTE_N			
res				rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bit 15:11	Reserved. Always read as 0.
Bit 10	PMEN: Plus Mode Enable Enable TMRx plus mode. In this mode, TMRx_CNT, TMRx_AR, and TMRx_CC1/2/3/4 are expanded from 16 bits to 32 bits. 0: Plus mode is not enabled. 1: Plus mode is enabled. <i>Note: Only TMR2 and TMR5 have this function. This bit is invalid in other TMRs.</i>

Bit 9:8	CLKDIV[1:0]: Clock division Define the division ratio between the timer clock (CK_INT) frequency and sampling frequency used by the digital filters (ETR, TIx). 00: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved
Bit 7	ARPEN: Auto-reload preload enable 0: TMRx_AR register is not buffered. 1: TMRx_AR register is buffered.
Bit 6:5	CMSEL[1:0]: Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set both when the counter is counting up or down. <i>Note: It is not allowed to switch from edge-aligned mode to center-aligned mode when the counter is enabled (CNTEN = 1).</i>
Bit 4	DIR: Direction 0: The counter counts up. 1: The counter counts down. <i>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode.</i>
Bit 3	OPMODE: One pulse mode 0: The counter does not stop at update event. 1: The counter stops at the next update event.
Bit 2	UEVRS: Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt or DMA request if update interrupt or DMA request is enabled: <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UEVG bit - An update is generated from the slave mode controller. 1: Only counter overflow/underflow generates an update interrupt or DMA request if update interrupt or DMA request is enabled.
Bit 1	UEVDIS: Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV is enabled. An update event (UEV) is generated by any of the following events: <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UEVG bit - An update is generated from the slave mode controller. Buffered registers are loaded with their preload values. 1: UEV is disabled. Update events are not generated, and shadow registers (AR, DIV, and CCx) keep their values. The counter and the prescaler are reinitialized if the UEVG bit is set or the slave mode controller sends a hardware reset.
Bit 0	CNTEN: Counter enable 0: The counter is disabled. 1: The counter is enabled. <i>Note: External clock, gated mode, and encoder mode can work only after the CNTEN bit is previously set by software. Trigger mode can set the CNTEN bit automatically by hardware.</i> CNTEN is cleared automatically in one-pulse mode when an update event occurs.

10.2.4.2 Control Register 2 (TMRx_CTRL2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
								TI1S EL		MMSEL[2:0]	CDS EL					
res								rw	rw	rw	rw	rw	rw	res		
Bit 15:8		Reserved. Always read as 0.														
Bit 7		TI1SEL: TI1 selection 0: TMRx_CH1 pin is connected to TI1 input. 1: TMRx_CH1, TMRx_CH2, and TMRx_CH3 pins are connected to the TI1 input after XORed. Please refer to Section 10.4.3.18 .														
Bit 6:4		MMSEL[2:0]: Master mode selection The 3 bits select the synchronization information (TRGO) to be sent from master mode to slave timers. The possible combinations are as follows: 000: Reset – The UEVG bit in the TMRx_EVEG register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller is in reset mode), the signal on TRGO is delayed compared with the actual reset. 001: Enable – The counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by a logic OR between CNTEN control bit and the trigger input signal in gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, unless the master/slave mode is selected (Please refer to the MSMODE bit description in the TMRx_SMC register). 010: Update – The update event is selected as trigger output (TRGO). For instance, a master timer can be used as a prescaler for a slave timer. 011: Compare Pulse – After a capture or a compare match occurs, the trigger output sends a positive pulse (TRGO) when the C1IF flag is to be set (even if it is already high). 100: Compare – OC1REF signal is used as trigger output (TRGO). 101: Compare – OC2REF signal is used as trigger output (TRGO). 110: Compare – OC3REF signal is used as trigger output (TRGO). 111: Compare – OC4REF signal is used as trigger output (TRGO).														
Bit 3		CDSEL: Capture/Compare DMA selection 0: CCx DMA request is sent when CCx event occurs. 1: CCx DMA request is sent when update event occurs.														
Bit 2:0		Reserved. Always read as 0.														

10.2.4.3 Slave Mode Control Register (TMRx_SMC)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ETR GP	ECL KEN	ETD[1:0]		ETDF[3:0]				MSM ODE	TRGSEL[2:0]				Reser ved	SMSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	
Bit 15		ETRGP: External trigger polarity This bit selects whether ETR or inverted ETR is used for trigger operations. 0: ETR is non-inverted and active at high level or rising edge. 1: ETR is inverted and active at low level or falling edge.														

Bit 14	<p>ECLKEN: External clock enable This bit enables external clock mode 2. 0: External clock mode 2 is disabled 1: External clock mode 2 is enabled. The counter is clocked by any active edge on the ETRF signal.</p> <p><i>Note 1: Setting the ECLKEN bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMSEL = 111 and TRGSEL = 111).</i></p> <p><i>Note 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode, and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TRGSEL bits must not be 111).</i></p> <p><i>Note 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</i></p>																
Bit 13:12	<p>ETD[1:0]: External trigger prescaler External trigger signal ETRP frequency must be 1/4 of CK_INT frequency at most. A prescaler can be used to reduce ETRP frequency when inputting faster external clocks. 00: Prescaler is OFF. 01: ETRP frequency is divided by 2. 10: ETRP frequency is divided by 4. 11: ETRP frequency is divided by 8.</p>																
Bit 11:8	<p>ETDF[3:0]: External trigger filter The bits define the frequency used to sample TETRP signal and the length of ETRP digital filter. In fact, the digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output:</p> <table> <tbody> <tr><td>0000: No filter, sampling is done at f_{DTS}</td><td>1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$</td></tr> <tr><td>0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$</td><td>1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$</td></tr> <tr><td>0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$</td><td>1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$</td></tr> <tr><td>0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$</td><td>1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$</td></tr> <tr><td>0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$</td><td>1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$</td></tr> <tr><td>0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$</td><td>1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$</td></tr> <tr><td>0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$</td><td>1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$</td></tr> <tr><td>0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$</td><td>1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$</td></tr> </tbody> </table>	0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$	0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$	1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$	0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$	1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$	0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$	1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$	0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$	1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$	0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$	1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$	0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$	1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$	0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$	1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$
0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$																
0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$	1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$																
0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$	1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$																
0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$	1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$																
0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$	1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$																
0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$	1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$																
0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$	1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$																
0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$	1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$																
Bit 7	<p>MSMODE: Master/Slave mode 0: No effect 1: The event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful when synchronizing several timers on a single external event.</p>																
Bit 6:4	<p>TRGSEL[2:0]: Trigger selection The bits select the trigger input used to synchronize the counter.</p> <table> <tbody> <tr><td>000: Internal Trigger 0 (ITR0), TMR1</td><td>100: TI1 Edge Detector (TI1F_ED)</td></tr> <tr><td>001: Internal Trigger 1 (ITR1), TMR2</td><td>101: Filtered Timer Input 1 (TI1FP1)</td></tr> <tr><td>010: Internal Trigger 2 (ITR2), TMR3</td><td>110: Filtered Timer Input 2 (TI2FP2)</td></tr> <tr><td>011: Internal Trigger (ITR3), TMR4</td><td>111: External Trigger Input (ETRF)</td></tr> </tbody> </table> <p>Please refer to Table 10-4 for the details on ITRx for each timer.</p> <p><i>Note: These bits must be changed only when they are not used (e.g. when SMSEL = 000) to avoid wrong edge detections at transitions.</i></p>	000: Internal Trigger 0 (ITR0), TMR1	100: TI1 Edge Detector (TI1F_ED)	001: Internal Trigger 1 (ITR1), TMR2	101: Filtered Timer Input 1 (TI1FP1)	010: Internal Trigger 2 (ITR2), TMR3	110: Filtered Timer Input 2 (TI2FP2)	011: Internal Trigger (ITR3), TMR4	111: External Trigger Input (ETRF)								
000: Internal Trigger 0 (ITR0), TMR1	100: TI1 Edge Detector (TI1F_ED)																
001: Internal Trigger 1 (ITR1), TMR2	101: Filtered Timer Input 1 (TI1FP1)																
010: Internal Trigger 2 (ITR2), TMR3	110: Filtered Timer Input 2 (TI2FP2)																
011: Internal Trigger (ITR3), TMR4	111: External Trigger Input (ETRF)																
Bit 3	Reserved. Always read as 0.																

Bit 2:0	SMSEL[2:0]: Slave mode selection When external signals are selected, the active edge of the trigger signal (TRGI) is relevant to the selected external input polarity (Please refer to input control register and control register description.)
	000: Slave mode is disabled - If CNTEN = 1, the prescaler is clocked directly by the internal clock.
	001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level.
	010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level.
	011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input.
	100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register signal.
	101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) once the trigger becomes low. Both start and stop of the counter are controlled.
	110: Trigger Mode - The counter starts at a rising edge of the trigger input TRGI (but it is not reset). Only the start of the counter is controlled.
	111: External Clock Mode 1 - Rising edges of the selected trigger input (TRGI) clocks the counter.
	<i>Note: The gated mode must not be used if TI1F_EN is selected as the trigger input (TRGSEL = 100). This is because, TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger input signal.</i>

Table 10-4 TMRx Internal Trigger Connection (Note)

Slave Timer	ITR0 (TRGSEL = 000)	ITR1 (TRGSEL = 001)	ITR2 (TRGSEL = 010)	ITR3 (TRGSEL = 011)
TMR2	TMR1	TMR8	TMR3	TMR4
TMR3	TMR1	TMR2	TMR5	TMR4
TMR4	TMR1	TMR2	TMR3	TMR8
TMR5	TMR2	TMR3	TMR4	TMR8

Note: When a corresponding timer is not present in the product, the corresponding trigger ITRx is not available.

10.2.4.4 DMA/Interrupt Enable Register (TMRx_DIE)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	TRG DE	Reserve d	C4D E	C3D E	C2D E	C1D E	UEV DE	Reserve d	TRG IE	Reserve d	C4I E	C3I E	C2I E	C1I E	UEV IE

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15		Reserved. Always read as 0.													
Bit 14		TRGDE: Trigger DMA request enable 0: Trigger DMA request is disabled. 1: Trigger DMA request is enabled.													
Bit 13		Reserved. Always read as 0.													
Bit 12		C4DE: Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request is disabled. 1: Capture/Compare 4 DMA request is enabled.													
Bit 11		C3DE: Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request is disabled. 1: Capture/Compare 3 DMA request is enabled.													

Bit 10	C2DE: Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request is disabled. 1: Capture/Compare 2 DMA request is enabled.
Bit 9	C1DE: Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request is disabled. 1: Capture/Compare 1 DMA request is enabled.
Bit 8	UEVDE: Update DMA request enable 0: Update DMA request is disabled. 1: Update DMA request is enabled.
Bit 7	Reserved. Always read as 0.
Bit 6	TRGIE: Trigger interrupt enable 0: Trigger interrupt is disabled. 1: Trigger interrupt is enabled.
Bit 5	Reserved. Always read as 0.
Bit 4	C4IE: Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt is disabled. 1: Capture/Compare 4 interrupt is enabled.
Bit 3	C3IE: Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt is disabled. 1: Capture/Compare 3 interrupt is enabled.
Bit 2	C2IE: Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt is disabled. 1: Capture/Compare 2 interrupt is enabled.
Bit 1	C1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt is disabled. 1: Capture/Compare 1 interrupt is enabled.
Bit 0	UEVIE: Update interrupt enable 0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.2.4.5 Status Register (TMRx_STS)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	C4OF	C3OF	C2OF	C1OF	Reserved	TRGIF	Reserved	C4IF	C3IF	C2IF	C1IF	UEVIF			

Bit 15:13	Reserved. Always read as 0.
Bit 12	C4OF: Capture/Compare 4 overcapture flag Please refer to C1OF description.
Bit 11	C3OF: Capture/Compare 3 overcapture flag Please refer to C1OF description.
Bit 10	C2OF: Capture/Compare 2 overcapture flag Please refer to C1OF description.
Bit 9	C1OF: Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured as input capture mode. It is cleared by writing '0'. 0: No overcapture is detected. 1: When the counter value is captured to the TMRx_CC1 register, C1IF flag is already set.
Bit 8:7	Reserved. Always read as 0.

Bit 6	TRGIF: Trigger interrupt flag This flag is set by hardware on trigger event (When the slave mode controller is in all modes except for gated mode, active edge is detected on TRGI input, or any edge in the gated mode). It is cleared by software. 0: No trigger event occurs. 1: Trigger interrupt is pending.
Bit 5	Reserved. Always read as 0.
Bit 4	C4IF: Capture/Compare 4 interrupt flag Please refer to C1IF description.
Bit 3	C3IF: Capture/Compare 3 interrupt flag Please refer to C1IF description.
Bit 2	C2IF: Capture/Compare 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF: Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This flag is set by hardware when the counter value matches the compare value, but not in center-aligned mode (Please refer to the CMSEL bits in the TMRx_CTRL1). It is cleared by software. 0: No match 1: The TMRx_CNT value matches the TMRx_CC1 value. If channel CC1 is configured as input mode: This bit is set by hardware on a capture. It is cleared by software or by reading the TMRx_CC1 register. 0: No input capture occurs. 1: The counter value is captured to the TMRx_CC1 register (An edge which matches the selected polarity is detected on IC1).
Bit 0	UEVIF: Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update event occurs. 1: Update interrupt is pending. This bit is set by hardware when the registers are updated: - If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when UEVG = 1 in the TMRx_EVEG register (the counter CNT is reinitialized by software). - If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when the counter CNT is reinitialized by trigger events (Please refer to the description of synchronized control registers).

10.2.4.6 Event Generation Register (TMRx_EVEG)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRG G	Res erve d	C4G	C3G	C2G	C1G	UEV G	
res								rw	res	rw	rw	rw	rw	rw	rw

Bit 15:7	Reserved. Always read as 0.
Bit 6	TRGG: Trigger generation This bit is set by software to generate a trigger event. It is cleared automatically by hardware. 0: No action 1: TRGIF = 1 in the TMRx_STS register, the corresponding interrupt and DMA is generated if enabled.
Bit 5	Reserved. Always read as 0.
Bit 4	C4G: Capture/Compare 4 generation Please refer to C1G description.

Bit 3	C3G: Capture/Compare 3 generation Please refer to C1G description.
Bit 2	C2G: Capture/Compare 2 generation Please refer to C1G description.
Bit 1	C1G: Capture/Compare 1 generation This bit is set by software to generate a capture/compare event. It is cleared automatically by hardware. 0: No action 1: A capture/compare event is generated on channel 1 CC1: If channel CC1 is configured as output: Set C1IF = 1. The corresponding interrupt and DMA request is generated if enabled. If channel CC1 is configured as input: The current value of the counter is captured to the TMRx_CC1 register. Set C1IF = 1. The corresponding interrupt and DMA request is generated if enabled. If C1IF is already '1', set C1OF = 1.
Bit 0	UEVG: Update generation This bit can be set by software, and it is automatically cleared by hardware. 0: No action 1: Re-initialize the counter and generate an update. Note that the prescaler counter is cleared, too (but the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected, or if DIR = 0 (upcounting), else it takes the TMRx_AR value if DIR = 1 (downcounting).

10.2.4.7 Capture/Compare Mode Register 1 (TMRx_CCM1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by the corresponding CxSEL bits. All the other bits of this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is in output mode, ICxx describes its function when the channel is in input mode. Attention must be given to the fact that the same bit can have a different meanings for the input stage and for the output stage.

Output compare mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 DIS	OC2MODE[2:0]	OC2 PEN	OC2 FEN	C2SEL[1:0]	OC1 DIS	OC1MODE[2:0]	OC1 PEN	OC1 FEN	C1SEL[1:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15	OC2DIS: Output compare 2 clear enable
Bit 14:12	OC2MODE[2:0]: Output compare 2 mode
Bit 11	OC2PEN: Output compare 2 preload enable
Bit 10	OC2FEN: Output compare 2 fast enable
Bit 9:8	C2SEL[1:0]: Capture/Compare 2 selection This bit defines the channel direction (input/output), and the selection of input pin: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C2SEL can be written only when the channel is disabled (C2EN = '0' in the TMRx_CCE register).</i>
Bit 7	OC1DIS: Output compare 1 clear enable 0: OC1REF is not affected by ETRF input. 1: Once high level is detected on ETRF input, clear OC1REF = 0.

Bit 6:4	<p>OC1MODE[2:0]: Output compare 1 enable The bits define the behavior of the output reference signal OC1REF, and OC1REF determines OC1 value. OC1REF is active high, while active level of OC1 is determined by the C1P bit.</p> <p>000: Frozen. The comparison between the output compare register TMRx_CC1 and the counter TMRx_CNT has no effect on OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF is forced high when the counter TMRx_CNT value matches the capture/compare register1 (TMRx_CC1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF is forced low when the counter TMRx_CNT value matches the capture/compare register1 (TMRx_CC1).</p> <p>011: Toggle. Toggle OC1REF level when TMRx_CC1 = TMRx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1— In upcounting, channel 1 is active once TMRx_CNT < TMRx_CC1, else inactive; in down counting, channel 1 is inactive once TMRx_CNT > TMRx_CC1 (OC1REF = 0), else active (OC1REF = 1).</p> <p>111: PWM mode 2—In upcounting, channel 1 is inactive once TMRx_CNT < TMRx_CC1, else active; in downcounting, channel 1 is active once TMRx_CNT > TMRx_CC1, else inactive.</p> <p><i>Note: In PWM mode 1 or PWM mode 2, OC1REF level changes only when the comparison result changes or when output compare mode switches from frozen mode to PWM mode .</i></p>
Bit 3	<p>OC1PEN: Output compare 1 preload enable 0: Preload function of TMRx_CC1 is disabled. TMRx_CC1 can be written at any time, and the new value takes effect immediately. 1: Preload function of TMRx_CC1 is enabled. Read/Write operations only access the preload register. TMRx_CC1 preload value is sent to the active register at each update event.</p> <p><i>Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPMODE = '1' in the TMRx_CTRL1 register). Else, the behavior is not guaranteed.</i></p>
Bit 2	<p>OC1FEN: Output compare 1 fast enable This bit is used to accelerate the CC output's response to trigger the input event. 0: CC1 behaves normally with the counter and CC1 values even when the trigger is ON. When an active edge occurs on the trigger input, the minimum delay to activate CC1 output is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match. Therefore, OC is set to the compare level which is irrelevant to the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. This bit only acts when channel is configured as PWM1 or PWM2 mode.</p>
Bit 1:0	<p>C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: CC1 channel is configured as input, IC1 is mapped on TI2. 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i></p>

Input capture mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3:0]		IC2DIV[1:0]		C2SEL[1:0]		IC1DF[3:0]		IC1DIV[1:0]		C1SEL[1:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:12		IC2DF[3:0]: Input capture 2 filter													
Bit 11:10		IC2DIV[1:0]: Input capture 2 prescaler													

Bit 9:8	C2SEL[1:0]: Capture/Compare 2 selection The bits define the channel direction (input/output), and input pin selection: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C2SEL can be written only when the channel is disabled (C2EN = '0' in the TMRx_CCE register).</i>
Bit 7:4	IC1DF[3:0]: Input capture 1 filter The bits define the frequency used to sample TI1 input and the length of digital filter. The digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$ 0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$ 1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$ 0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$ 1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$ 0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$ 1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$ 0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$ 1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$ 0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$ 1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$ 0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$ 1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$ 0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$ 1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$
Bit 3:2	IC1DIV[1:0]: Input capture 1 prescaler The bits define CC1 input (IC1) prescaler ratio. Once C1EN = '0' (in the TMRx_CCE register), prescaler is reset. 00: No prescaler. Capture is done each time when an edge is detected on the capture input. 01: Capture is done once every 2 events. 10: Capture is done once every 4 events 11: Capture is done once every 8 events.
Bit 1:0	C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: CC1 channel is configured as input, IC1 is mapped on TI2. 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i>

10.2.4.8 Capture/Compare Mode Register 2 (TMRx_CCM2)

Address offset: 0x1C

Reset value: 0x0000 Please refer to the above-mentioned CCM1 register description.

Output compare mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 DIS	OC4MODE[2:0]	OC4 PEN	OC4 FEN	C4SEL[1:0]	OC3 DIS	OC3MODE[2:0]	OC3 PEN	OC3 FEN	C3SEL[1:0]						
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

Bit 15	OC4DIS: Output compare 4 clear enable
Bit 14:12	OC4MODE[2:0]: Output compare 4 mode
Bit 11	OC4PEN: Output compare 4 preload enable
Bit 10	OC4FEN: Output compare 4 fast enable

Bit 9:8	C4SEL[1:0]: Capture/Compare 4 selection The bits define the channel direction (input/output), and input pin selection: 00: CC4 channel is configured as output. 01: CC4 channel is configured as input, IC4 is mapped on TI4. 10: CC4 channel is configured as input, IC4 is mapped on TI3. 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C4SEL can be written only when the channel is disabled (C4EN = '0' in the TMRx_CCE register).</i>
Bit 7	OC3DIS: Output compare 3 clear enable
Bit 6:4	OC3MODE[2:0]: Output compare 3 mode
Bit 3	OC3PEN: Output compare 3 preload enable
Bit 2	OC3FEN: Output compare 3 fast enable
Bit 1:0	C3SEL[1:0]: Capture/Compare 3 selection The bits define the channel direction (input/output), and input pin selection: 00: CC3 channel is configured as output. 01: CC3 channel is configured as input, IC3 is mapped on TI3. 10: CC3 channel is configured as input, IC3 is mapped on TI4. 11: CC3 channel is configured as input, IC3 is mapped on TRGI. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C3SEL can be written only when the channel is disabled (C3EN = '0' in the TMRx_CCE register).</i>

Input capture mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4DF[3:0]				IC4DIV[1:0]		C4SEL[1:0]		IC3DF[3:0]				IC3DIV[1:0]		C3SEL[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:12	IC4DF[3:0]: Input capture 4 filter
Bit 11:10	IC4DIV[1:0]: Input capture 4 prescaler
Bit 9:8	C4SEL[1:0]: Capture/compare 4 selection The bits define the channel direction (input/output), and input pin selection: 00: CC4 channel is configured as output. 01: CC4 channel is configured as input, IC4 is mapped on TI4. 10: CC4 channel is configured as input, IC4 is mapped on TI3. 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C4SEL can be written only when the channel is disabled (C4EN = '0' in the TMRx_CCE register).</i>
Bit 7:4	IC3DF[3:0]: Input capture 3 filter
Bit 3:2	IC3DIV[1:0]: Input capture 3 prescaler
Bit 1:0	C3SEL[1:0]: Capture/Compare 3 selection The bits define the channel direction (input/output), and input pin selection: 00: CC3 channel is configured as output. 01: CC3 channel is configured as input, IC3 is mapped on TI3. 10: CC3 channel is configured as input, IC3 is mapped on TI4. 11: CC3 channel is configured as input, IC3 is mapped on TRGI. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C3SEL can be written only when the channel is disabled (C3EN = '0' in the TMRx_CCE register).</i>

10.2.4.9 Capture/Compare Enable Register (TMRx_CCE)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	C4P	C4EN	C3NP	Reserved	C3P	C3EN	C2NP	Reserved	C2P	C2EN	C1NP	Reserved	C1P	C1EN	
res	rw	rw	res	rw	rw	rw	res	rw	rw	rw	res	rw	rw	rw	rw

Bit 15:14	Reserved. Always read as 0.
Bit 13	C4P: Capture/Compare 4 output polarity Please refer to C1P description.
Bit 12	C4EN: Capture/Compare 4 output enable Please refer to C1EN description.
Bit 11	C3NP: Capture/Compare 3 complementary output polarity Please refer to C1NP description.
Bit 10	Reserved. Always read as 0.
Bit 9	C3P: Capture/Compare 3 output polarity Please refer to C1P description.
Bit 8	C3EN: Capture/Compare 3 output enable Please refer to C1EN description.
Bit 7	C2NP: Capture/Compare 2 complementary output polarity Please refer to C1NP description.
Bit 6	Reserved. Always read as 0.
Bit 5	C2P: Capture/Compare 2 output polarity Please refer to C1P description.
Bit 4	C2EN: Capture/Compare 2 output enable Please refer to C1EN description.
Bit 3	C1NP: Capture/Compare 1 complementary output polarity CC1 channel is set as output: C1NP must be kept in clear state. CC1 channel is set as input: The polarity of TI1FP1/TI2FP1 is defined by C1NP according to C1P. (Refer to C1P description)
Bit 2	Reserved. Always read as 0.
Bit 1	C1P: Capture/Compare 1 output polarity CC1 channel is configured as output: 0: OC1 is active high. 1: OC1 is active low. CC1 channel is configured as input: This bit selects whether IC1 or inverted IC1 is used for trigger or capture signal. 0: Not inverted: Capture is done on a rising edge of IC1. When used as external trigger, IC1 is not inverted. 1: Inverted: Capture is done on a falling edge of IC1. When used as external trigger, IC1 is inverted.
Bit 0	C1EN: Capture/Compare 1 output enable CC1 channel is configured as output: 0: OFF— OC1 output is disabled. 1: ON— OC1 is outputted on the corresponding output pin. CC1 channel is configured as input: This bit determines whether the counter value can be captured to TMRx_CC1 register. 0: Capture is disabled. 1: Capture is enabled.

Table 10-5 Standard OCx Channel Output Control Bit

CxEN Bit	OCx Output State
0	Output disabled ($OCx = 0$, $OCx_EN = 0$)
1	$OCx = OCxREF + Polarity$, $OCx_EN = 1$

Note: The state of the external I/O pins connected to the standard OCx channels is determined by the OCx channel state and the GPIO and AFIO registers.

10.2.4.10 Counter (TMRx_CNT)

Address offset: 0x24

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:16		CNT[31:16]: Counter value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), CNT is expanded to 32 bits.													
Bit 15:0		CNT[15:0]: Counter value													

10.2.4.11 Prescaler (TMRx_DIV)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
Bit 15:0		DIV[15:0]: Prescaler value The counter clock frequency CK_CNT is fCK_DIV/(DIV[15:0]+1). DIV contains the value loaded in the active prescaler register at each update event													

10.2.4.12 Auto-reload Register (TMRx_AR)

Address offset: 0x2C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
Bit 31:16		AR[31:16]: Auto-reload value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), AR is expanded to 32 bits.													

Bit 15:0	AR[15:0]: Auto-reload value AR contains the value to be loaded to the actual auto-reload register. When auto-reload value is null, the counter does not work. Please refer to Section 10.2.3.1 for more details on AR update and behavior.
----------	---

10.2.4.13 Capture/Compare Register 1 (TMRx_CC1)

Address offset: 0x34

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
CC1[31:16]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CC1[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:16		CC1[31:16]: Capture/Compare 1 value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), CC1 is expanded to 32 bits.														
Bit 15:0		CC1[15:0]: Capture/Compare 1 value If CC1 channel is configured as output: CC1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM1 register (the OC1PEN bit). Else, the preload value is loaded in the active capture/compare 1 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates the output on OC1. If CC1 channel is configured as input: CC1 is the counter value transferred by the last input capture 1 event (IC1).														

10.2.4.14 Capture/Compare Register 2 (TMRx_CC2)

Address offset: 0x38

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
CC2[31:16]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CC2[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:16		CC2[31:16]: Capture/Compare 2 value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), CC2 is expanded to 32 bits.														
Bit 15:0		CC2[15:0]: Capture/Compare 2 value If CC2 channel is configured as output: CC2 is the value to be loaded in the actual capture/compare 2 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM2 register (the OC2PEN bit). Else the preload value is loaded in the active capture/compare 2 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates output on OC2. If CC2 channel is configured as input: CC2 is the counter value transferred by the last input capture 2 event (IC2).														

10.2.4.15 Capture/Compare Register 3 (TMRx_CC3)

Address offset: 0x3C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:16	CC3[31:16]: Capture/Compare 3 value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), CC3 is expanded to 32 bits.														
	CC3[15:0]: Capture/Compare 3 value If CC3 channel is configured as output: CC3 is the value to be loaded in the actual capture/compare 3 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM3 register (the OC3PEN bit). Else the preload value is loaded in the active capture/compare 3 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates output on OC3. If CC3 channel is configured as input: CC3 is the counter value transferred by the last input capture 3 event (IC3).														

10.2.4.16 Capture/Compare Register 4 (TMRx_CC4)

Address offset: 0x40

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CC4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:16	CC4[31:16]: Capture/Compare 4 value When TMR2 or TMR5 enables plus mode (The PMEN bit in the TMR_CTRL1 register), CC4 is expanded to 32 bits.														
	CC4[15:0]: Capture/Compare 4 value If CC4 channel is configured as output: CC4 is the value to be loaded in the actual capture/compare 4 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM4 register (the OC4PEN bit). Else the preload value is loaded in the active capture/compare 4 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates output on OC4. If CC4 channel is configured as input: CC4 is the counter value transferred by the last input capture 4 event (IC4).														

10.2.4.17 DMA Control Register (TMRx_DMAR)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
res	rw	rw	rw	rw	rw	rw	rw	res	rw						
Bit 15:13	Reserved. Always read as 0.														
Bit 12:8	DBLEN[4:0]: DMA burst length The bits define the number of DMA transfers in burst mode (the timer recognizes a burst transfer when a read or a write access is done to the TMRx_DMABA address). Namely, defining the byte number of transfer. 00000: 1 byte 00001: 2 bytes 00010: 3 bytes 10001: 18 bytes														
Bit 7:5	Reserved. Always read as 0.														
Bit 4:0	ADDR[4:0]: DMA base address The bits define the base address for DMA transfers in burst mode (when read/write access is done to the TMRx_DMABA register). ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register. 00000: TMRx_CTRL1, 00001: TMRx_CTRL2, 00010: TMRx_SMC,														

10.2.4.18 DMA Address in Burst Mode (TMRx_DMABA)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0	DMABA[15:0]: DMA register for burst accesses A read or write to the TMRx_DMABA register accesses the register located at the following address: TMRx_CTRL1 address + ADDR + DMA index, where: TMRx_CTRL1 address is the address of the control register 1(TMRx_CTRL1). "ADDR" is the base address defined in the TMRx_DMAR register. "DMA index" is the offset automatically controlled by the DMA, and is determined by the DBLEN bit configured in the TMRx_DMAR register.														

10.3 General-purpose Timer (TMR9 to TMR14)

10.3.1 TMRx Introduction

The general-purpose timers consist of a 16-bit auto-reload counter driven by a programmable prescaler. They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

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

The timers are completely independent, and they do not share any resources and can be synchronized. Please refer to [Section 10.3.3.12](#).

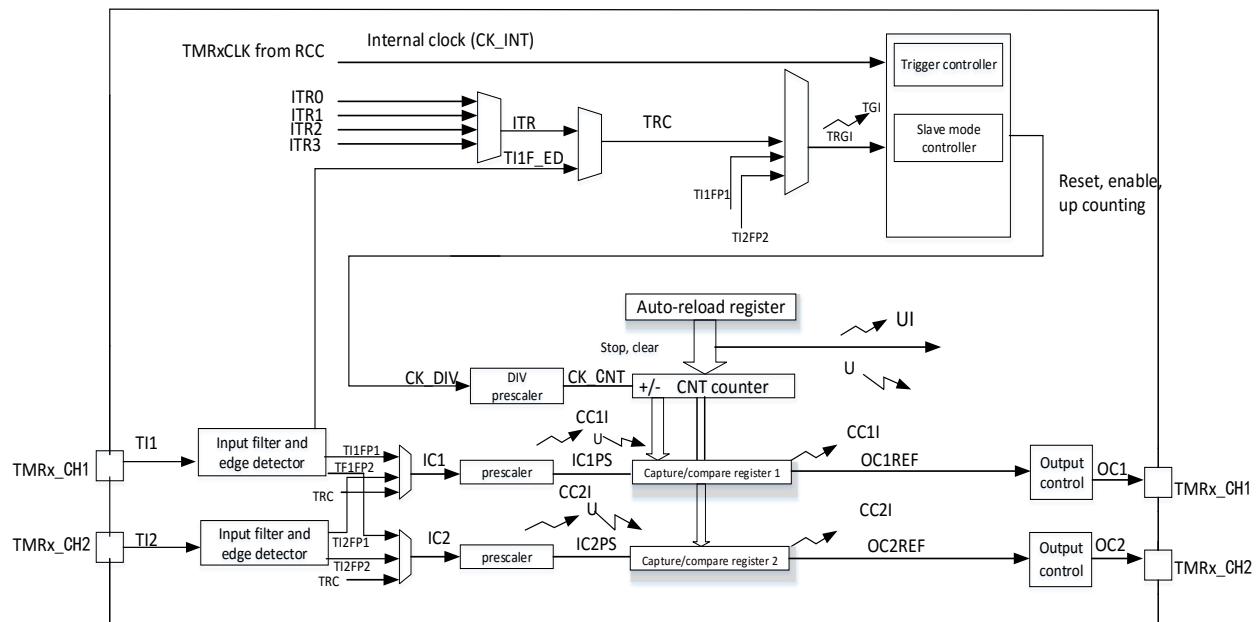
10.3.2 TMRx Main Function

10.3.2.1 TMR9 and TMR12 Main Function

The main functions of general-purpose TMRx (TMR9 and TMR12) include:

- 16-bit up auto-reload counter
- 16-bit programmable prescaler (can be modified on the fly) used to divide the counter clock frequency by any factor between 1 ~ 65536
- 2 independent channels:
 - Input capture
 - Output compare
 - PWM generation (Edge-aligned mode)
 - One-pulse mode output
- Timers and the interconnected synchronization circuit are controlled by external signals.
- Interrupt generation at the following events:
 - Update: Counter overflow, counter initialization (by software or internal trigger)
 - Trigger event (counter starts, stops, initialized, or counts by internal trigger)
 - Input capture
 - Output compare

Figure 10-57 Block Diagram of General-purpose Timer TMR9/12



Note: transfers the contents in the preload registers to active registers on U event according to the control bits.

Event

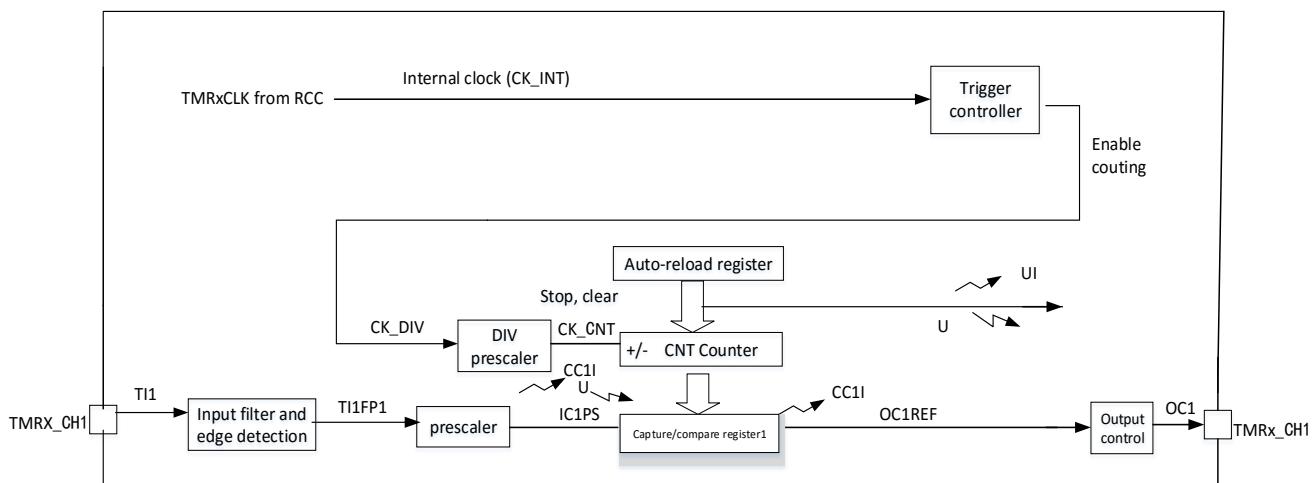
Interrupt

10.3.2.2 TMR10, TMR11, TMR13, and TMR14 Main Function

The main functions of general-purpose TMRx (TMR10, TMR11, TMR13, and TMR14) include:

- 16-bit up auto-reload counter
- 16-bit programmable prescaler (can be modified on the fly) used to divide the counter clock frequency by any factor between 1 ~ 65536
- 1 independent channel:
 - Input capture
 - Output compare
 - PWM generation (Edge-aligned mode)
 - One-pulse mode output
- Interrupt generation at the following events:
 - Update: Counter overflow, counter initialization (by software or internal trigger)
 - Input capture
 - Output compare

Figure 10-58 Block Diagram of General-purpose Timers TMR10/11/13/14



Note: transfers the contents in the preload registers to active registers on U event according to the control bits.

- Event
- ↗ Interrupt output

10.3.3 TMRx Function Overview

10.3.3.1 Time-base Unit

This programmable general-purpose timer mainly consists of a 16-bit counter and relevant auto-reload registers. The counter can count up. The counter clock is obtained through a prescaler.

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

The time-base unit includes:

- Counter register (TMRx_CNT)
- Prescaler register (TMRx_DIV)
- Auto-reload register (TMRx_AR)

The auto-reload register is preloaded. Each read or write to the auto-reload register will access the preload register. The contents written to the preload register are transferred into its shadow register immediately or at each update event (UEV) according to the auto-reload preload enable bit (ARPEN) in the TMRx_CTRL1 register. When the UEVDIS bit in the TMRx_CTRL1 register is '0', update events will be generated once the counter reaches the overflow value. Update events can also be generated by software. Please refer to the following sections for the detailed introduction on generation of the update events under different configurations.

The counter is driven by the prescaler clock output, CK_CNT, and it is enabled when the counter enable bit (CNTEN) in the TMRx_CTRL1 register is set. (Please refer to the controller slave mode descriptions for more details on counter enable.)

Note: The actual counter enable signal, CNT_EN, is set 1 clock cycle after CNTEN is set..

Prescaler

The prescaler can divide the counter clock frequency by any factor between 1 ~ 65536, and this is achieved with a 16-bit counter controlled by a 16-bit register (in the TMRx_DIV register). Its value can be changed on the fly since the control register is buffered. The new prescaler ratio takes effect at the next update event.

[Figure 10-59](#) and [Figure 10-60](#) are examples of on-the-fly prescaler ratio change.

Figure 10-59 Counter Timing Diagram with Prescaler Division Changing from 1 to 2

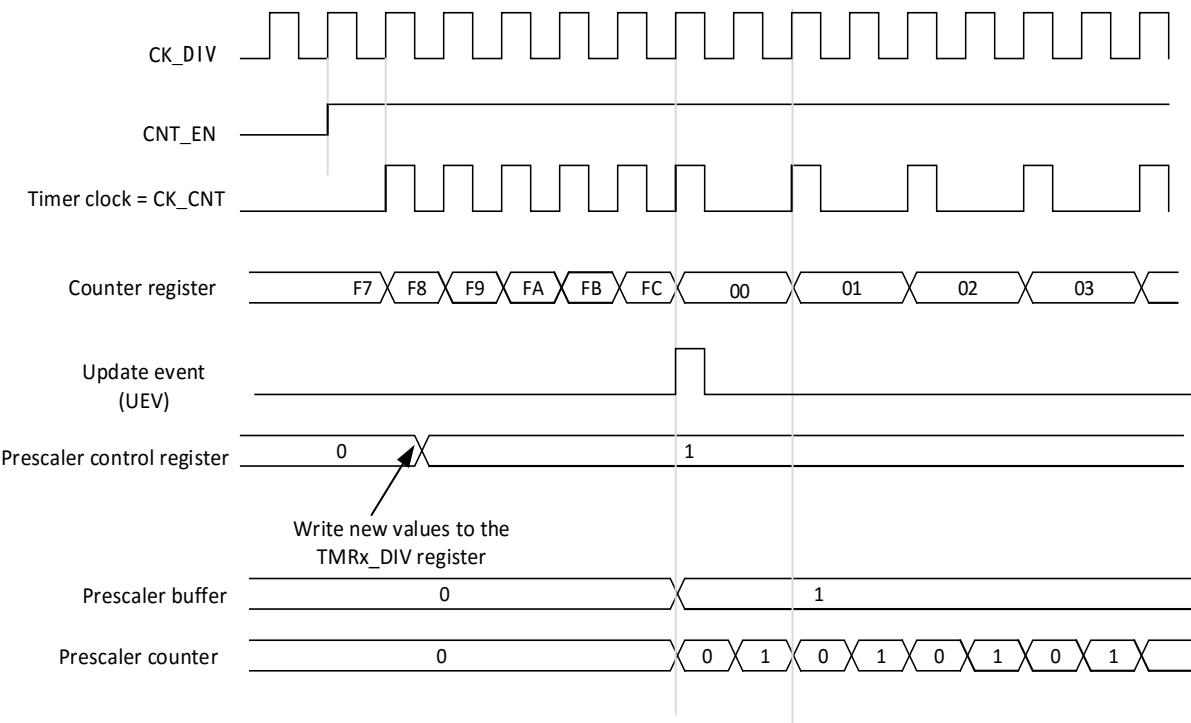
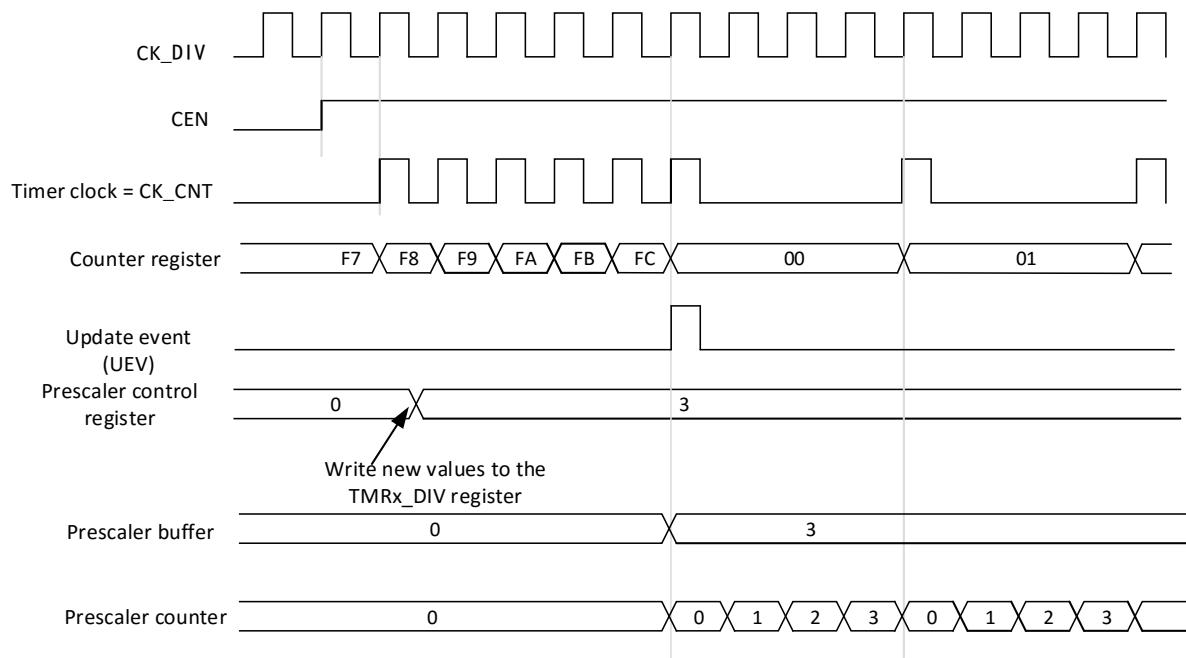


Figure 10-60 Counter Timing Diagram with Prescaler Division Changing from 1 to 4



10.3.3.2 Counter Mode

Upcounting mode

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

An update event can be generated at each counter overflow; setting the UEVG bit in the TMRx_EVEG register (by software, for TMR9 and 12, even by the slave mode controller) also generates the update events.

Update events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing new values to the preload registers. No update event occurs until the UEVDIS bit is cleared. However, when the update events are generated, the counter and the

prescaler are still cleared (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set by hardware (that is, no interrupt is sent). This can avoid update and capture interrupt generation when the counter is cleared in capture mode.

When an update event occurs, all the registers are updated, and the update flag (the UEVIF bit in the TMRx_STS register) is set by hardware (according to the UEVRS bit).

- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).
- The auto-reload shadow register is updated with the preload value (TMRx_AR).

Figure 10-61 ~ Figure 10-66 show the examples of the counter behaviors for different clock frequencies when TMRx_AR = 0x36:

Figure 10-61 Counter Timing Diagram with Internal Clock Divided by 1

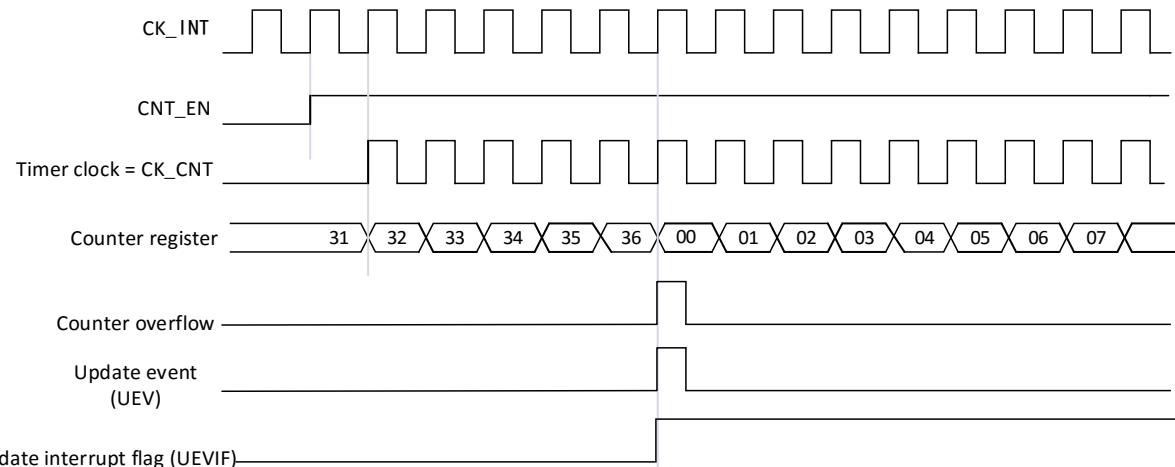


Figure 10-62 Counter Timing Diagram with Internal Clock Divided by 2

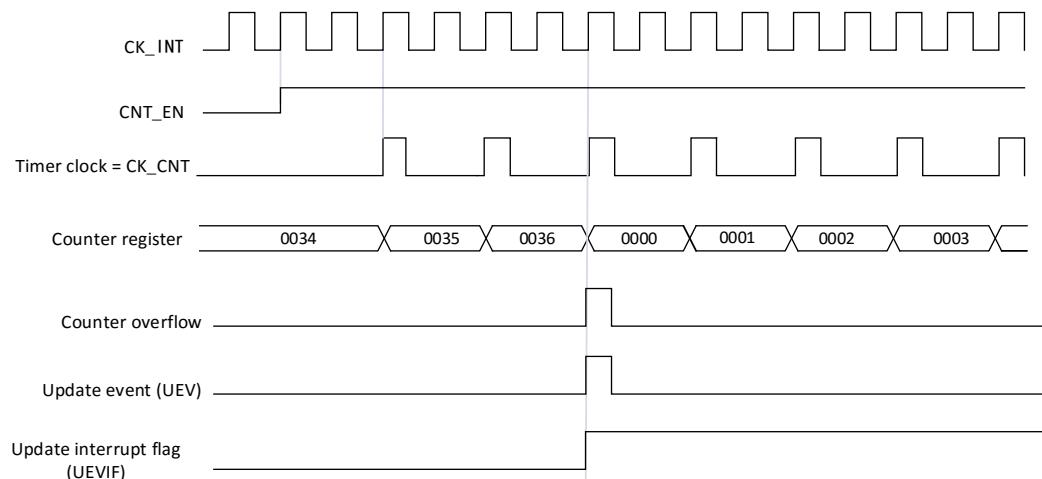


Figure 10-63 Counter Timing Diagram with Internal Clock Divided by 4

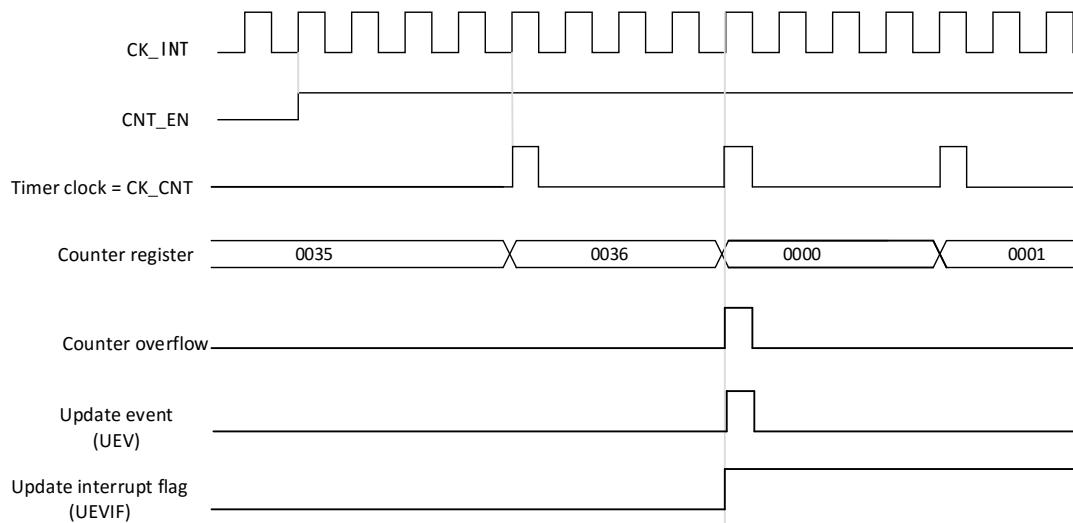


Figure 10-64 Counter Timing Diagram with Internal Clock Divided by N

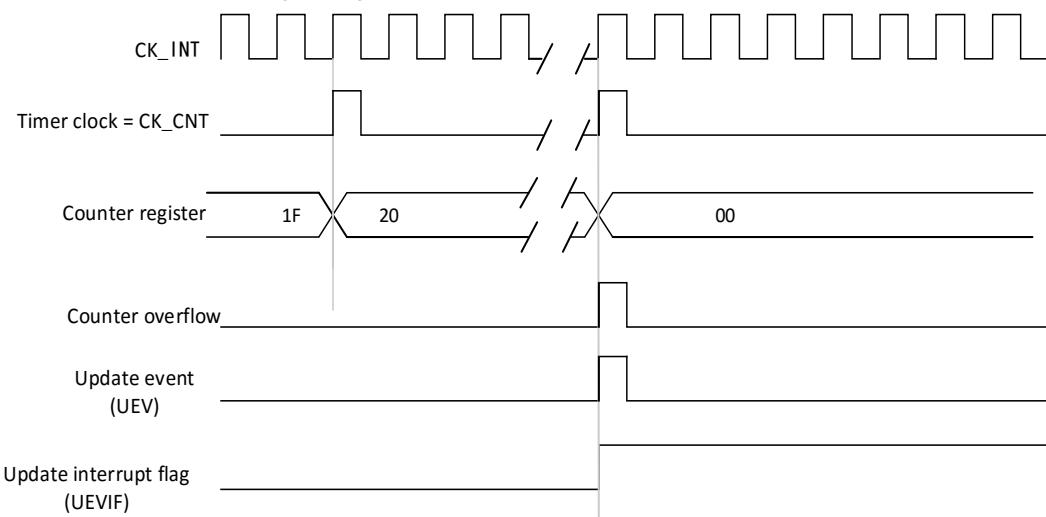


Figure 10-65 Counter Timing Diagram with Update Event When ARPEN = 0 (TMRx_AR Not Preloaded)

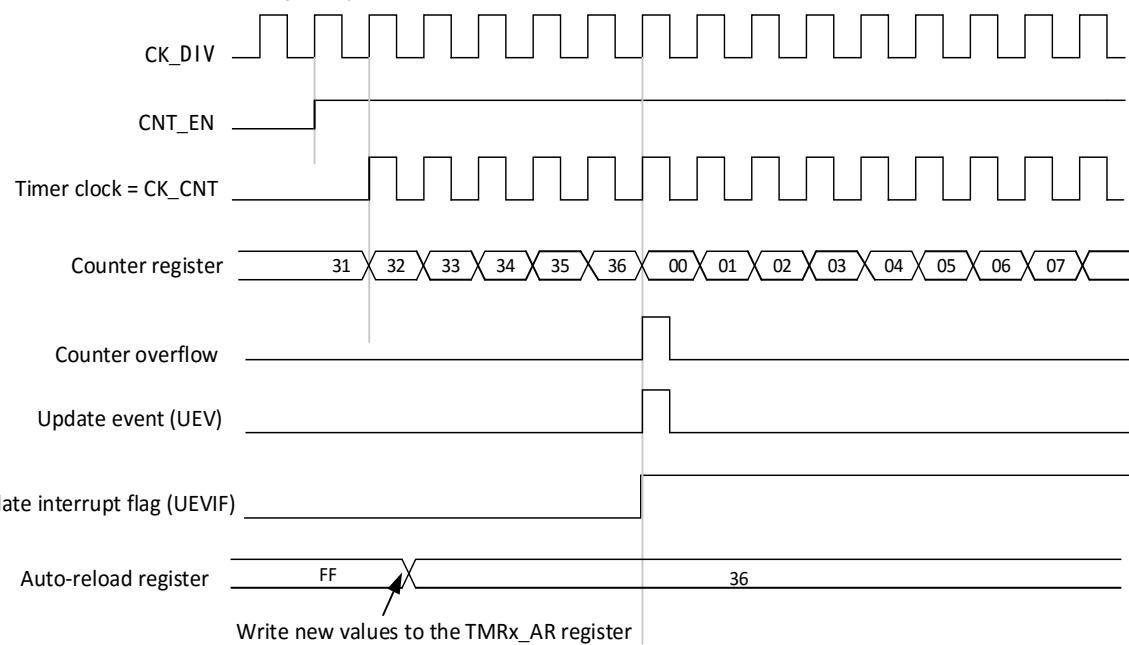
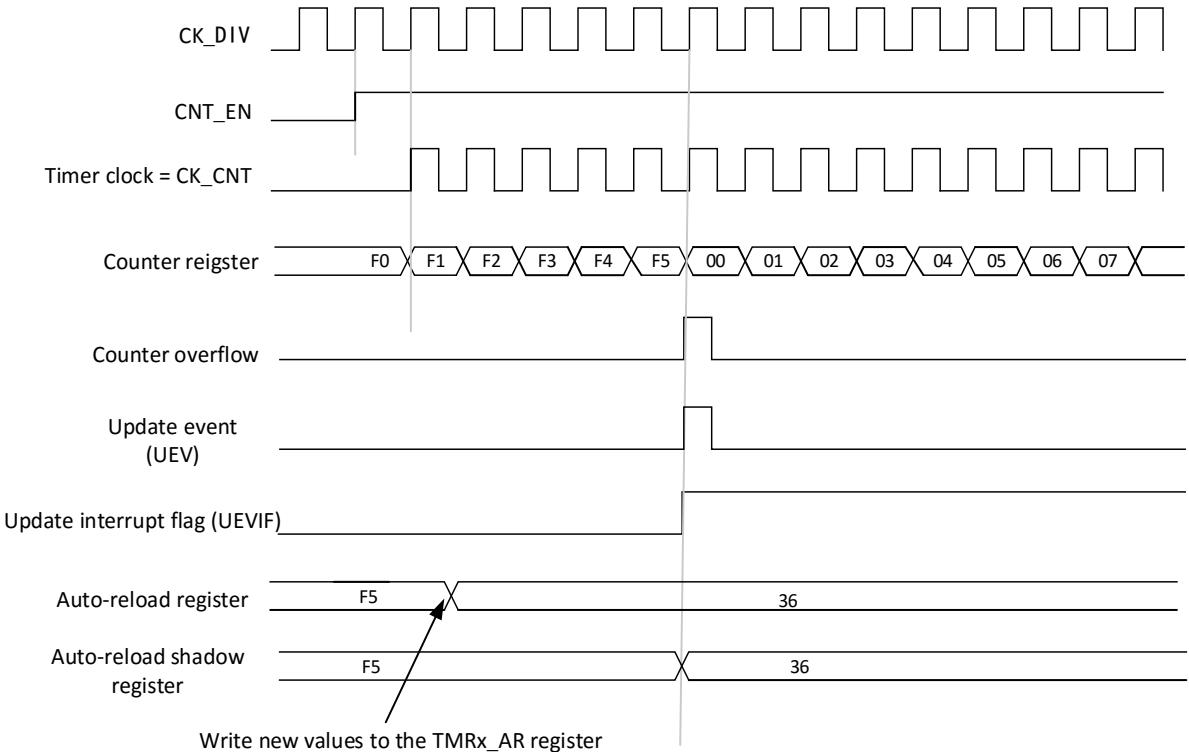


Figure 10-66 Counter Timing Diagram with Update Event When ARPEN = 1 (TMRx_AR is Preloaded)



10.3.3.3 Clock Selection

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

- Internal clock (CK_INT)
- External clock mode 1 (only TMR9 and TMR12): External input pin (TIx)
- Internal trigger input (ITRx) (only TMR9 and TMR12): The trigger signal connected to another timer. For example, users can configure Timer 1 to act as the prescaler for Timer 2. Please refer to [Section 10.3.3.12](#).

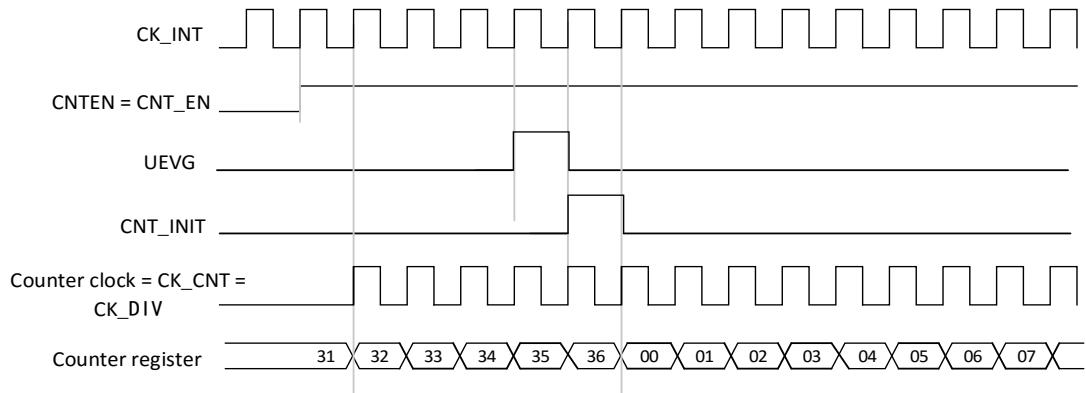
Internal clock source (CK_INT)

For TMR10/TMR11 and TMR13/TMR14, internal clock source is the clock source by default.

For TMR9 and TMR12, if the slave mode controller is disabled (SMSEL = 000 in the TMRx_SMC register), the CNTEN, DIR (in the TMRx_CTRL1 register), and UEVG bits (in the TMRx_EVEG register) are the actual control bits, and they can be changed only by software (except for the UEVG bit, which will be cleared automatically). As soon as the CNTEN bit is written '1', the prescaler is clocked by the internal clock, CK_INT.

Figure 10-67 shows the behavior of the control circuit and the up counter in normal mode, without prescaler.

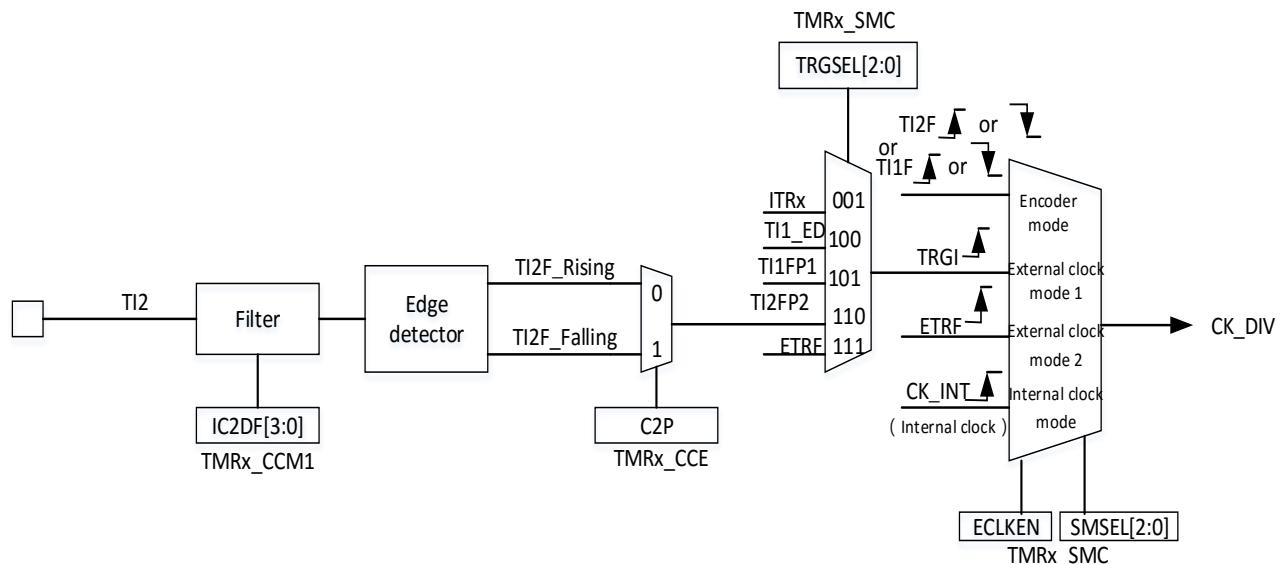
Figure 10-67 Control Circuit in Normal Mode with Internal Clock Divided by 1



External clock mode 1 (TMR9 and TMR12)

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

Figure 10-68 TI2 External Clock Connection Example



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

1. Configure channel 2 to detect the rising edges on the TI2 input by writing C2SEL = '01' in the TMRx_CCM1 register.
2. Configure the input filter duration by writing the IC2DF[3:0] bits in the TMRx_CCM1 register (if no filter is needed, keep IC2F = 0000).

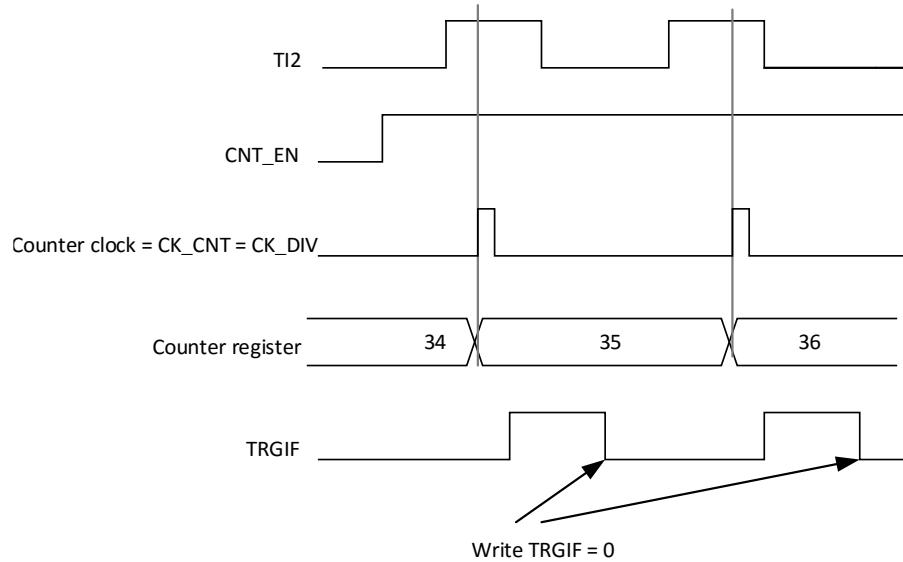
Note: The capture prescaler is not used as trigger, so it does not need to be configured.

3. Select the rising edge polarity by writing C2P = '0' in the TMRx_CCE register.
4. Select the timer external clock mode 1 by writing SMSEL = '111' in the TMRx_SMC register.
5. Select TI2 as the trigger input source by writing TRGSEL = '110' in the TMRx_SMC register.
6. Enable the counter by writing CNTEN = '1' in the TMRx_CTRL1 register.

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

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

Figure 10-69 Control Circuit in External Clock Mode 1

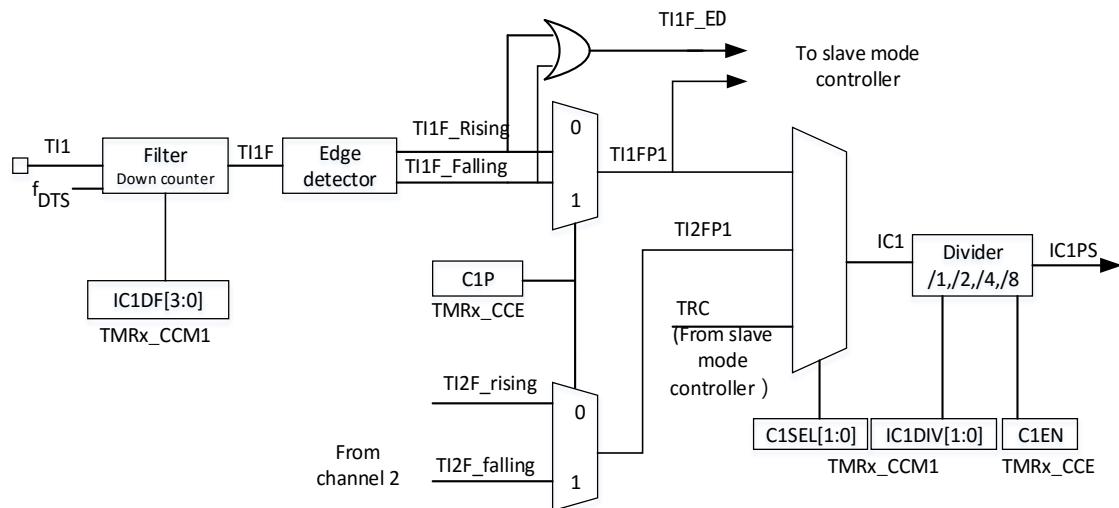


10.3.3.4 Capture/Compare Channel

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

Figure 10-70 ~ Figure 10-72 provide an overview of one capture/compare channel. The input stage samples the corresponding TIx input signal to generate a filtered signal TIxF. Then, an edge detector with polarity selection generates a signal (TIxFPx) which can be the trigger input for the slave mode controller, or be the capture command. This signal enters the capture register through the prescaler.

Figure 10-70 Capture/Compare Channel (e.g. Channel 1 Input Stage)



The output stage generates an intermediate waveform OCxRef (active high) which serves as reference. The polarity of final output signal is determined by the end of the chain.

Figure 10-71 Capture/Compare Channel 1 Main Circuit

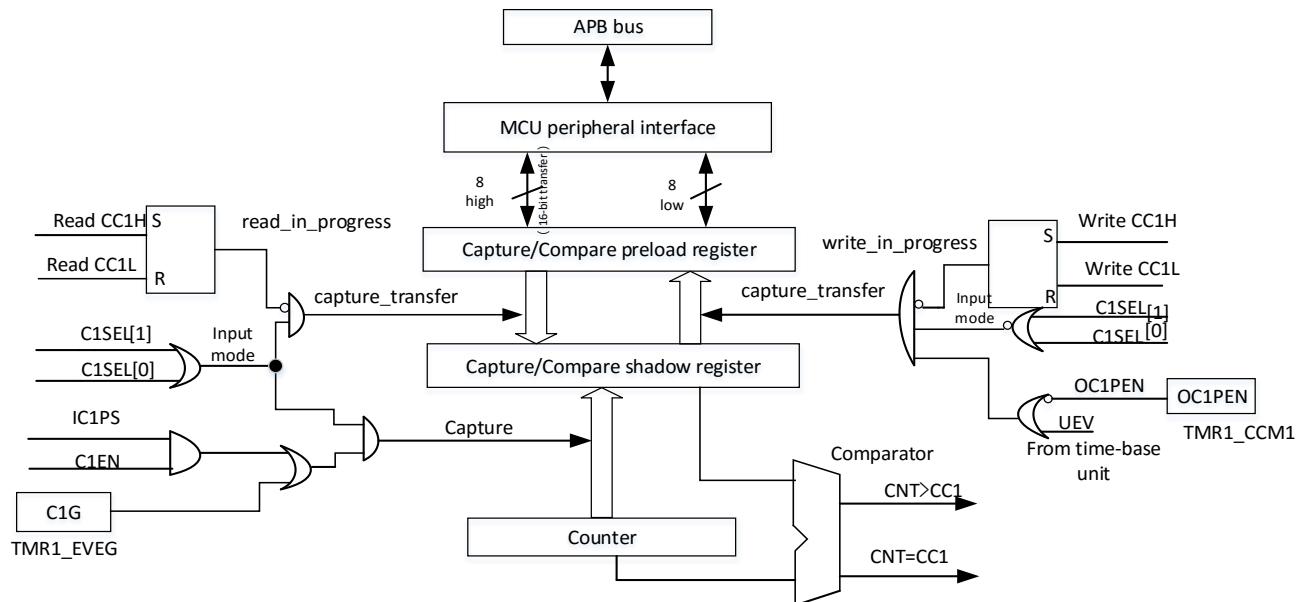
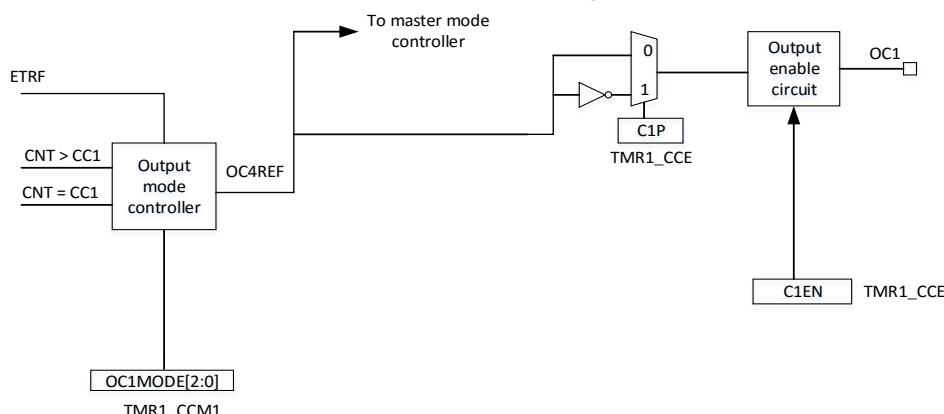


Figure 10-72 Capture/Compare Channel Output Stage (Channel 1)



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

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

In compare mode, the content of the preload register is copied into the shadow register, and then the content of the shadow register is compared with the counter.

10.3.3.5 Input Capture Mode

In input capture mode, the capture/compare registers (TMRx_CCx) latch the current counter value after a corresponding edge on the ICx signal is detected. When a capture occurs, the corresponding CxIF flag (TMRx_STS register) is set. An interrupt or a DMA request can be sent if enabled. If a capture occurs when the CxIF flag is already high, the over-capture flag CxOF (TMRx_STS register) is set. CxIF can be cleared by writing CxIF = 0 or by reading the captured data stored in the TMRx_CCx register. CxOF = 0 is cleared when writing CxOF = 0.

The following example shows how to capture the counter value to the TMRx_CC1 register at TI1 input rising edge:

- Select the active input: Write C1SEL = 01 in the TMRx_CC1 register since the TMRx_CC1 register must be linked to the TI1 input. The channel is configured as input as long as the C1SEL is not '00'. The TM1_CC1 register becomes read-only.
- Program the desired input filter duration according to the input signal (if the input

is TIx, the control bit of the input filter is the ICxDF bit in the TMRx_CCMx register). If the input signal is not stable during five internal clock cycles at most, the filter duration should be configured longer than five clock cycles. Therefore, an actual edged transition on TI1 can be validated by 8 consecutive samples (at f_{DTS} frequency). That is, write IC1DF = 0011 in the TMRx_CCM1 register.

- Select the active transition edge on the TI1 channel by writing C1P = 0 (rising edge) in the TMRx_CCE register.
- Program the input prescaler. In this example, each valid transition is expected to be captured, so the prescaler is disabled (write IC1DIV = 00 in the TMRx_CCM1 register).
- Set C1EN = 1 in the TMRx_CCE register to allow the counter value to be captured into the capture register
- If needed, enable the relevant interrupt request by setting the C1IE bit in the TMRx_DIE register.

When an input capture occurs:

- The counter value is sent to the TMRx_CC1 register on active transition.
- C1IF flag is set (interrupt flag). When at least two consecutive captures occur and the C1IF flag is not cleared, C1OF is also set.
- An interrupt is generated if the C1IE bit is set.
- In case of overcapture, it is recommended that the data is read before the overcapture flag. This is to avoid missing an overcapture information that is generated after reading the overcapture flag and before reading the data.

Note: Input capture interrupt can be generated by setting the corresponding CxG bit in the TMRx_EVEG register with software.

10.3.3.6 PWM Input Mode (Only TMR9 and TMR12)

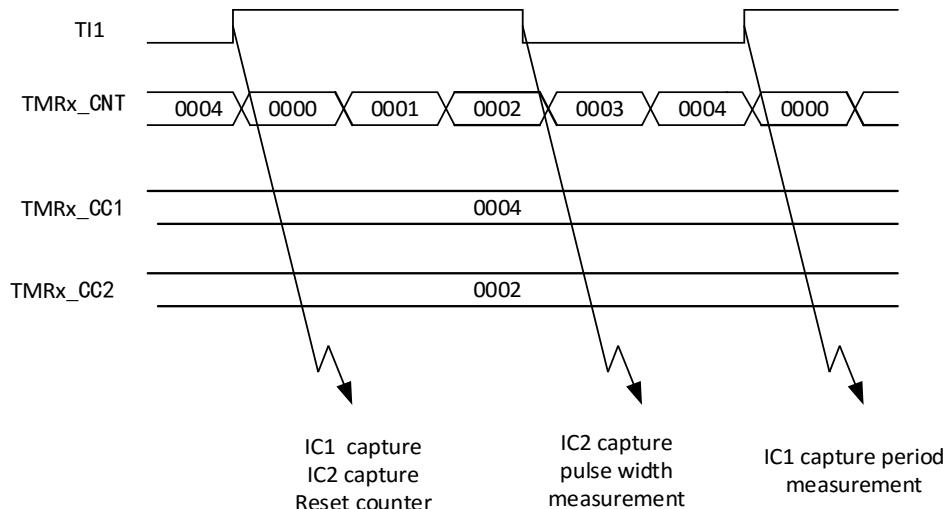
This mode is a special case of input capture mode. The procedure is the same as input capture mode except for the following conditions:

- Two ICx signals are mapped on the same TIx input.
- The two ICx signals are active on edges but with opposite polarities.
- One of the two TIxFP signals is used as trigger input signal and the slave mode controller is configured as reset mode.

For example, if users need to measure the period (TMRx_CC1 register) and the duty cycle (TMRx_CC2 register) of the PWM signal input on TI1, the procedure is as follows (depending on CK_INT frequency and prescaler value):

- Select the active input for TMRx_CC1: Write C1SEL = 01 in the TMRx_CCM1, register (TI1 selected).
- Select the active polarity for TI1FP1 (used to capture data into TMRx_CC1 and clear the counter): Write C1P = 0 (active on rising edge).
- Select the active input for TMRx_CC2: Write C2SEL = 10 in the TMRx_CCM1 register (TI1 selected).
- Select the active polarity for TI1FP2 (capture data to TMRx_CC2): Write C2P = 1 (active on falling edge).
- Select the valid trigger input signal: Write TRGSEL = 101 in the TMRx_SMC register (TI1FP1 selected).
- Configure the slave mode controller as reset mode: Write SMSEL = 100 in the TMRx_SMC register.
- Enable the captures: Write C1EN = 1 and C2EN = 1 in the TMRx_CCE register.

Figure 10-73 PWM Input Mode Timing



Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can be used only with the TMRx_CH1/TMRx_CH2 signals.

10.3.3.7 Forced Output Mode

In output mode ($CxSEL = 00$ in the TMRx_CCMx register), output compare signal (OCxREF and the corresponding OCx) can be forced to be active or inactive level directly by software, independent of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, users can write the corresponding OCxMODE = 101 in the TMRx_CCMx register. In this way, OCxREF is forced high (OCxREF is always active high), and at the same time OCx gets the opposite value of the CxP polarity bit.

For example: CxP = 0 (OCx active high), then OCx is forced to be high level.

The OCxREF signal can be forced low by writing OCxMODE = 100 in the TMRx_CCx register.

In this mode, the comparison between the TMRx_CCx shadow register and the counter is still ongoing, and the corresponding flag is also modified. Accordingly, the corresponding interrupt and DMA requests are still generated. This will be described in the “Output Compare Mode” section.

10.3.3.8 Output Compare Mode

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

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

- Output the value defined by the output compare mode (the OCxMODE bit in the TMRx_CCMx register) and the output polarity (the CxP bit in the TMRx_CCE register) to the corresponding pin. When being compared and matched, the output pin can keep its level (OCxMODE = 000), be set active (OCxMODE = 001), be set inactive (OCxMODE = 010), or toggle (OCxMODE = 011).
- Set the flag bit in the interrupt status register (the CxIF bit in the TMRx_STS register).
- An interrupt will be generated if the corresponding interrupt mask is set (the CxIE bit in the TMRx_DIE register).

The OCxPEN bit in the TMRx_CCMx register can be configured to choose whether the TMRx_CCx register uses the preload register.

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

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

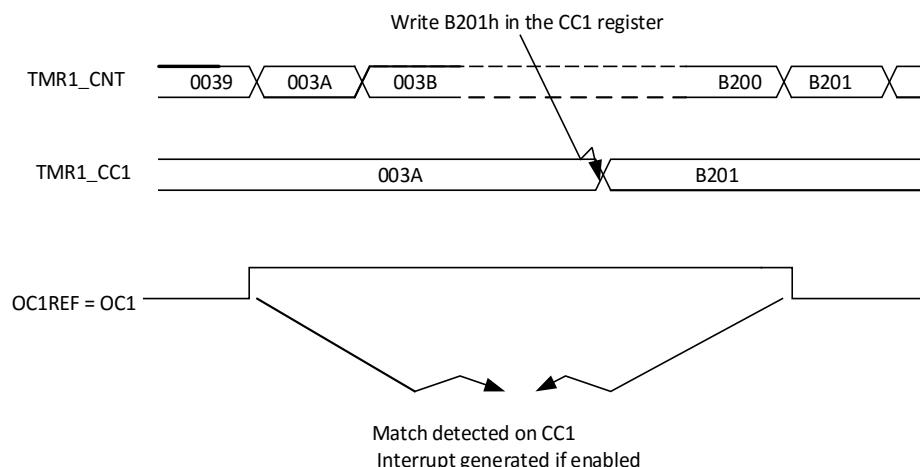
Output compare mode configuration procedure is as follows:

1. Select the counter clock (internal, external, and prescaler).

2. Write the corresponding data to the TMRx_AR and the TMRx_CCx registers.
3. Set the CxIE and/or CxDE bits if an interrupt and/or DMA request is to be generated.
4. Select the output mode. For example, when the counter CNT and the CCx bit are matched, toggle the output pin of OCx, CCx preload is not disabled, and the OCx output is enabled with active high, users must configure OCxMODE = '011', OCxPEN = '0', CxP = '0', and CxEN = '1'.
5. Enable the counter by setting the CNTEN bit in the TMRx_CTRL1 register.

The TMRx_CCx register can be updated at any time by software to control the output waveform, under the condition that the preload register is not enabled (OCxPEN = '0', else the TMRx_CCx shadow register can only be updated at the next update event). An example is shown in Figure 10-74.

Figure 10-74 Output Compare Mode, Toggle on OC1



10.3.3.9 PWM Mode

Pulse width modulation mode can generate a signal with its frequency determined by the TMRx_AR register and its duty cycle determined by TMRx_CCx register.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxMODE bit in the TMRx_CCMx register can independently configure each OCx output channel and generate single-channel PWM. The OCxPEN bit in the TMRx_CCMx register must be configured to enable the corresponding preload register. Finally, the ARPEN bit in the TMRx_CTRL1 register (in upcounting or center-aligned modes) must be set to enable the auto-reload preload register.

Since the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, users must initialize all the registers by setting the UEVG bit in the TMRx_EVEG register. OCx polarity is software programmable by setting the CxP bit in the TMRx_CCE register. It can be programmed as active high or active low. OCx output enable is controlled by the CxEN bit in the TMRx_CCE register. Please refer to the description of TMRx_CCE register for more details.

In PWM mode (1 or 2), TMRx_CNT and TMRx_CCx are always compared to confirm that $TMRx_CCx \leq TMRx_CNT$ or $TMRx_CNT \leq TMRx_CCx$.

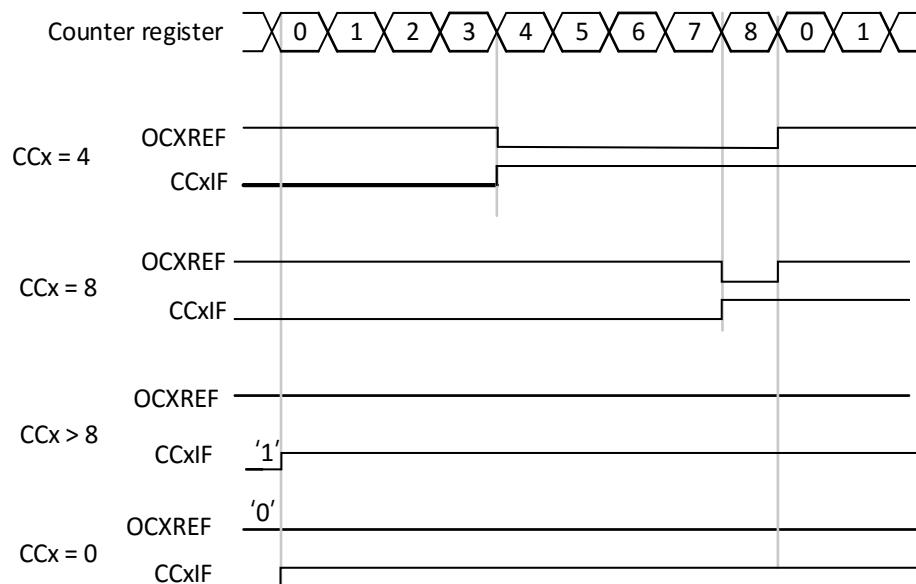
In edge-aligned mode, timers generate PWM only when the counter counts up.

PWM edge-aligned mode

The following is an example of PWM mode 1. The reference PWM signal, OCxREF, is high once $TMRx_CNT < TMRx_CCx$; else, it becomes low. If the compare value in TMRx_CCx is greater than the auto-reload value (in TMRx_AR), OCxREF is held at '1'.

If the compare value is 0, then OCxREF is held at '0'. Figure 10-75 shows an example of edge-aligned PWM waveforms with TMRx_AR = 8.

Figure 10-75 Edge-aligned PWM Waveforms (AR = 8)



10.3.3.10 One-pulse Mode

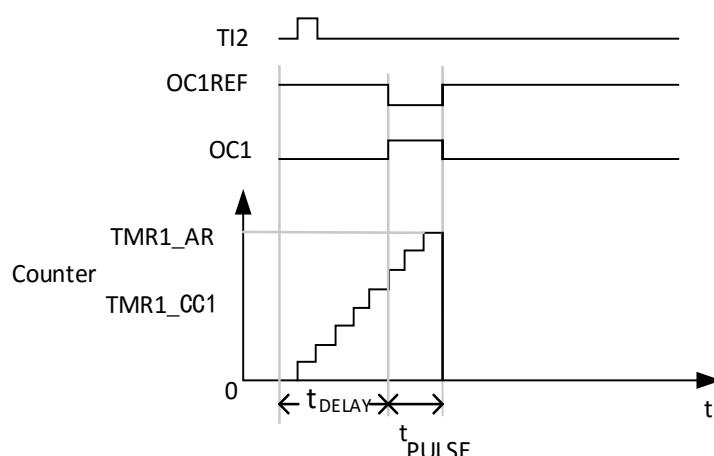
One-pulse mode (OPM) is quite different from the above-mentioned modes. It allows the counter to respond to a stimulus and to generate a pulse with a programmable length after a programmable delay.

The counter can be enabled through the slave mode controller, generating waveforms in output compare mode or PWM mode. One-pulse mode is selected by setting the OPMODE bit in the TMRx_CTRL1 register, and this can stop the counter automatically at the next update event UEV.

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

$$\text{CNT} < \text{CCx} \leq \text{AR} \text{ (especially, } 0 < \text{CCx})$$

Figure 10-76 Example of One-pulse Mode



For example, if users want to generate a positive pulse with a length of t_{PULSE} on OC1 once a rising edge is detected on the TI2 input pin and after a delay t_{DELAY} .

If TI2FP2 is used as trigger 1:

- Map TI2FP2 to TI2 by writing C2SEL = '01' in the TMRx_CCM1 register.
- Write C2P = '0' in the TMRx_CCE register to enable TI2FP2 to detect a rising edge.
- Write TRGSEL = '110' in the TMRx_SMC register to make TI2FP2 to be the trigger of the slave mode controller (TRGI).
- Write SMSEL = '110' in the TMRx_SMC register (trigger mode), and TI2FP2 is used to enable the counter.

The OPM waveform is defined by the value written to the compare registers (the clock frequency and

the counter prescaler should be taken into account).

- t_{DELAY} is defined by the value written to the TMRx_CC1 register.
- t_{PULSE} is defined by the comparison result of auto-reload value and the compare value (TMRx_AR - TMRx_CC1).
- Assume that users want to generate a waveform with a transition from '0' to '1' when a compare match occurs, and a transition from '1' to '0' when the counter reaches the preload value. First, set OC1MODE = '111' in the TMRx_CCM1 register to enter PWM mode 2. Enable the desired preload registers by setting OC1PEN = '1' in the TMRx_CCM1 register and the ARPEN bit in the TMRx_CTRL1 register. Afterwards, write compare value in the TMRx_CC1 register, write auto-reload value in the TMRx_AR register, modify the UEVG bit to generate an update event, and wait for an external trigger event on TI2. In this example, C1P = '0'.

In this example, since only one pulse is needed, '1' must be written to the OPMODE bit in the TMRx_CTRL1 register to stop the counter at the next update event (when the counter rolls over to 0 from the auto-reload value).

When OPMODE = '0', exit one-pulse mode.

Special case: OCx fast enable

In one-pulse mode, the counter is enabled by setting the CNTEN bit on the edge detection logic of TIx input pin. Then, the comparison between the counter and the compare value makes output toggle. However, since several clock cycles are needed for these operations, it limits the minimum delay t_{DELAY} that users can get.

The OCxFEN bit in the TMRx_CCMx register can be set to output a waveform with the minimum delay. In this case, OCxREF (and OCx) are forced to respond to the stimulus, instead of relying on the comparison result. The waveform outputted is the same as the one being compared and matched. OCxFEN acts only if the channel is configured as PWM1 or PWM2 mode.

10.3.3.11 Timer and External Trigger Synchronization (TMR9 and TMR12 Only)

The TMRx timer can be synchronized with an external trigger in several modes: reset mode, gated mode, and trigger mode.

Slave mode: Reset mode

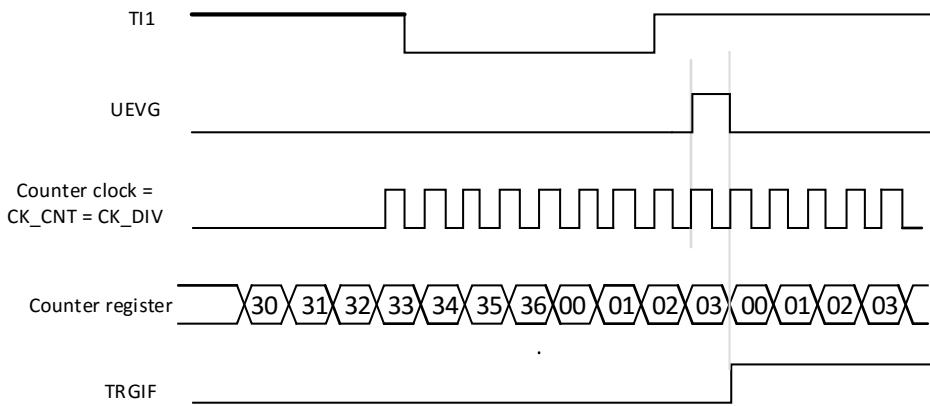
The counter and its prescaler can be reinitialized when a trigger input event occurs. In this case, if the UEVRS bit of the TMRx_CTRL1 register is low, an update event UEV is generated. Then, all the preloaded registers (TMRx_AR, TMRx_CCx) will be updated.

In the following example, the up counter is cleared due to a rising edge on TI1 input:

- Configure channel 1 to detect the rising edges on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bit selects the input capture source only, that is, C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 0 in the TMRx_CCE register to confirm the polarity (and detect rising edges only).
- Configure the timer as reset mode by writing SMSEL = 100 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Start the counter by writing CNTEN = 1 in the TMRx_CTRL1 register.

The counter starts counting on the internal clock, and runs normally until TI1 rising edge occurs; at this time, the counter is cleared and restarts from 0. In the meantime, the trigger flag (the TRGIF bit in the TMRx_STS register) is set, and an interrupt request can be sent depending on the configuration of the TRGIE bit in the TMRx_DIE register (interrupt enable). Figure 10-77 shows the behavior when the auto-reload register TMRx_AR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is determined by the resynchronization circuit on TI1 input.

Figure 10-77 Control Circuit in Reset Mode



Slave mode: Gated mode

The counter can be enabled according to the selected level of input.

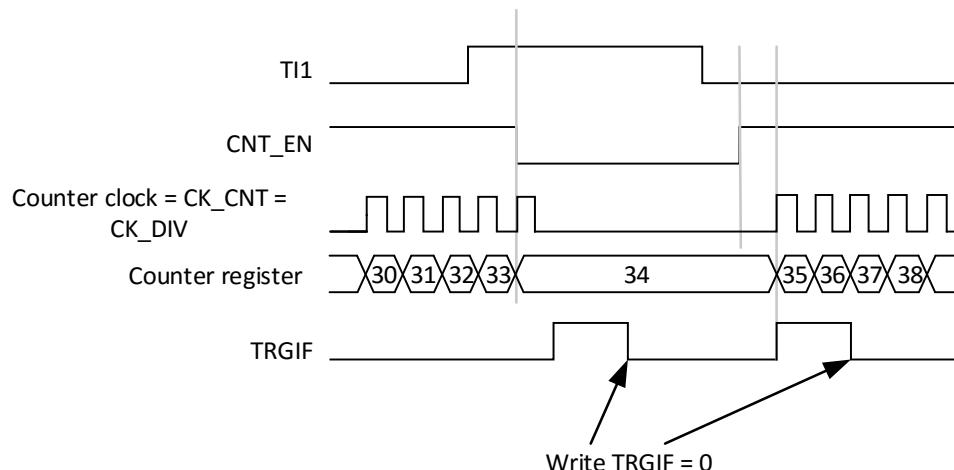
In the following example, the counter counts up only when TI1 input is low:

- Configure channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bits select the input capture source. Set C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 1 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as gated mode by writing SMSEL = 101 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Enable the counter by writing CNTEN = 1 in the TMRx_CTRL1 register. In gated mode, the counter cannot be enabled if CNTEN = 0, no matter what the trigger input level is.

The counter starts counting on the internal clock as long as TI1 is low, and stops as soon as TI1 becomes high. The TRGIF flag in the TMRx_STS register is set both when the counter starts and stops.

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

Figure 10-78 Control Circuit in Gated Mode



Slave mode: Trigger mode

The counter can be enabled according to the selected level of input.

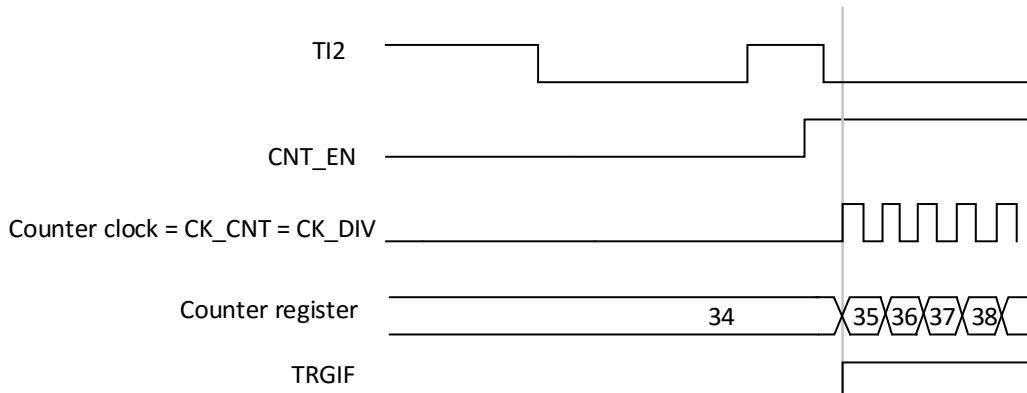
In the following example, the counter starts to count up at rising edge on TI2 input:

- Configure channel 2 to detect the rising edge on TI2. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C2SEL bits are only used to select the input capture source. Set C2SEL = 01 in the TMRx_CCM1 register. Write C2P = 1 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as trigger mode by writing SMSEL = 110 in the TMRx_SMC register. Select TI2 as the input source by writing TRGSEL = 110 in the TMRx_SMC register.

The counter starts counting on the internal clock as long as a rising edge occurs on TI2, and at the same time the TRGIF flag is set.

The delay between the rising edge on TI2 and enabling counting is determined by the resynchronization circuit on TI2 input.

Figure 10-79 Control Circuit in Trigger Mode



10.3.3.12 Timer Synchronization (TMR9 and TMR12 Only)

All TMRx timers are linked together internally for timer synchronization or chaining. Please refer to [Section 10.2.3.15](#) for more details.

Note: Before receiving the master timer trigger signal, slave timer clock should be enabled first. When receiving trigger from master timer, do not change it while running.

10.3.3.13 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter either continues to work normally or stops, depending on DBG_TMRx_STOP configuration in DBG module. For more details, please refer to [Section 22.2](#).

10.3.4 TMR9 and TMR12 Register Description

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

In Table 10-6, all the TMRx registers are mapped to a 16-bit addressable space.

Table 10-6 TMRx – Register Table and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	TMRx_CTRL1	Reserved																				CLKDIV[1:0]		ARPEN		Reserved		OPMODE		UEVRS		UEVDIS		CNTEN	
	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

0x08	TMRx_SMC	Reserved												
	0x0000													
0x0C	TMRx_DIE	Reserved												
	0x0000													
0x10	TMRx_STS	Reserved												
	0x0000													
0x14	TMRx_EVEG	Reserved												
	0x0000													
0x18	TMRx_CCM1 output compare mode	Reserved				OC2MODE[2:0]		OC2PEN		OC2FEN		C2SEL[1:0]		
	0x0000					0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	
	TMRx_CCM1 input capture mode	Reserved				IC2DF[3:0]		IC2DIV[1:0]		C2SEL[1:0]		OC1MODE[2:0]		
	0x0000					0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	0 0 0 0	
0x1C		Reserved												
0x20	TMRx_CCE	Reserved												
	0x0000													
0x24	TMRx_CNT	Reserved	CNT[15:0]											
	0x0000		0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
0x28	TMRx_DIV	Reserved	DIV[15:0]											
	0x0000		0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
0x2C	TMRx_AR	Reserved	AR[15:0]											
	0x0000		0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
0x34	TMRx_CC1	Reserved	CC1[15:0]											
	0x0000		0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0
0x38	TMRx_CC2	Reserved	CC2[15:0]											
	0x0000		0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0

10.3.4.1 Control Register 1 (TMRx_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					CLKDIV [1:0]	ARPE N		Reserved		OPM ODE	UEVR S	UEVD IS		CNTE N	
RES				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit 15:10	Reserved. Always read as 0.														
Bit 9:8	CLKDIV[1:0]: Clock division Define the division ratio between the timer clock (CK_INT) frequency and sampling frequency used by the digital filters (ETR, TIx). 00: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved														
Bit 7	ARPEN: Auto-reload preload enable 0: TMRx_AR register is not buffered. 1: TMRx_AR register is buffered.														
Bit 6:4	Reserved. Always read as 0.														
Bit 3	OPMODE: One pulse mode 0: The counter does not stop at update event. 1: The counter stops at the next update event (the CNTE bit is cleared).														
Bit 2	UEVRS: Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt if update interrupt request is enabled: - Counter overflow - Setting the UEVG bit 1: Only counter overflow generates an update interrupt if update interrupt is enabled.														
Bit 1	UEVDIS: Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV is enabled. An update event (UEV) is generated by any of the following events: - Counter overflow - Setting the UEVG bit 1: UEV is disabled. Update events are not generated, and shadow registers (AR, DIV, and CCx) keep their values. The counter and the prescaler are reinitialized if the UEVG bit is set or the slave mode controller sends a hardware reset.														
Bit 0	CNTEN: Counter enable 0: The counter is disabled. 1: The counter is enabled. CNTEN is cleared automatically in one-pulse mode when an update event occurs.														

10.3.4.2 Slave Mode Control Register (TMRx_SMC)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRGSEL[2:0]				Res erve d	SMSEL[2:0]		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RES	RW	RW	RW
Bit 15:7		Reserved. Always read as 0.													

Bit 6:4	TRGSEL[2:0]: Trigger selection The bits select the trigger input used to synchronize the counter. 000: Internal trigger 0 (ITR0) 100: TI1 edge detector (TI1F_ED) 001: Internal trigger 1 (ITR1) 101: Filtered timer input 1 (TI1FP1) 010: Internal trigger 2 (ITR2) 110: Filtered timer input 2 (TI2FP2) 011: Internal trigger 3 (ITR3) 111: Reserved Please refer to Table 10-7 for the details on ITRx for each timer. <i>Note: These bits must be changed only when they are not used (e.g. when SMSEL = 000) to avoid wrong edge detections at transitions.</i>
Bit 3	Reserved. Always read as 0.
Bit 2:0	SMSEL[2:0]: Slave mode selection When external signals are selected, the active edge of the trigger signal (TRGI) is relevant to the selected external input polarity (Please refer to input control register and control register description.) 000: Slave mode is disabled - If CNTEN = 1, the prescaler is clocked directly by the internal clock. 001: Reserved 010: Reserved 011: Reserved 100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register signal. 101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) once the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger input TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger input (TRGI) clocks the counter. <i>Note 1: The gated mode must not be used if TI1F_EN is selected as the trigger input (TRGSEL = 100). This is because TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger input signal.</i> <i>Note 2: Before receiving the master timer trigger signal, slave timer clock should be enabled first. While receiving trigger signal from master timer, do not change slave timer clock.</i>

Table 10-7 TMRx Internal Trigger Connection ^(Note)

Slave Timer	ITR0 (TRGSEL = 000)	ITR1 (TRGSEL = 001)	ITR2 (TRGSEL = 010)	ITR3 (TRGSEL = 011)
TMR9	TMR2_TRGO	TMR3_TRGO	TMR10_OC	TMR11_OC
TMR12	TMR4_TRGO	TMR5_TRGO	TMR13_OC	TMR14_OC

Note: When a corresponding timer is not present in the product, the corresponding trigger ITRx is not available.

10.3.4.3 DMA/Interrupt Enable Register (TMRx_DIE)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								TRG IE	Reserved		C2I E	C1I E	UEV IE		
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit 15:7		Reserved. Always read as 0.													
Bit 6		TRGIE: Trigger interrupt enable 0: Trigger interrupt is disabled. 1: Trigger interrupt is enabled.													
Bit 5:3		Reserved. Always read as 0.													

Bit 2	C2IE: Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt is disabled. 1: Capture/Compare 2 interrupt is enabled.
Bit 1	C1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt is disabled. 1: Capture/Compare 1 interrupt is enabled.
Bit 0	UEVIE: Update interrupt enable 0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.3.4.4 Status Register (TMRx_STS)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				Reserved	C2O F	C1O F	Reserved	TRG IF		Reserved	C2I F	C1I F	UEV IF		
RES	RW	RW	RW	RW	RES	RW	RES	RW	RW	RW	RW	RW	RW	RW	RW

Bit 15:11	Reserved. Always read as 0.
Bit 10	C2OF: Capture/Compare 2 overcapture flag Please refer to C1OF description.
Bit 9	C1OF: Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured as input capture mode. It is cleared by writing '0'. 0: No overcapture is detected. 1: When the counter value is captured to the TMRx_CC1 register, C1IF flag is already set.
Bit 8:7	Reserved. Always read as 0.
Bit 6	TRGIF: Trigger interrupt flag This flag is set by hardware at trigger event (When the slave mode controller is in all modes except for gated mode, active edge is detected on TRGI input, or any edge in the gated mode). It is cleared by software. 0: No trigger event occurs. 1: Trigger interrupt is pending.
Bit 5:3	Reserved. Always read as 0.
Bit 2	C2IF: Capture/Compare 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF: Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This flag is set by hardware when the counter value matches the compare value, but not in center-aligned mode (Please refer to the CMSEL bits in the TMRx_CTRL1 register). It is cleared by software. When the TMRx_CC1 value is higher than TMRx_AR, the CC1F bit is set high after counter overflow. 0: No match 1: The TMRx_CNT value matches the TMRx_CC1 value. If channel CC1 is configured as input mode: This bit is set by hardware on a capture. It is cleared by software or by reading the TMRx_CC1 register. 0: No input capture occurs. 1: The counter value is captured to the TMRx_CC1 register (An edge which matches the selected polarity is detected on IC1).

Bit 0	<p>UEVIF: Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update event occurs. 1: Update interrupt is pending. This bit is set by hardware when the registers are updated: – If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when UEVG = 1 in the TMRx_EVEG register (the counter CNT is reinitialized by software). – If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when the counter CNT is reinitialized by trigger events. This bit is set at counter overflow when UEVDIS = 0</p>
-------	---

10.3.4.5 Event Generation Register (TMRx_EVEG)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		TRG G		Reserved		C2G		C1G		UEV G					
RES		RW		RES		RW		RW		RW		RW		RW	

Bit 15:7	Reserved. Always read as 0.
Bit 6	<p>TRGG: Trigger generation This bit is set by software to generate a trigger event. It is cleared automatically by hardware. 0: No action 1: TRGIF = 1 in the TMRx_STS register, the corresponding interrupt is generated if enabled.</p>
Bit 5:3	Reserved. Always read as 0.
Bit 2	<p>C2G: Capture/compare 2 generation Please refer to C1G description.</p>
Bit 1	<p>C1G: Capture/compare 1 generation This bit is set by software to generate a capture/compare event. It is cleared automatically by hardware. 0: No action 1: A capture/compare event is generated on channel CC1: If channel CC1 is configured as output: Set C1IF = 1. The corresponding interrupt request is generated if enabled. If channel CC1 is configured as input: The current value of the counter is captured to the TMRx_CC1 register. Set C1IF = 1. The corresponding interrupt request is generated if enabled. If C1IF is already '1', set C1OF = 1.</p>
Bit 0	<p>UEVG: Update generation This bit can be set by software, and it is automatically cleared by hardware. 0: No action 1: Re-initialize the counter and generate an update. Please note that the prescaler counter is also cleared, but the prescaler ratio is not affected. The counter is cleared.</p>

10.3.4.6 Capture/Compare Mode Register 1 (TMRx_CCM1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by the corresponding CxSEL bits. All the other bits of this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is in output mode, ICxx describes its function when the channel is in input mode. Attention must be given to the fact that the same bit can have different definitions for the input stage and for the output stage.

Output compare mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	OC2MODE[2:0]	OC2 PEN	OC2 FEN	C2SEL[1:0]	Reserve d	OC1MODE[2:0]	OC1 PEN	OC1 FEN	C1SEL[1:0]						
RES	RW	RW	RW	RW	RW	RW	RW	RES	RW	RW	RW	RW	RW	RW	RW

Bit 15	OC2DIS: Reserved.
Bit 14:12	OC2MODE[2:0]: Output compare 2 mode
Bit 11	OC2PEN: Output compare 2 preload enable
Bit 10	OC2FEN: Output compare 2 fast enable
Bit 9:8	<p>C2SEL[1:0]: Capture/Compare 2 selection This bit defines the channel direction (input/output) and the selection of input pin: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C2SEL can be written only when the channel is disabled (C2EN = '0' in the TMRx_CCE register).</i></p>
Bit 7	Reserved.
Bit 6:4	<p>OC1MODE[2:0]: Output compare 1 enable The bits define the behavior of the output reference signal, OC1REF, and OC1REF determines the OC1 value. OC1REF is active high, while the active level of OC1 is determined by the C1P bit.</p> <p>000: Frozen. The comparison between the output compare register TMRx_CC1 and the counter TMRx_CNT has no effect on OC1REF. 001: Set channel 1 to active level on match. OC1REF is forced high when the value of the counter TMRx_CNT matches the capture/compare register 1 (TMRx_CC1). 010: Set channel 1 to inactive level on match. OC1REF is forced low when the value of the counter TMRx_CNT matches the capture/compare register 1 (TMRx_CC1). 011: Toggle. Toggle OC1REF level when TMRx_CC1 = TMRx_CNT. 100: Force inactive level. OC1REF is forced low. 101: Force active level. OC1REF is forced high. 110: PWM mode1—Channel 1 is active once TMRx_CNT < TMRx_CC1, else inactive. 111: PWM mode2—Channel 1 is inactive once TMRx_CNT < TMRx_CC1, else active.</p> <p><i>Note: In PWM mode 1 or PWM mode 2, OC1REF level changes only when the comparison result changes or when output compare mode switches from frozen mode to PWM mode .</i></p>
Bit 3	<p>OC1PEN: Output compare 1 preload enable 0: Preload function of TMRx_CC1 is disabled. TMRx_CC1 can be written at any time, and the new value takes effect immediately. 1: Preload function of TMRx_CC1 is enabled. Read/Write operations only access the preload register. TMRx_CC1 preload value is sent to the active register at each update event.</p> <p><i>Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPMODE = '1' in the TMRx_CTRL1 register). Else, the behavior is not guaranteed.</i></p>
Bit 2	<p>OC1FEN: Output compare 1 fast enable This bit is used to accelerate the CC output's response to trigger the input event. 0: CC1 behaves normally with the counter and CC1 values even when the trigger is ON. When an active edge occurs on the trigger input, the minimum delay to activate CC1 output is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match. Therefore, OC is set to the compare level which is irrelevant to the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. This bit only acts when channel is configured as PWM1or PWM2 mode.</p>

Bit 1:0	C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: CC1 channel is configured as input, IC1 is mapped on TI2. 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i>
---------	--

Input capture mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3:0]		IC2DIV[1:0]		C2SEL[1:0]		IC1DF[3:0]		IC1DIV[1:0]		C1SEL[1:0]					
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW

Bit 15:12	IC2DF[3:0]: Input capture 2 filter
Bit 11:10	IC2DIV[1:0]: Input capture 2 prescaler
Bit 9:8	C2SEL[1:0]: Capture/compare 2 selection The bits define the channel direction (input/output), and input pin selection: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C2SEL can be written only when the channel is disabled (C2EN = '0' in the TMRx_CCE register).</i>
Bit 7:4	IC1DF[3:0]: Input capture 1 filter The bits define the frequency used to sample TI1 input and the length of digital filter. The digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$ 0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$ 1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$ 0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$ 1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$ 0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$ 1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$ 0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$ 1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$ 0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$ 1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$ 0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$ 1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$ 0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$ 1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$
Bit 3:2	IC1DIV[1:0]: Input capture 1 prescaler The bits define CC1 input (IC1) prescaler ratio. Once C1EN = '0' (in the TMRx_CCE register), the prescaler is reset. 00: No prescaler. Capture is done each time when an edge is detected on the capture input. 01: Capture is done once every 2 events. 10: Capture is done once every 4 events 11: Capture is done once every 8 events.
Bit 1:0	C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: CC1 channel is configured as input, IC1 is mapped on TI2. 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i>

10.3.4.7 Capture/Compare Enable Register (TMRx_CCE)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								C2NP	Reserve d	C2P	C2EN	C1NP	Reserve d	C1P	C1E N
							RES	RW	RES	RW	RW	RW	RES	RW	RW
Bit 15:8															
Bit 7															
Bit 6															
Bit 5															
Bit 4															
Bit 3															
Bit 2															
Bit 1															
Bit 0															

Table 10-8 Standard OCx Channel Output Control Bit

CxEN Bit	OCx Output State
0	Output disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The state of the external I/O pins connected to the standard OCx channels is determined by the OCx channel state and the GPIO and AFIO registers.

10.3.4.8 Counter (TMRx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

10.3.4.9 Prescaler (TMRx_DIV)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit 15:0	DIV[15:0]: Prescaler value The counter clock frequency CK_CNT is fCK_DIV/(DIV[15:0]+1). DIV contains the value loaded in the active prescaler register at each update event.														

10.3.4.10 Auto-reload Register (TMRx_AR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit 15:0	AR[15:0]: Auto-reload value AR contains the value to be loaded to the actual auto-reload register. Please refer to Section 10.3.3.1 for more details on AR update and behavior. When auto-reload value is null, the counter does not work.														

10.3.4.11 Capture/Compare Register 1 (TMRx_CC1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15:0]															
RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
Bit 15:0	CC1[15:0]: Capture/Compare 1 value If CC1 channel is configured as output: CC1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM1 register (the OC1PEN bit). Else, the preload value is loaded in the active capture/compare 1 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates the output on OC1. If CC1 channel is configured as input: CC1 is the counter value transferred by the last input capture 1 event (IC1).														

10.3.4.12 Capture/Compare Register 2 (TMRx_CC2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC2[15:0]															

RW RW

Bit 15:0	CC2[15:0]: Capture/Compare 2 value If CC2 channel is configured as output: CC2 is the value to be loaded in the actual capture/compare 2 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM2 register (the OC2PEN bit). Else the preload value is loaded in the active capture/compare 2 register when an update event occurs. If CC2 channel is configured as input: CC2 is the counter value transferred by the last input capture 2 event (IC2).

10.3.5 TMR10, TMR11, TMR13, and TMR14 Registers Description

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

In Table 10-9, all the TMRx registers are mapped to a 16-bit addressable space.

Table 10-9 TMRx –Register Table and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0x00	TMRx_CTRL1	Reserved																CLKDIV[1:0]		ARLEN		Reserved						UEVRS		UEVDIS		CNTEN					
	Reset Value																	0		0								0		0		0					
0x0C	TMRx_DIE	Reserved																C1IE		C1EVIE								0		0		0					
	Reset Value																																				
0x10	TMRx_STS	Reserved																C1OF		Reserved								C1IF		UEVIF		0					
	Reset Value																	0																			
0x14	TMRx_EVEG	Reserved																C1G		UEVG								0		0		0					
	Reset Value																																				
0x18	TMRx_CCM1 output compare mode	Reserved																OC1MODE[2:0]		OC1PEN		OC1FEN								C1SEL[1:0]		C1SEL[1:0]		0			
	Reset Value																	0		0		0								0		0		0			
	TMRx_CCM1 input capture mode	Reserved																IC1DF[3:0]		IC1DIV[1:0]		C1SEL[1:0]								0		0		0			
	Reset Value																	0		0		0								0		0		0			

10.3.5.1 Control Register 1 (TMRx_CTRL1)

Address offset: 0x00

Reset value: 0x0000

Bit 15:10	Reserved. Always read as 0.
Bit 9:8	<p>CLKDIV[1:0]: Clock division Define the division ratio between the timer clock (CK_INT) frequency and sampling frequency used by the digital filters (ETR, TIx).</p> <p>00: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved</p>
Bit 7	<p>ARPEN: Auto-reload preload enable 0: TMRx_AR register is not buffered. 1: TMRx_AR register is buffered.</p>
Bit 6:3	Reserved. Always read as 0.
Bit 2	<p>UEVRS: Update request source This bit is set and cleared by software to select the UEV event sources.</p> <p>0: Any of the following events generates an update interrupt request if update interrupt is enabled: - Counter overflow - Setting the UEVG bit</p> <p>1: Only counter overflow generates an update interrupt request if update interrupt request is enabled.</p>
Bit 1	<p>UEVDIS: Update disable This bit is set and cleared by software to enable/disable UEV event generation.</p> <p>0: UEV is enabled. An update event (UEV) is generated by any of the following events: - Counter overflow - Setting the UEVG bit</p> <p>1: UEV is disabled. Update events are not generated, and shadow registers (AR, DIV, and CCx) keep their values. The counter and the prescaler are reinitialized if the UEVG bit is set or the slave mode controller sends a hardware reset.</p>
Bit 0	<p>CNTEN: Counter enable 0: The counter is disabled. 1: The counter is enabled.</p> <p>CNTEN is cleared automatically in one-pulse mode when an update event occurs.</p>

10.3.5.2 DMA/Interrupt Enable Register (TMRx_DIE)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														C1I E	UEV IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:2															
Bit 1														C1IE	UEVIE
Bit 0														0: Capture/Compare 1 interrupt is disabled. 1: Capture/Compare 1 interrupt is enabled.	0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.3.5.3 Status Register (TMRx_STS)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved							C1O F	Reserved							C1I F	UEV IF
res	rw	rw	rw	rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw	rw	
Bit 15:10																
Bit 9														C1OF	Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured as input capture mode. It is cleared by writing '0'. 0: No overcapture is detected. 1: When the counter value is captured to the TMRx_CC1 register, C1IF flag is already set.	
Bit 8:2																
Bit 1														C1IF	Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This flag is set by hardware when the counter value matches the compare value. It is cleared by software. 0: No match 1: The TMRx_CNT value matches the TMRx_CC1 value. C1F is only '1' at counter overflow. If channel CC1 is configured as input mode: This bit is set by hardware on a capture. It is cleared by software or by reading the TMRx_CC1 register. 0: No input capture occurs. 1: The counter value is captured to the TMRx_CC1 register (An edge which matches the selected polarity is detected on IC1).	
Bit 0														UEVIF	Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update event occurs. 1: Update interrupt is pending. This bit is set by hardware when the registers are updated: – If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when UEVG = 1 in the TMRx_EVEG register (the counter CNT is reinitialized by software). – If UEVDIS = 0 in the TMRx_CTRL1 register, the counter overflows.	

10.3.5.4 Event Generation Register (TMRx_EVEG)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														C1G	UEVG
res														rw	rw
Bit 15:2															
Bit 1															
C1G: Capture/Compare 1 generation This bit is set by software to generate a capture/compare event. It is cleared automatically by hardware. 0: No action 1: A capture/compare event is generated on channel 1 CC1: If channel CC1 is configured as output: Set C1IF = 1. The corresponding interrupt request is generated if enabled. If channel CC1 is configured as input: The current value of the counter is captured to the TMRx_CC1 register. Set C1IF = 1. The corresponding interrupt request is generated if enabled. If C1IF is already '1', set C1OF = 1.															
Bit 0															
UEVG: Update generation This bit can be set by software, and it is automatically cleared by hardware. 0: No action 1: Re-initialize the counter and generate an update. Please note that the prescaler counter is also cleared, but the prescaler ratio is not affected.															

10.3.5.5 Capture/Compare Mode Register 1 (TMRx_CCM1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by the corresponding CxSEL bits. All the other bits of this register have different functions in input and output modes. For a given bit, OCx describes its function when the channel is in output mode, ICx describes its function when the channel is in input mode. Attention must be given to the fact that the same bit can have different definitions for the input stage and for the output stage.

Output compare mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved														OC1PEN	OC1FEN
res														rw	rw
Bit 15:7															
Bit 6:4															
OC1MODE[2:0]: Output compare 1 enable The bits define the behavior of the output reference signal, OC1REF, and OC1REF determines the OC1 value. OC1REF is active high, while the active level of OC1 is determined by the C1P bit. 000: Frozen. The comparison between the output compare register TMRx_CC1 and the counter TMRx_CNT has no effect on OC1REF. 001: Set channel 1 to active level on match. OC1REF is forced high when the value of the counter TMRx_CNT matches the capture/compare register1 (TMRx_CC1). 010: Set channel 1 to inactive level on match. OC1REF is forced low when the value of the counter TMRx_CNT matches the capture/compare register1 (TMRx_CC1). 011: Toggle. Toggle OC1REF level when TMRx_CC1 = TMRx_CNT. 100: Force inactive level. OC1REF is forced low. 101: Force active level. OC1REF is forced high. 110: PWM mode1—Once TMRx_CNT < TMRx_CC1, channel 1 is active, else inactive. 111: PWM mode2—Once TMRx_CNT < TMRx_CC1, channel 1 inactive, else active. <i>Note: In PWM mode 1 or PWM mode 2, OC1REF level changes only when the comparison result changes or when output compare mode switches from frozen mode to PWM mode .</i>															

Bit 3	<p>OC1PEN: Output compare 1 preload enable 0: Preload function of TMRx_CC1 is disabled. TMRx_CC1 can be written at any time, and the new value takes effect immediately. 1: Preload function of TMRx_CC1 is enabled. Read/Write operations only access the preload register. TMRx_CC1 preload value is sent to the active register at each update event.</p> <p><i>Note: The PWM mode can be used without validating the preload register only in one-pulse mode (OPMODE = '1' in the TMRx_CTRL1 register). Else, the behavior is not guaranteed.</i></p>
Bit 2	<p>OC1FEN: Output compare 1 fast enable This bit is used to accelerate the CC output's response to trigger the input event. 0: CC1 behaves normally with the counter and CC1 values even when the trigger is ON. When an active edge occurs on the trigger input, the minimum delay to activate CC1 output is 5 clock cycles. 1: An active edge on the trigger input acts like a compare match. Therefore, OC is set to the compare level which is irrelevant to the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. This bit only acts when channel is configured as PWM1 or PWM2 mode.</p>
Bit 1:0	<p>C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: Reserved 11: Reserved</p> <p><i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i></p>

Input capture mode:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	IC1DF[3:0]	IC1DIV [1:0]	C1SEL [1:0]
res																		
Bit 15:8																		
Bit 7:4																		
Bit 3:2																		
Bit 1:0																		
IC1DF[3:0]: Input capture 1 filter																		
The bits define the frequency used to sample TI1 input and the length of digital filter. The digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output:																		
0000: No filter, sampling is done at f_{DTS}																		
0001: $f_{SAMPLING} = f_{CK_INT}$, N = 2																		
0010: $f_{SAMPLING} = f_{CK_INT}$, N = 4																		
0011: $f_{SAMPLING} = f_{CK_INT}$, N = 8																		
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6																		
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8																		
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6																		
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8																		
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6																		
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8																		
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5																		
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6																		
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8																		
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5																		
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6																		
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8																		
IC1DIV[1:0]: Input capture 1 prescaler																		
The bits define CC1 input (IC1) prescaler ratio.																		
Once C1EN = '0' (in the TMRx_CCE register), the prescaler is reset.																		
00: No prescaler. Capture is done each time when an edge is detected on the capture input.																		
01: Capture is done once every 2 events.																		
10: Capture is done once every 4 events.																		
11: Capture is done once every 8 events.																		
C1SEL[1:0]: Capture/Compare 1 selection																		
The bits define the channel direction (input/output), and input pin selection:																		
00: CC1 channel is configured as output.																		
01: CC1 channel is configured as input, IC1 is mapped on TI1.																		
10: Reserved																		
11: Reserved																		
<i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i>																		

10.3.5.6 Capture/Compare Enable Register (TMRx_CCE)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										CC1 NP	Reserve d	C1P	C1E N			
										res	rw	res	rw	rw		
Bit 15:4		Reserved. Always read as 0.														
Bit 3		C1NP: Capture/Compare 1 complementary output polarity CC1 channel is configured as output: C1NP should be kept cleared. CC1 channel is configured as input: C1NP combined with C1P defines TI1FP1 polarity (Please refer to C1P description).														
Bit 3		Reserved. Always read as 0.														
Bit 1		C1P: Capture/Compare 1 output polarity CC1 channel is configured as output: 0: OC1 is active high. 1: OC1 is active low. CC1 channel is configured as input: The CC1NP/CC1P bit select TI1FP1 polarity signal as trigger or capture signal. 00: Not inverted/Rising edge. Circuit is sensitive to TIxFP1 rising edge (capture mode). TIxFP1 is not inverted. 01: Inverted/Falling edge. Circuit is sensitive to TIxFP1 falling edge (capture mode). TIxFP1 is inverted. 10: Reserved 11: Not inverted/Both edges. Circuit is sensitive to both TIxFP1 rising and falling edge (capture mode). TIxFP1 is not inverted														
Bit 0		C1EN: Capture/Compare 1 output enable CC1 channel is configured as output: 0: OFF— OC1 output is disabled. 1: ON— OC1 is outputted on the corresponding output pin. CC1 channel is configured as input: This bit determines whether the counter value can be captured to TMRx_CC1 register. 0: Capture is disabled. 1: Capture is enabled.														

Table 10-10 Standard OCx Channel Output Control Bit

CxEN Bit	OCx Output State
0	Output disabled (OCx = 0, OCx_EN = 0)
1	OCx = OCxREF + Polarity, OCx_EN = 1

Note: The state of the external I/O pins connected to the standard OCx channels is determined by the OCx channel state and the GPIO and AFIO registers.

10.3.5.7 Counter (TMRx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

10.3.5.8 Prescaler (TMRx_DIV)

Address offset: 0x28

Reset value: 0x0000

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
DIV[15:0]								Bit 15:0							

Bit 15:0

DIV[15:0]: Prescaler value

The counter clock frequency CK_CNT is fCK_DIV/(DIV[15:0]+1).

DIV contains the value loaded in the active prescaler register at each update event.

10.3.5.9 Auto-reload Register (TMRx_AR)

Address offset: 0x2C

Reset value: 0x0000

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
AR[15:0]								Bit 15:0							

Bit 15:0

AR[15:0]: Auto-reload value

AR contains the value to be loaded to the actual auto-reload register.

10.3.5.10 Capture/Compare Register 1 (TMRx_CC1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15:0]															

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
CC1[15:0]								Bit 15:0							

Bit 15:0

CC1[15:0]: Capture/Compare 1 value

If CC1 channel is configured as output:

CC1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM1 register (the OC1PEN bit). Else, the preload value is loaded in the active capture/compare 1 register when an update event occurs.

The active capture/compare register involves in the comparison with the counter, TMRx_CNT, and generates the output on OC1.

If CC1 channel is configured as input:

CC1 is the counter value transferred by the last input capture 1 event (IC1).

10.4 Advanced-control Timer (TMR1, TMR8, and TMR15)

10.4.1 TMR1, TMR8, and TMR15 Introduction

The advanced-control timer (TMR1, TMR8, and TMR15) consist of a 16-bit auto-reload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, embedded dead-time, and complementary PWM, and so on).

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

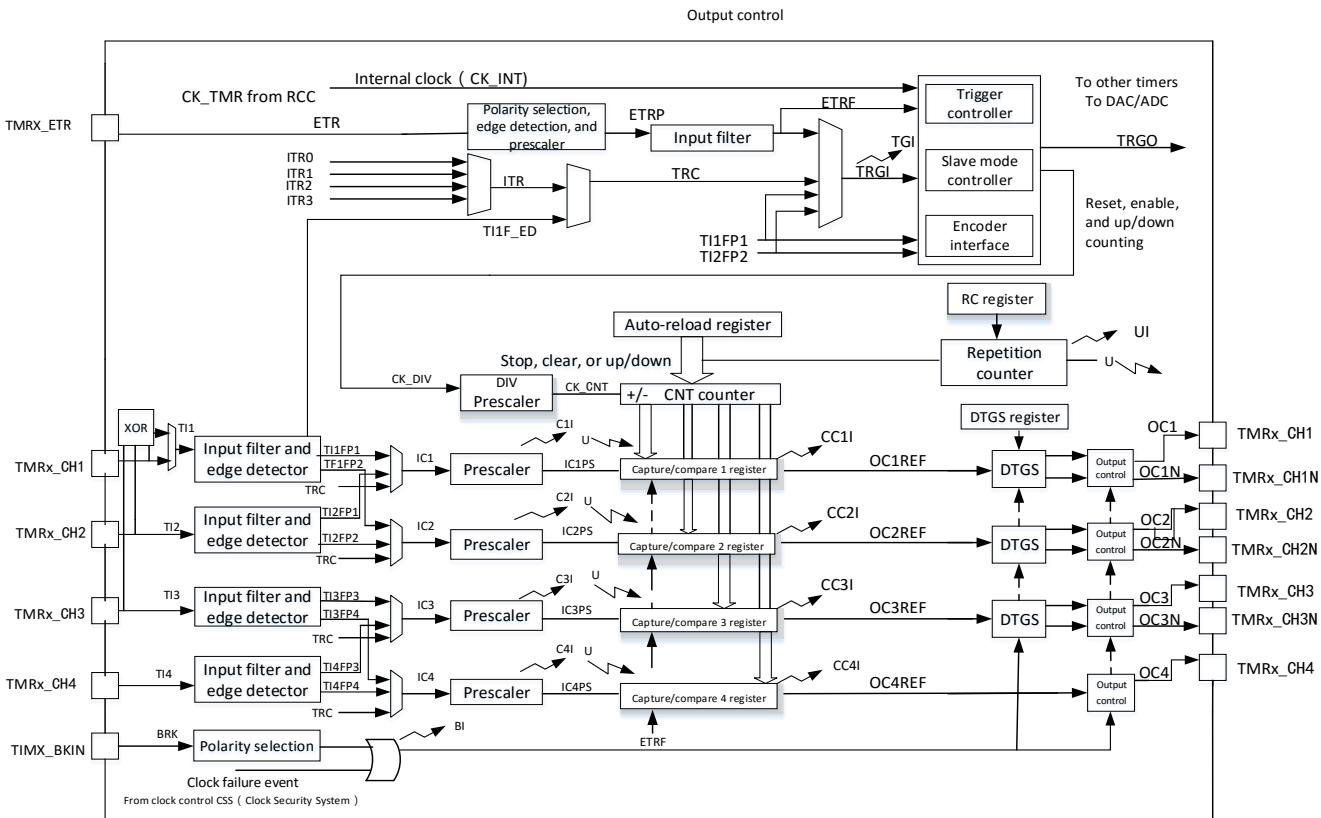
The advanced-control timer (TMR1, TMR8, and TMR15) and general-purpose timers (TMRx) are completely independent, do not share any resources, and can be synchronized. Please refer to [Section 10.4.3.20](#).

10.4.2 TMR1, TMR8, and TMR15 Main Features

The main functions of TMR1, TMR8, and TMR15 include:

- 16-bit up, down, and up/down auto-reload counter
- 16-bit programmable prescaler (can be modified on the fly) used to divide the counter clock frequency by any factor between 1 ~ 65536
- 4 independent channels for:
 - Input capture
 - Output compare
 - PWM generation (Edge-aligned or center-aligned modes)
 - One-pulse mode output
- Complementary outputs with programmable dead-time
- Timers and the interconnected synchronization circuit are controlled by external signals.
- Repetition counter to update the timer registers after a given number of counter cycles
- Break input to put the timer's output signals at reset state or at a known state
- Interrupt/DMA is generated at the following events:
 - Update: Counter overflow/underflow, counter initialization (by software or internal/external trigger)
 - Trigger event (counter starts, stops, initialized, or counts by internal/external trigger)
 - Input capture
 - Output compare
 - Break signal input
- Supports incremental (quadrature) encoder and Hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management

Figure 10-80 Block Diagram of Advanced-control Timer



Note: transfers preload registers to active registers on U event according to the control bits.

Event

Interrupt and DMA output

10.4.3 TMR1, TMR8, and TMR15 Function Overview

10.4.3.1 Time-base Unit

This programmable advanced-control timer mainly consists of a 16-bit counter and relevant auto-reload registers. The counter can count up, down, or both up and down. The counter clock is obtained through a prescaler.

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

The time-base unit includes:

- Counter register (TMRx_CNT)
- Prescaler register (TMRx_DIV)
- Auto-reload register (TMRx_AR)
- Repetition counter register (TMRx_RC)

The auto-reload register is preloaded. Each read or write to the auto-reload register will access the preload register. The contents written to the preload register are transferred into its shadow register immediately or at each update event (UEV) according to the auto-reload preload enable bit (ARPEN) in the TMRx_CTRL1 register. When the UEVDIS bit in the TMRx_CTRL1 register is '0', update events will be generated once the counter reaches the overflow value (or underflow when downcounting). Update events can also be generated by software. Please refer to the following sections for the detailed introduction on generation of the update events.

The counter is driven by the prescaler clock output CK_CNT, and it is enabled when the counter enable bit (CNTEN) in the TMRx_CTRL1 register is set. (Please refer to the controller slave mode descriptions for more details on counter enable.)

Please note that after the CNTEN bit in the TMRx_CTRL register is set, the counter starts counting after one clock cycle.

Prescaler

The prescaler can divide the counter clock frequency by any factor between 1 ~ 65536, and this is achieved with a 16-bit counter controlled by a 16-bit register (in the TMRx_DIV register). Its value can be changed on the fly since the control register is buffered. The new prescaler ratio takes effect at the next update event.

[Figure 10-81](#) and [Figure 10-82](#) are examples of on-the-fly prescaler ratio change.

Figure 10-81 Counter Timing Diagram with Prescaler Division Changing from 1 to 2

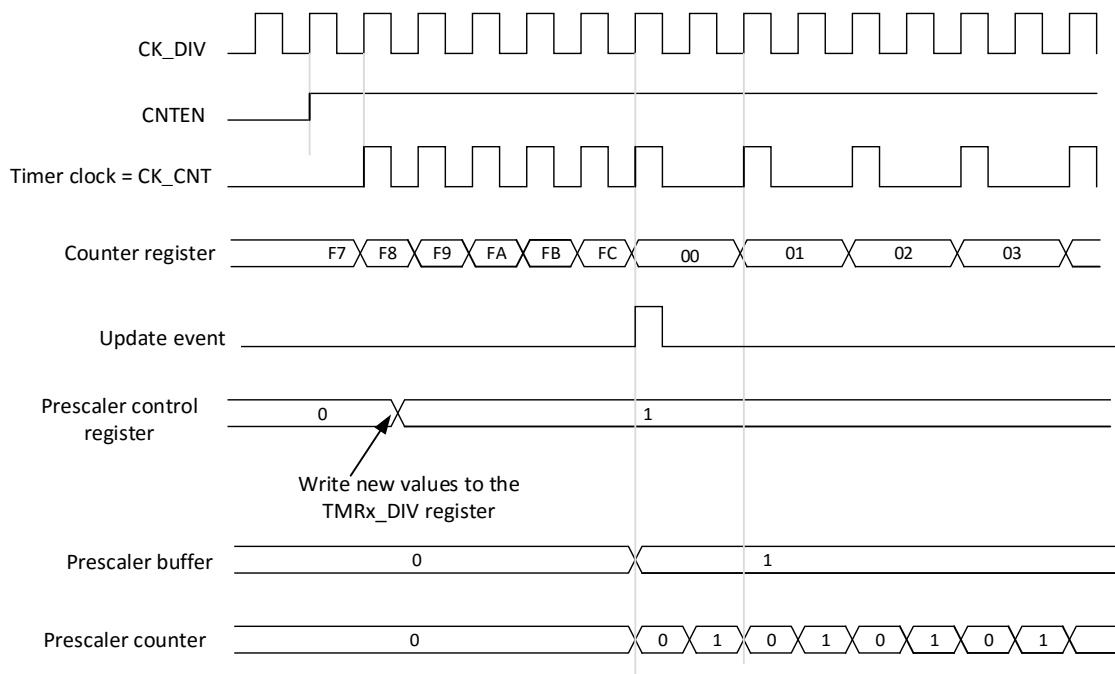
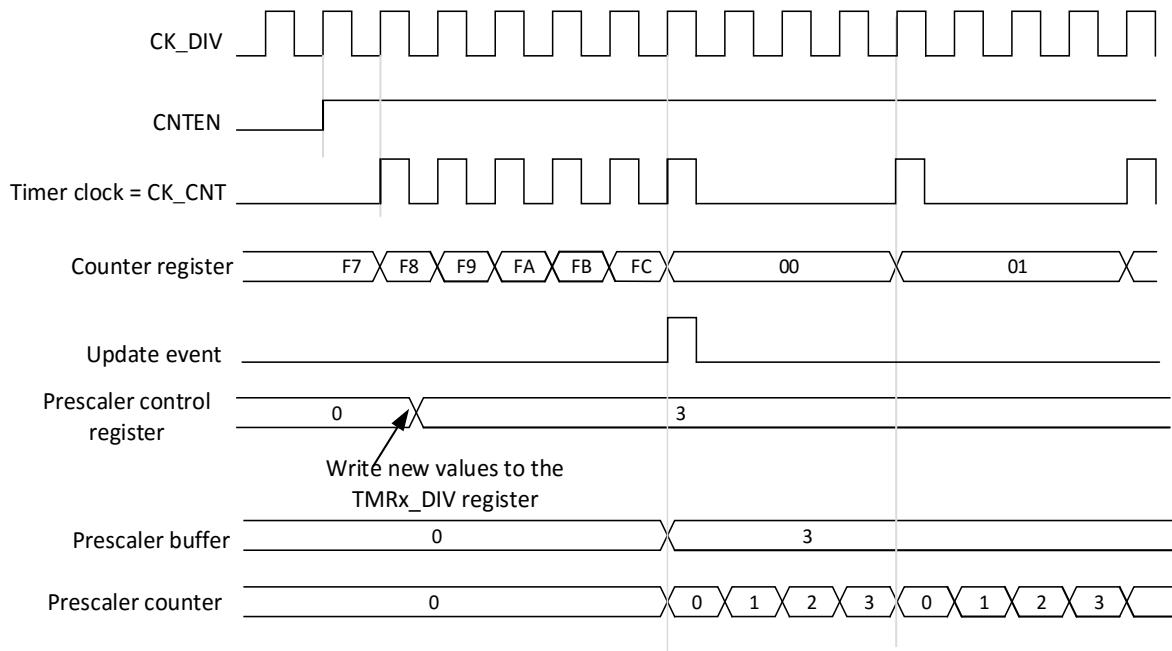


Figure 10-82 Counter Timing Diagram with Prescaler Division Changing from 1 to 4



10.4.3.2 Counter Mode

Upcounting mode

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

If the repetition counter is used, the update event (UEV) is generated after upcounting reaches the programmed count in the repetition counter register (TMRx_RC). Else, the update event is generated at each counter overflow.

An update event can also be generated by setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller).

Update events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing new values to the preload registers. No update event occurs until the UEVDIS bit is cleared. However, when the update events are generated, the counter and the prescaler are still cleared (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set by hardware (that is, no interrupt or DMA request is sent). This can avoid update and capture interrupt generation when the counter is cleared in capture mode.

When an update event occurs, all the registers are updated, and the update flag (the UEVIF bit in the TMRx_STS register) is set by hardware (according to the UEVRS bit).

- The repetition counter is reloaded with the content of TMRx_RC register.
- The auto-reload shadow register is updated with the preload value (TMRx_AR).
- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).

Figure 10-83 ~ Figure 10-88 show the examples of the counter behaviors for different clock frequencies when TMRx_AR = 0x36:

Figure 10-83 Counter Timing Diagram with Internal Clock Divided by 1

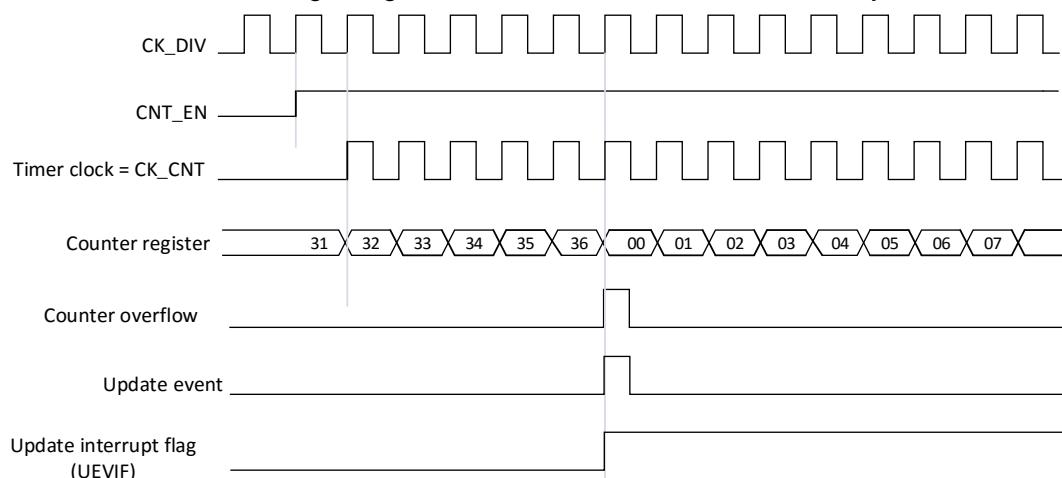


Figure 10-84 Counter Timing Diagram with Internal Clock Divided by 2

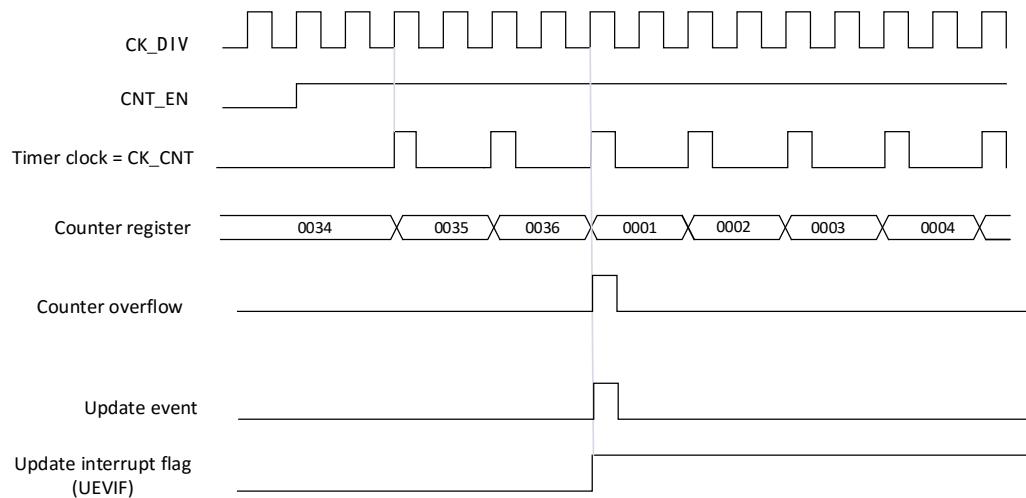


Figure 10-85 Counter Timing Diagram with Internal Clock Divided by 4

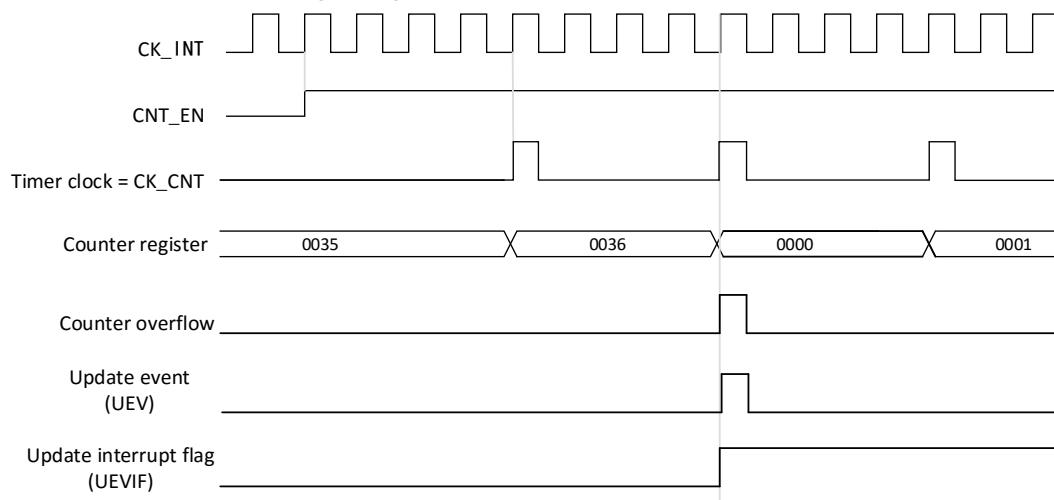


Figure 10-86 Counter Timing Diagram with Internal Clock Divided by N

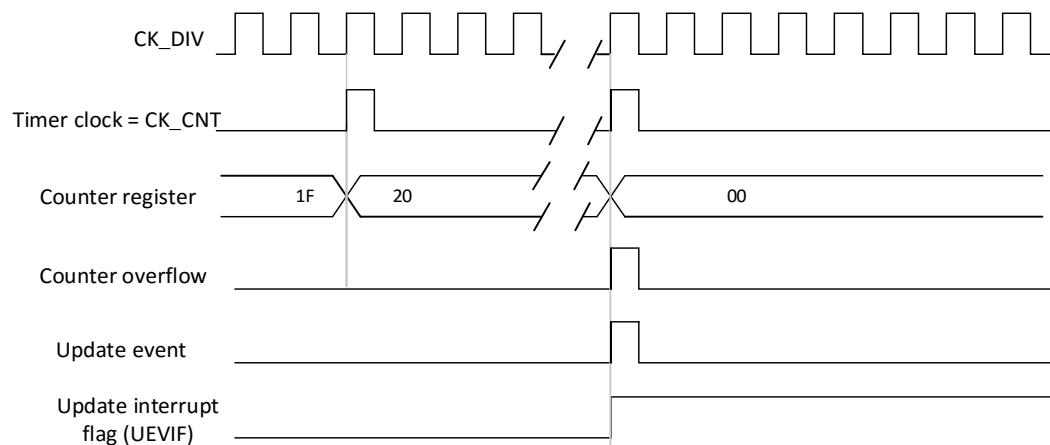


Figure 10-87 Counter Timing Diagram with Update Event When ARPEN = 0 (TMRx_AR Not Preloaded)

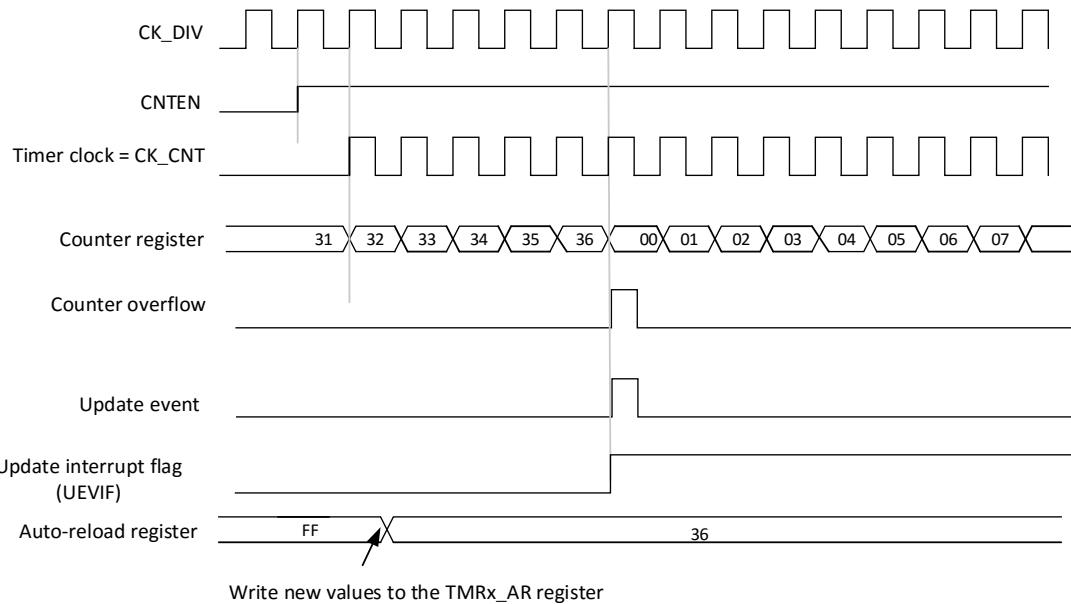
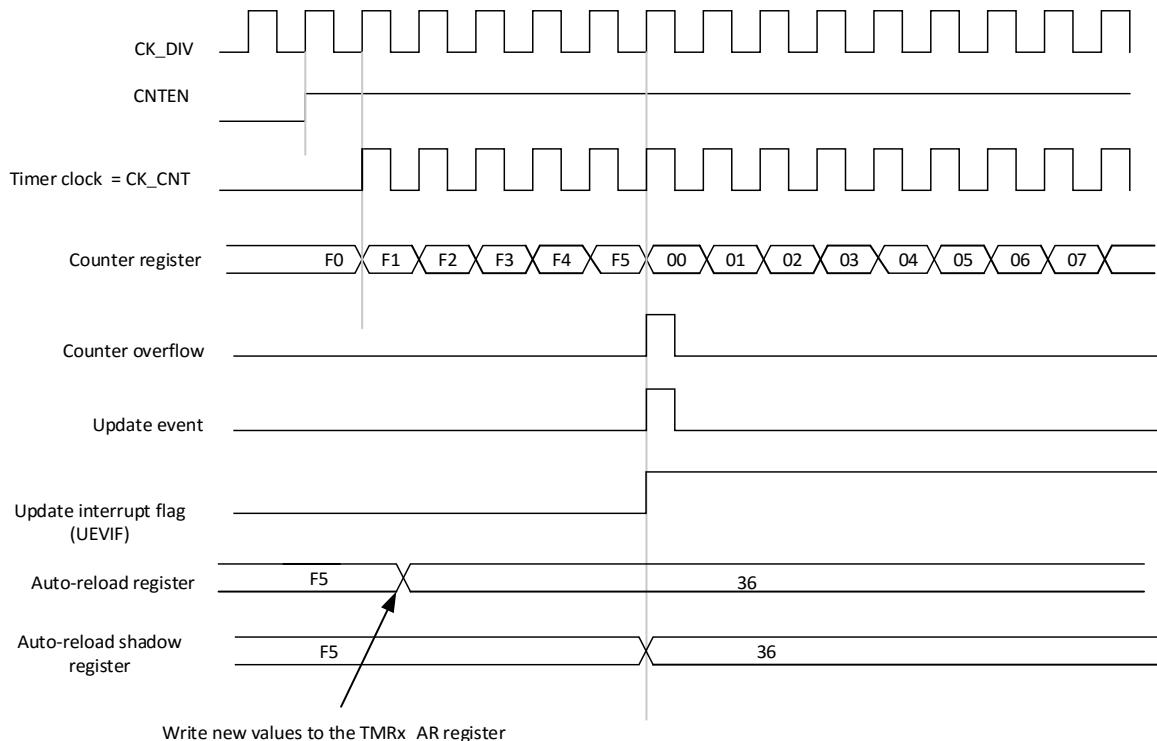


Figure 10-88 Counter Timing Diagram with Update Event When ARPEN = 1 (TMRx_AR is Preloaded)



Downcounting mode

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

If the repetition counter is used, an update event (UEV) is generated after downcounting reaches the programmed count in the repetition counter register (TMRx_RC). Else, the update event is generated at each counter underflow.

An update event can also be generated by setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller).

Update events can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This avoids changing the shadow registers while writing new values to the preload registers. Thus, no update event occurs until the UEVDIS bit is cleared. However, the counter still counts from the current auto-reload

value, and the prescaler counter restarts from 0 (but the prescaler ratio does not change). In addition, if the UEVRS (update request selection) bit in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV, but the UEVIF flag is not set (so no interrupt or DMA request is generated). This can avoid update and capture interrupts generation when a capture event occurs and the counter is cleared.

When an update event occurs, all the registers are updated, and the update flag bit (the UEVIF bit in the TMRx_STS register) is set (according to the UEVRS bit).

- The repetition counter is reloaded with the content of TMRx_RC register.
- The current auto-reload register is updated with the preload value (content of the TMRx_AR register).
- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).

Note: Auto-reload is updated before the counter is reloaded, so the next cycle will be the expected value.

Figure 10-89 ~ Figure 10-93 show the examples of the counter behavior for different clock frequencies when TMRx_AR = 0x36:

Figure 10-89 Counter Timing Diagram with Internal Clock Divided by 1

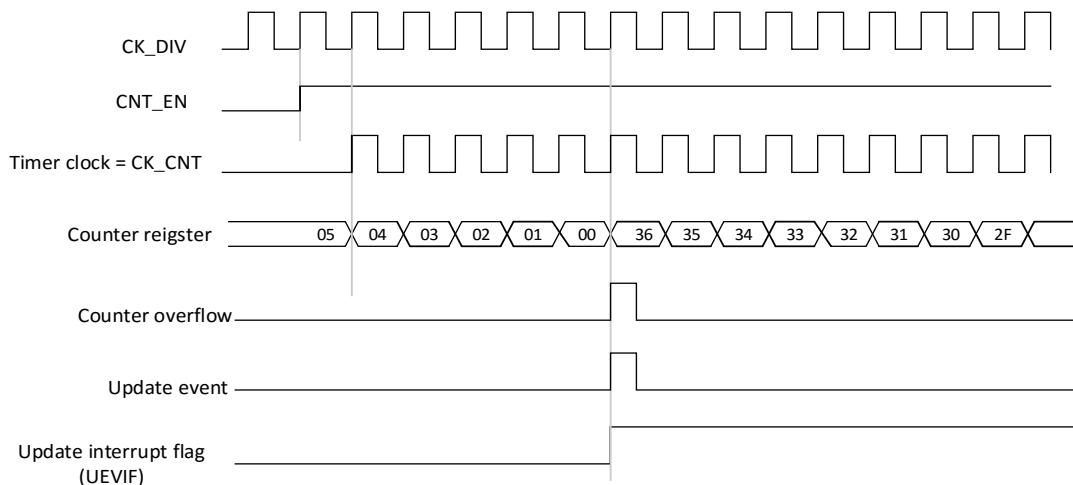


Figure 10-90 Counter Timing Diagram with Internal Clock Divided by 2

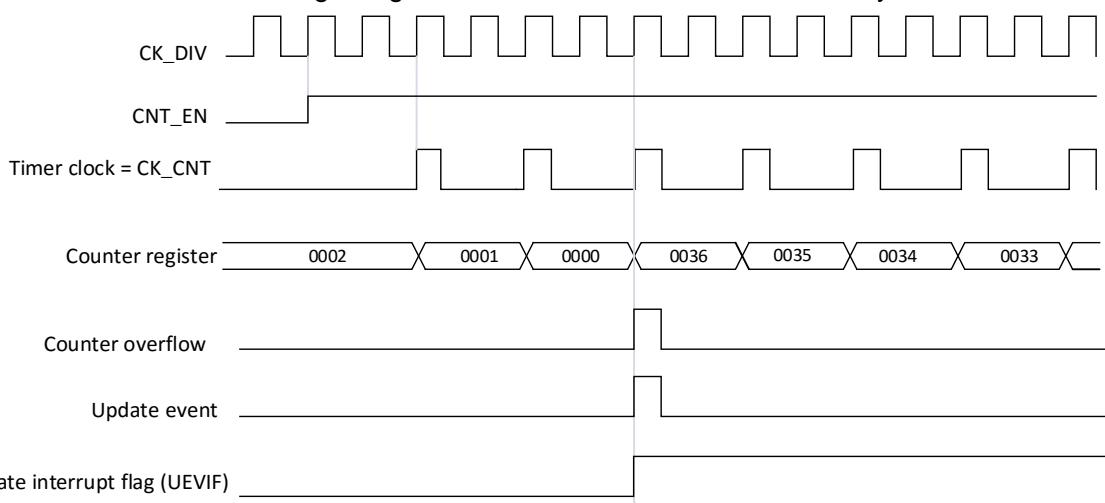


Figure 10-91 Counter Timing Diagram with Internal Clock Divided by 4

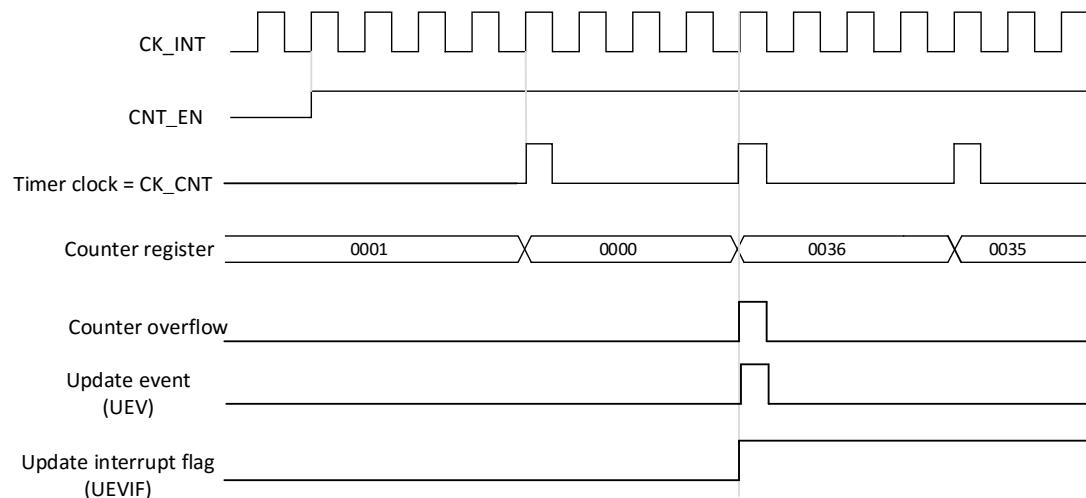


Figure 10-92 Counter Timing Diagram with Internal Clock Divided by N

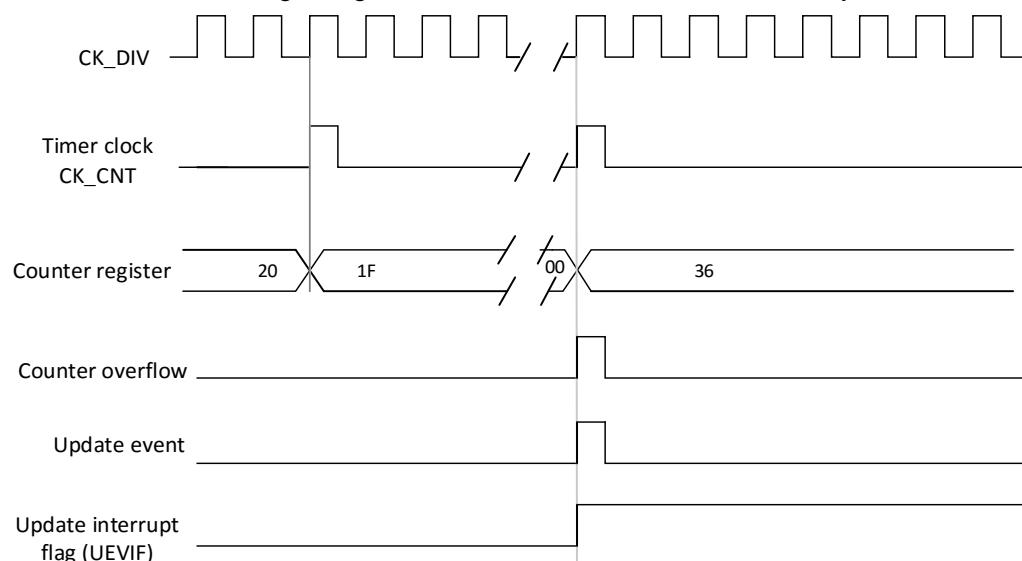
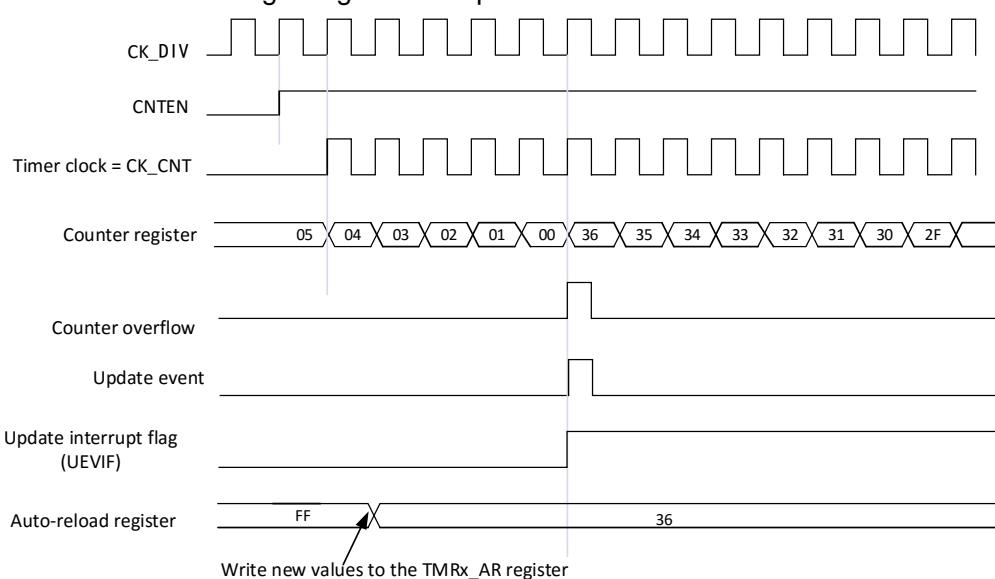


Figure 10-93 Counter Timing Diagram of Update Event without Auto-reload



Center-aligned mode (Up/Down counting)

In center-aligned mode, the counter counts from 0 to the auto-reload value (content of the TMRx_AR register) – 1, generates a counter overflow event, and counts from the auto-reload value down to 1 and

generates a counter underflow event. Then, it restarts counting from 0.

In this mode, the DIR direction bit in the TMRx_CTRL1 register cannot be written. It is updated by hardware and indicates the current counting direction.

The update event can be generated at each counter overflow and at each counter underflow, or by setting the UEVG bit in the TMRx_EVEG register (by software or by using the slave mode controller). Afterwards, the counter as well as the prescaler restarts counting from 0.

The UEV update event can be disabled by setting the UEVDIS bit in the TMRx_CTRL1 register. This can avoid updating the shadow registers while writing new values to the preload registers. Therefore, no update event occurs until the UEVDIS bit is cleared. However, the counter continues counting up or down based on the current auto-reload value. In addition, if the UEVRS bit (update request selection) in the TMRx_CTRL1 register is set, setting the UEVG bit generates an update event UEV without setting the UEVIF flag (so no interrupt or DMA request is generated). This can avoid generating both update and capture interrupts when the capture event occurs and the counter is cleared.

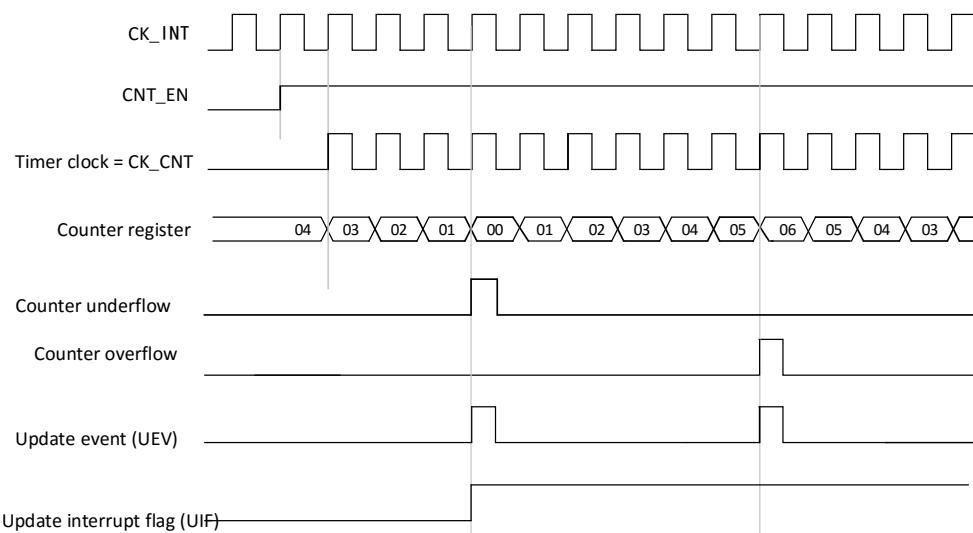
When an update event occurs, all the registers are updated, and the update flag bit (the UEVIF bit in the TMRx_STS register) is set (according to the UEVRS bit).

- The repetition counter is reloaded with the content of TMRx_RC register.
- The buffer of the prescaler is reloaded with the preload value (content of the TMRx_DIV register).
- The current auto-reload register is updated with the preload value (content of the TMRx_AR register).

Note: *If an update is generated because of counter overflow, auto-reload is updated before the counter is reloaded, so the next cycle will be the expected value (the counter is reloaded with the new value).*

Figure 10-94 ~ Figure 10-99 show the examples of the counter behavior for different clock frequencies:

Figure 10-94 Counter Timing Diagram with Internal Clock Divided by 1, TMRx_AR = 0x6



Note: Center-aligned mode 1 is used here (Please refer to [Section 10.4.4.1](#) for the register description).

Figure 10-95 Counter Timing Diagram with Internal Clock Divided by 2

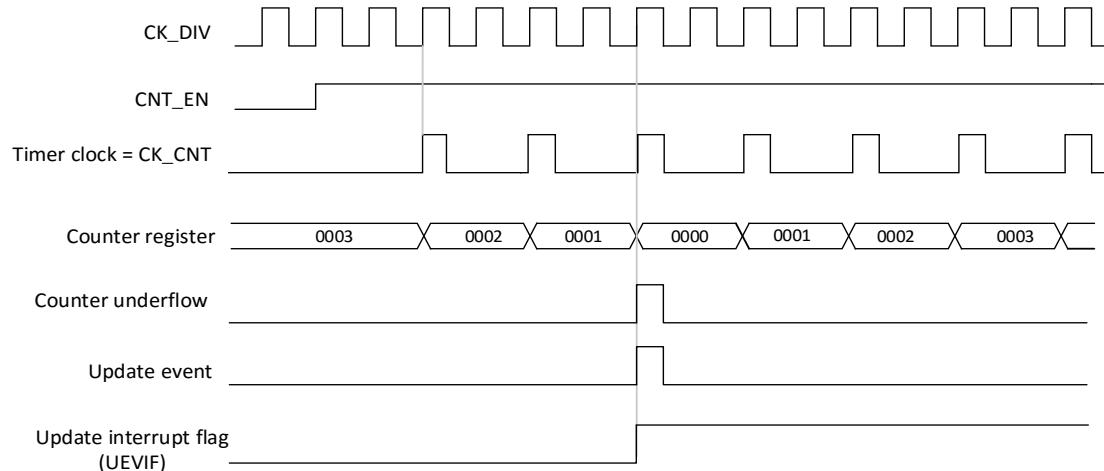
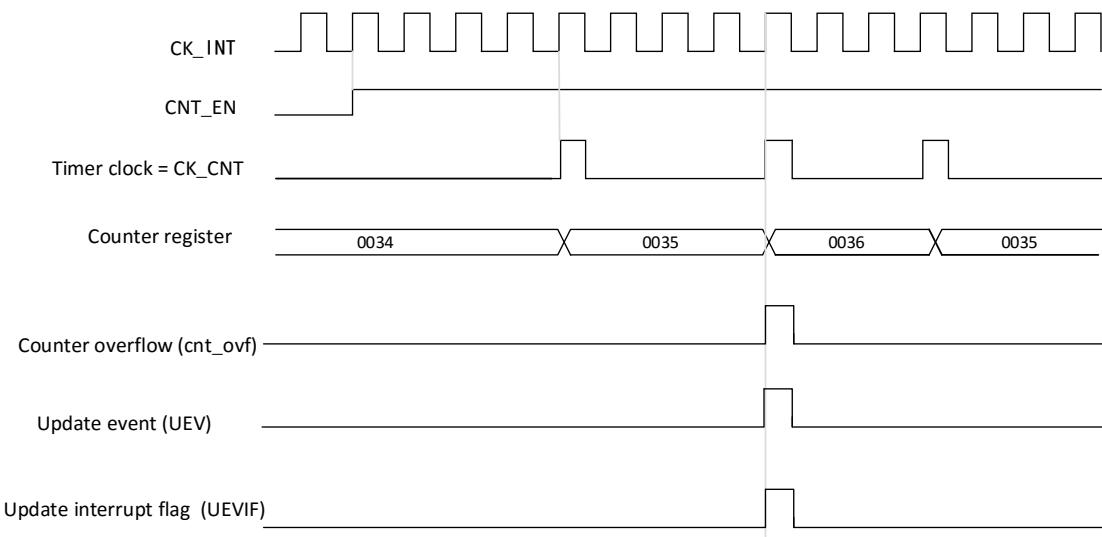


Figure 10-96 Counter Timing Diagram with Internal Clock Divided by 4, TMRx_AR = 0x36



Note: Center-aligned mode 2 or 3 is used here, set UEVIF at counter overflow.

Figure 10-97 Counter Timing Diagram with Internal Clock Divided by N

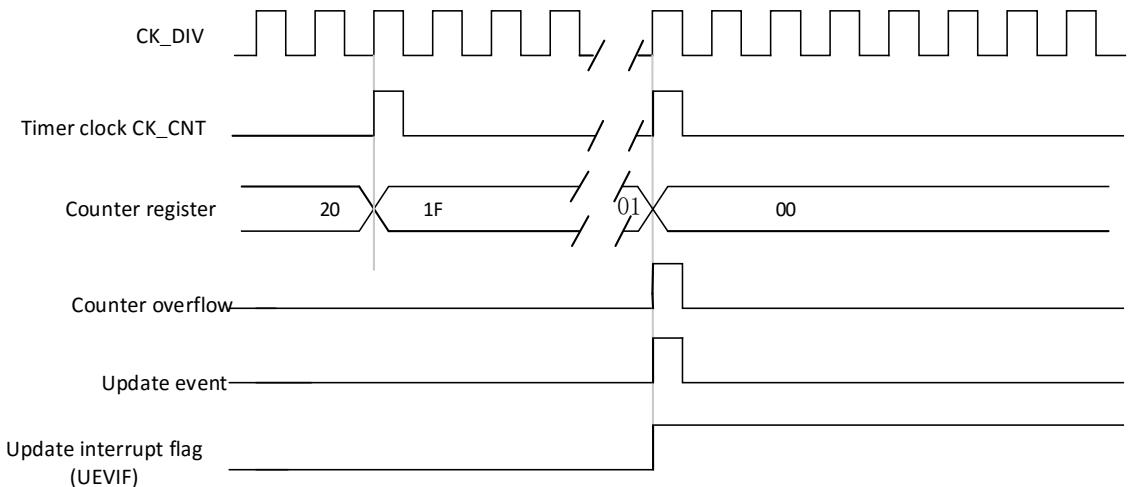


Figure 10-98 Counter Timing Diagram with Update Event When ARPEN = 1 (Counter underflow)

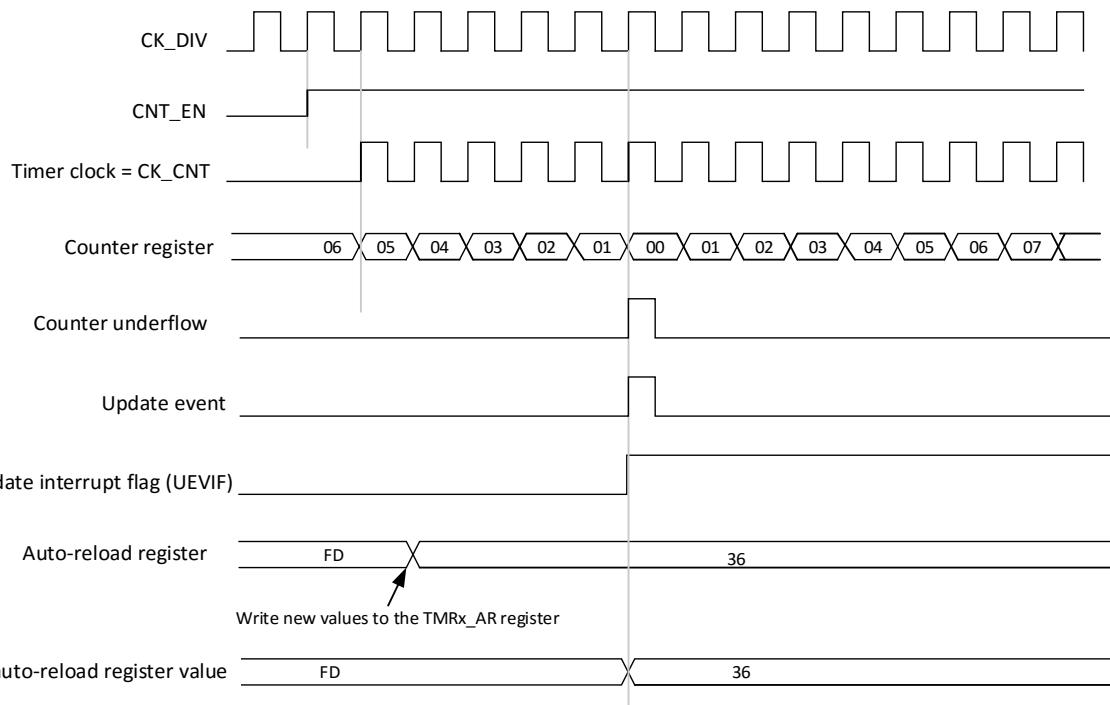
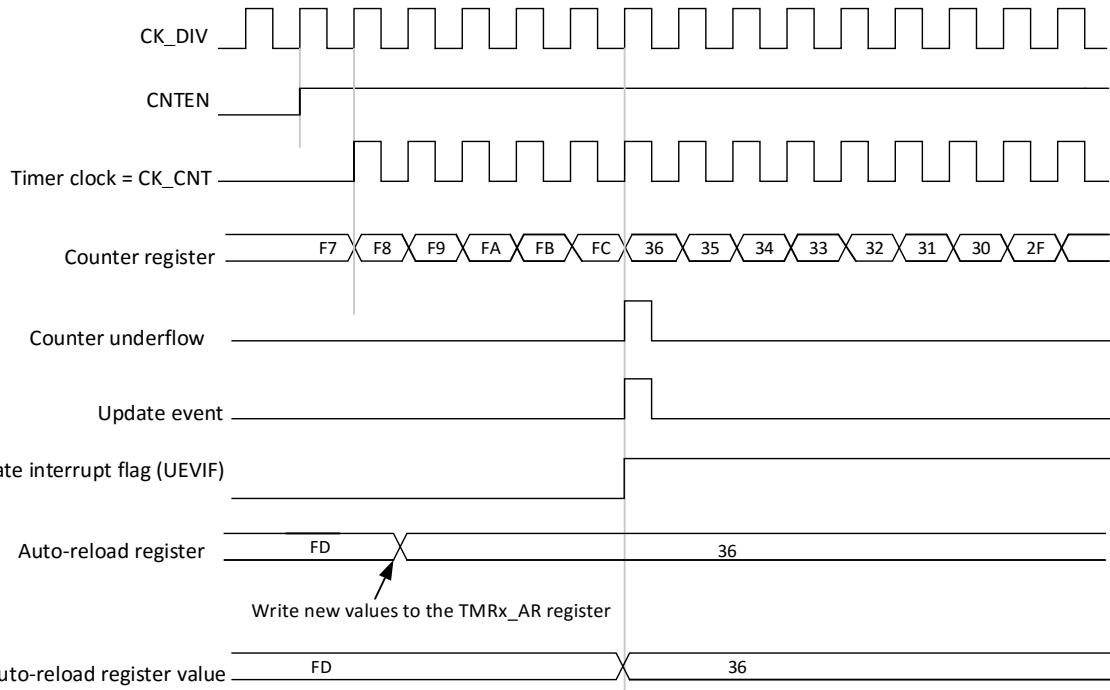


Figure 10-99 Counter Timing Diagram with Update Event When ARPEN = 1 (Counter overflow)



10.4.3.3 Repetition Counter

[Section 10.4.3.1](#) describes how an update event (UEV) is generated on counter overflows/underflows. It is actually generated only when the repetition counter reaches '0'. This feature can be very useful when generating PWM signals.

This means that when every N times the counter overflows or underflows, data is transferred from the preload registers to the shadow registers (TMRx_AR auto-reload register, TMRx_DIV preload register, also capture/compare registers (TMRx_CCx in compare mode), where N is the value in the TMRx_RC repetition counter register.

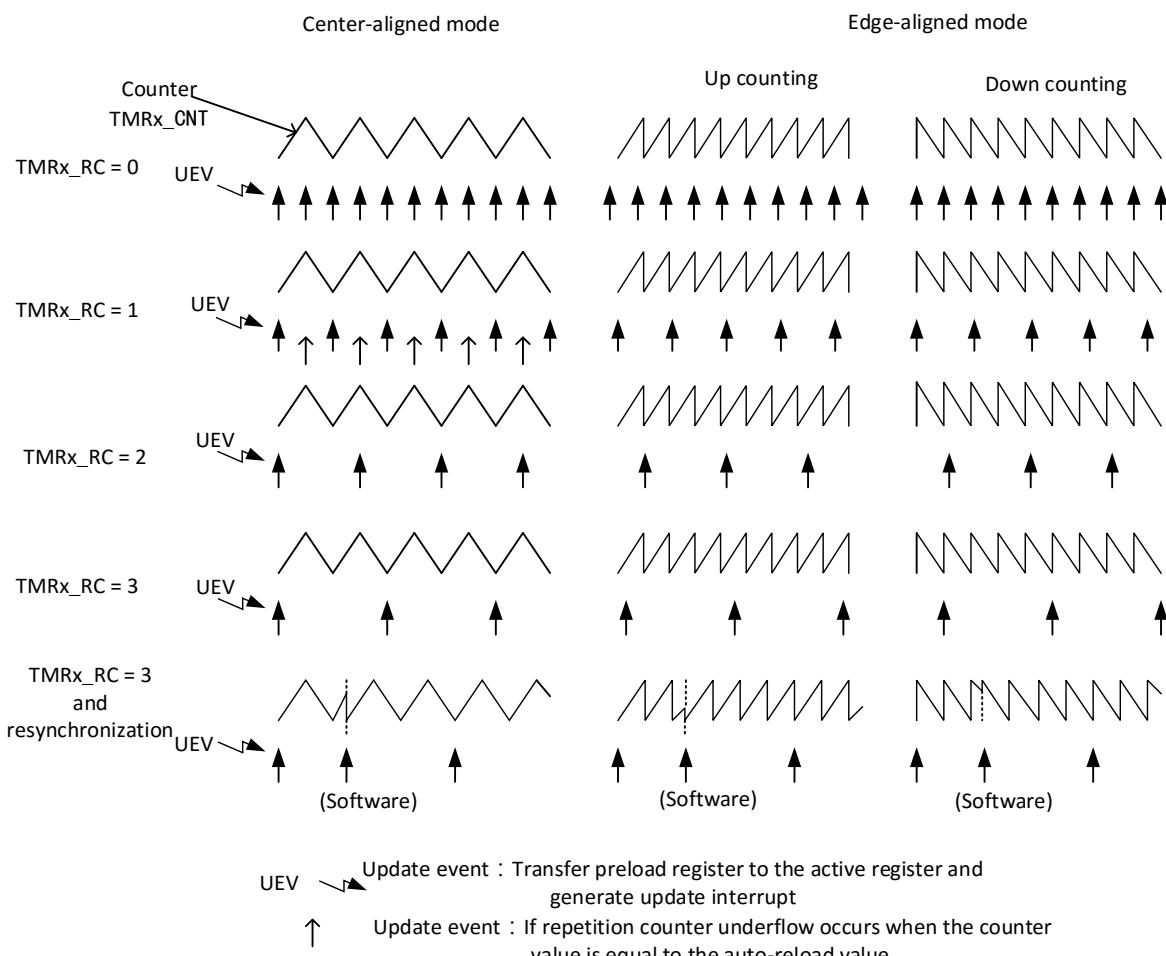
The repetition counter is decremented under any of the following conditions:

- At each counter overflow in upcounting mode
 - At each counter underflow in downcounting mode
 - At each counter overflow and at each counter underflow in center-aligned mode

Although this limits the maximum number of repetition to 128 PWM cycles, it makes updating the duty cycle twice per PWM period possible. Because of the symmetry of waveforms, if the compare registers are refreshed only once per PWM period in center-aligned mode, the maximum resolution is $2xT_{ck}$.

The repetition counter is auto-reloaded, and the repetition rate is defined by the TMRx_RC register value (Please refer to [Figure 10-100](#)). When an update event is generated by software (by setting the UEVG bit in the TMRx_EVEG register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is, and the repetition counter is reloaded with the content of the TMRx RC register.

Figure 10-100 Examples of Updating Rates in Different Modes and the TMRx RC Register Configuration



10.4.3.4 Clock Selection

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

- External clock mode 1: External input pin
 - External clock mode 2: External trigger input ETR
 - Internal trigger inputs (ITRx): Using one timer as the prescaler for another timer. For example, users can configure Timer 1 to act as the prescaler for Timer 2. Please refer to [Section 10.4.3.20](#) for more details.

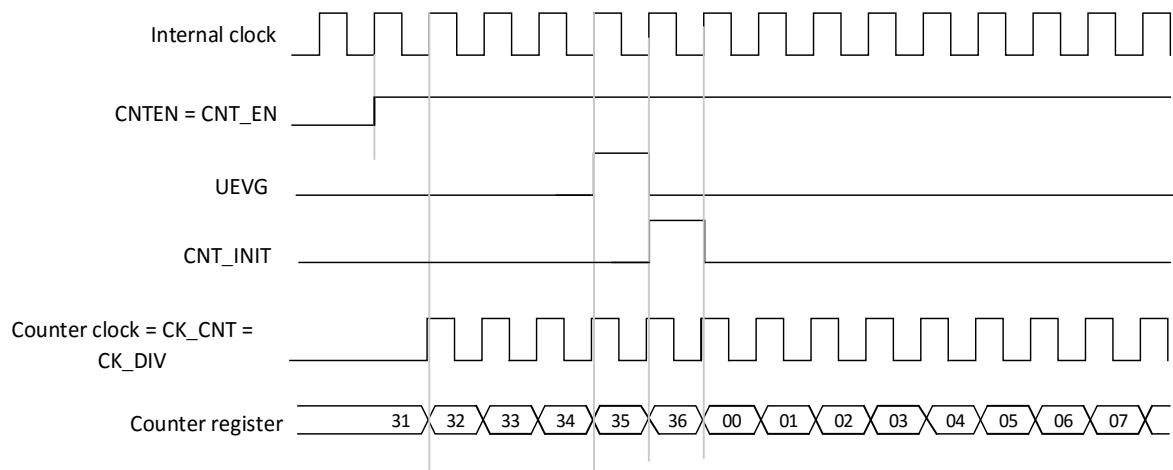
Internal clock source (CK_INT)

If the slave mode controller is disabled ($\text{SMSEL} = 000$), the **CNTEN**, **DIR** (in the **TMRx_CTRL1** register), and **UEVG** bits (in the **TMRx_EVEG** register) are the actual control bits, and they can be changed only by software (except for the **UEVG** bit, which will be cleared automatically). As soon as the **CNTEN** bit is

written '1', the prescaler is clocked by the internal clock CK_INT.

Figure 10-101 shows the behavior of the control circuit and the up counter in normal mode, without prescaler.

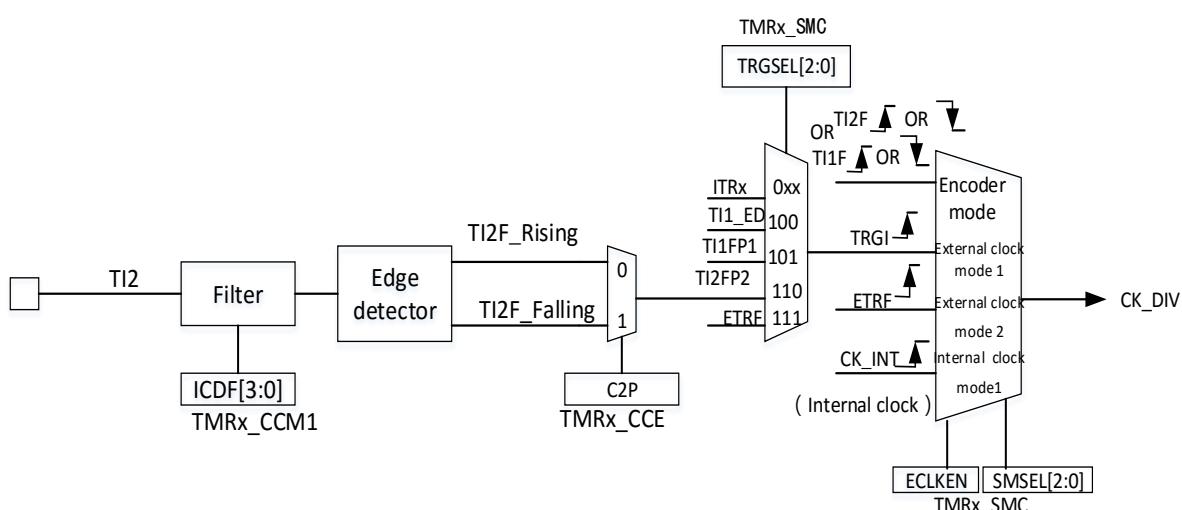
Figure 10-101 Control Circuit in Normal Mode with Internal Clock Divided by 1



External clock mode 1

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

Figure 10-102 TI2 External Clock Connection Example



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

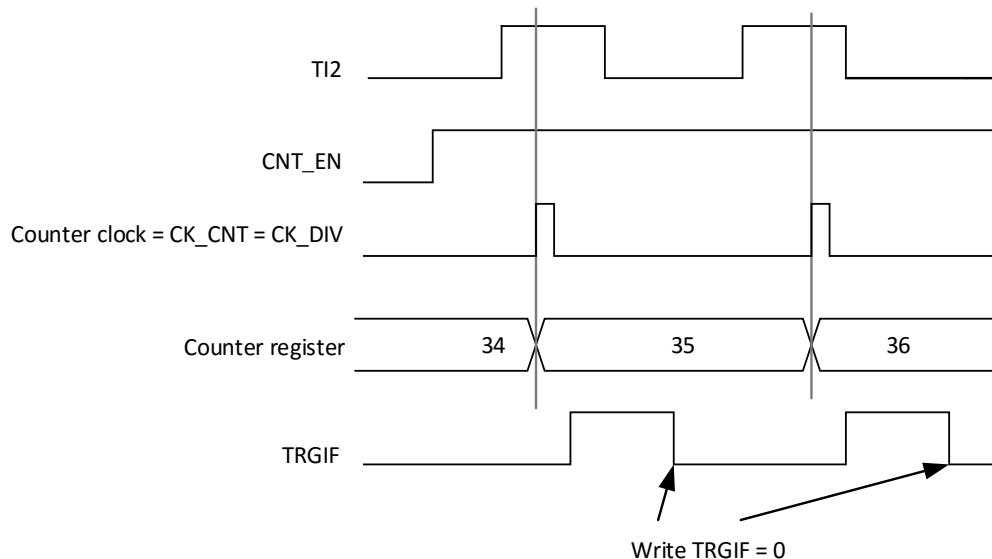
1. Configure channel 2 to detect the rising edges on the TI2 input by writing C2SEL = '01' in the TMRx_CCM1 register.
2. Configure the input filter duration by writing the IC2DF[3:0] bits in the TMRx_CCM1 register (if no filter is needed, keep IC2DF = 0000).
3. Select the rising edge polarity by writing C2P = '0' in the TMRx_CCE register.
4. Select the timer external clock mode 1 by writing SMSEL = '111' in the TMRx_SMC register.
5. Select TI2 as the trigger input source by writing TRGSEL = '110' in the TMRx_SMC register.
6. Enable the counter by writing CNTEN = '1' in the TMRx_CTRL1 register.

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

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

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

Figure 10-103 Control Circuit in External Clock Mode 1



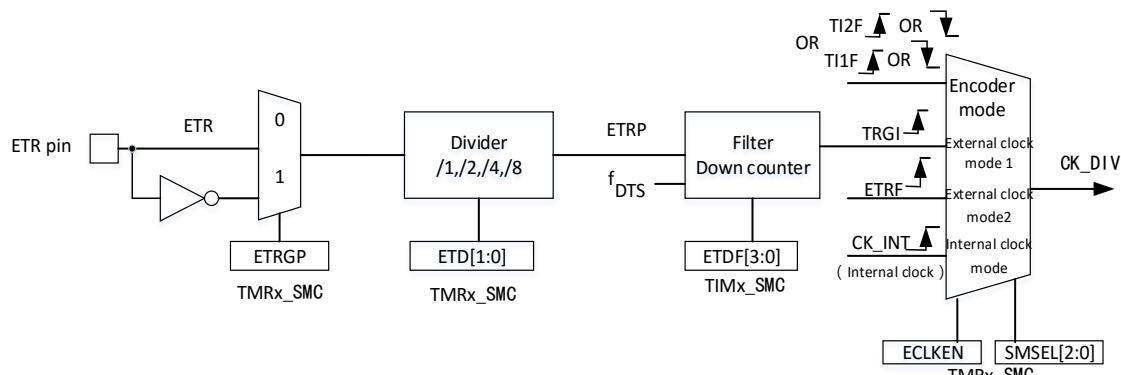
External clock mode 2

This mode is selected by writing ECLKEN = 1 in the TMRx_SMC register.

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

Figure 10-104 shows the block diagram of the external trigger input.

Figure 10-104 Block Diagram of External Trigger Input



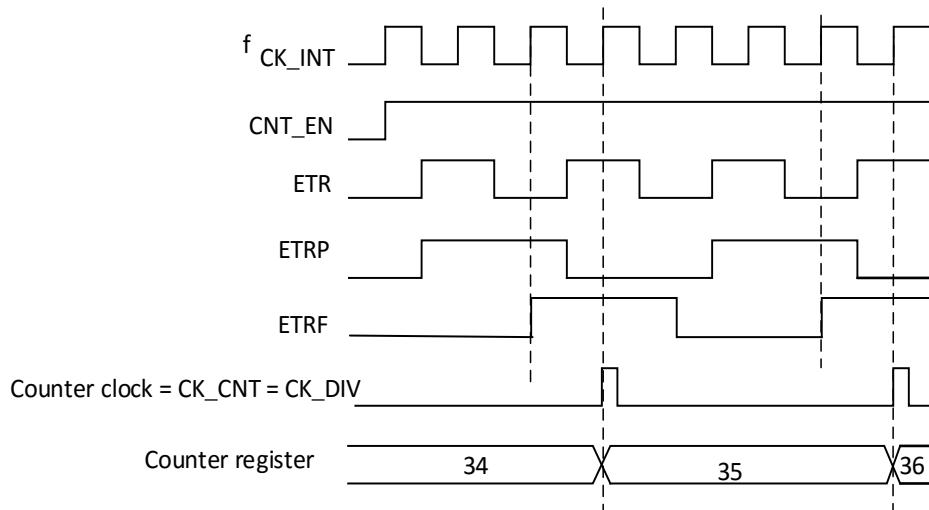
For example, to configure the up counter to count each 2 rising edges on ETR, use the following procedure:

1. Since no filter is needed in this example, write ETDF[3:0] = 0000 in the TMRx_SMC register.
2. Set the prescaler by writing ETD[1:0] = 01 in the TMRx_SMC register.
3. Select the rising edge detection on ETR by writing ETRGP = 0 in the TMRx_SMC register.
4. Enable the external clock mode 2 by writing ECLKEN = 1 in the TMRx_SMC register.
5. Enable the counter by writing CNTEN = 1 in the TMRx_CTRL1 register.

The counter counts once each 2 ETR rising edges.

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

Figure 10-105 Control Circuit in External Clock Mode 2

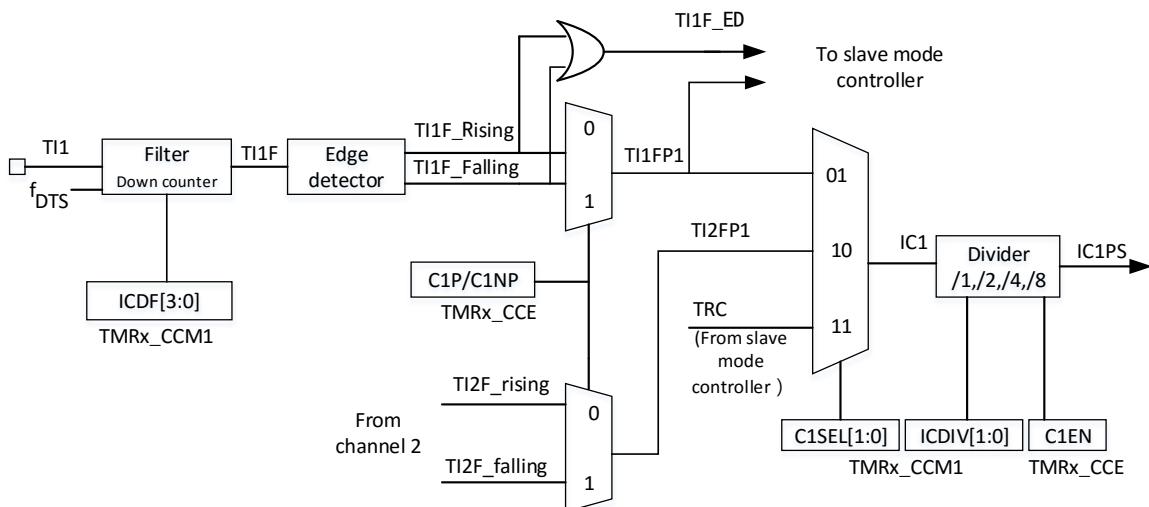


10.4.3.5 Capture/Compare Channel

Each capture/compare channel is built around a capture/compare register (including a shadow register), also an input stage for capture (digital filter, multiplexing, and prescaler) and an output stage (comparator and output control). Figure 10-106 ~ Figure 10-109 provide an overview of one capture/compare channel.

The input stage samples the corresponding Tlx input signal to generate a filtered signal Tlx_F . Then, an edge detector with polarity selection generates a signal (Tlx_FPx) which can be the trigger input for the slave mode controller, or be the capture command. This signal enters the capture register ($ICxPS$) through the prescaler.

Figure 10-106 Capture/Compare Channel (e.g. Channel 1 Input Stage)



The output stage generates an intermediate waveform $OCxRef$ (active high) which serves as reference. The polarity of final output signal is determined by the end of the chain.

Figure 10-107 Capture/Compare Channel 1 Main Circuit

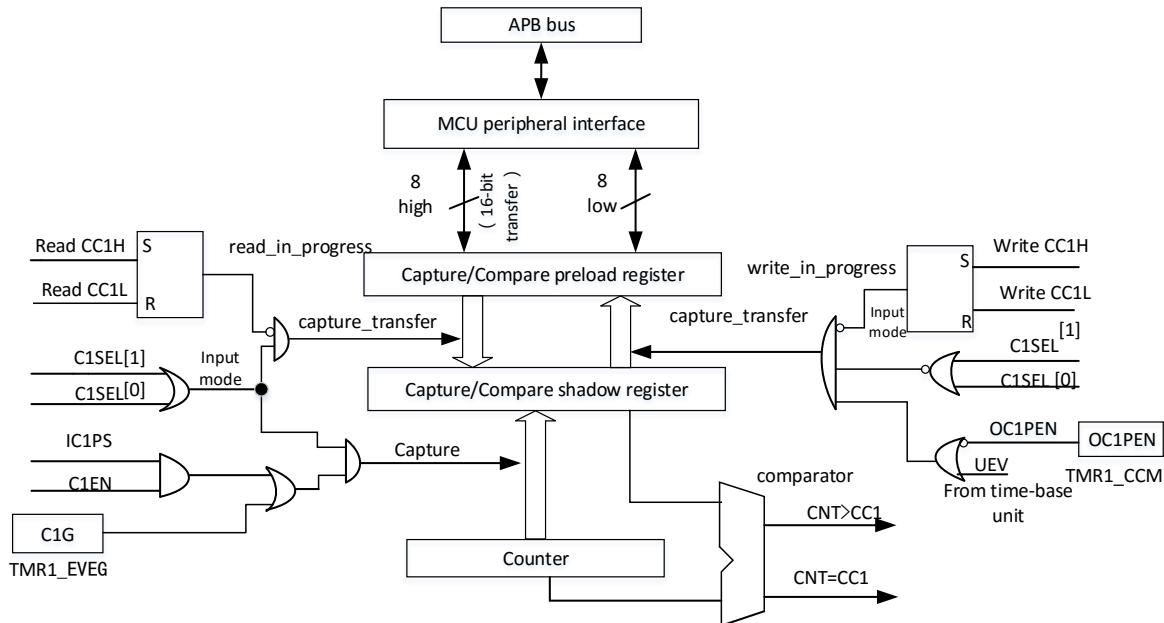


Figure 10-108 Capture/Compare Channel Output Stage (Channel 1 to 3)

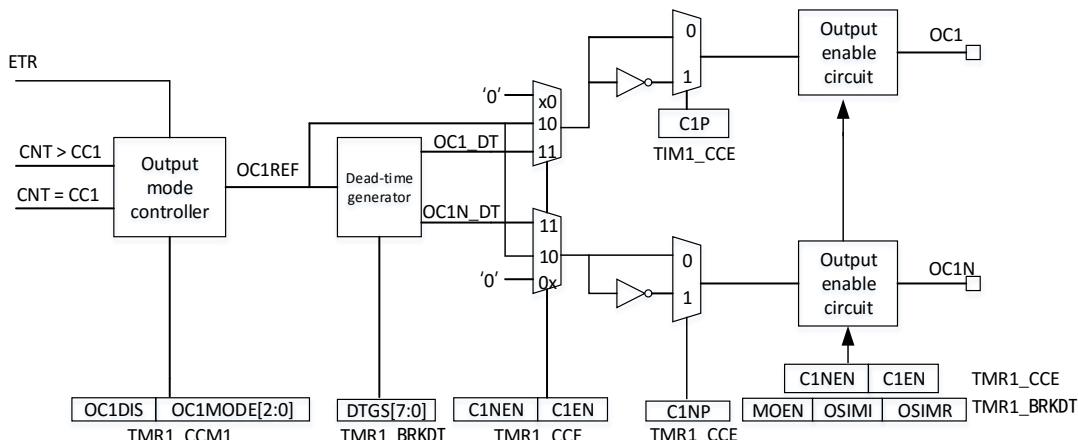
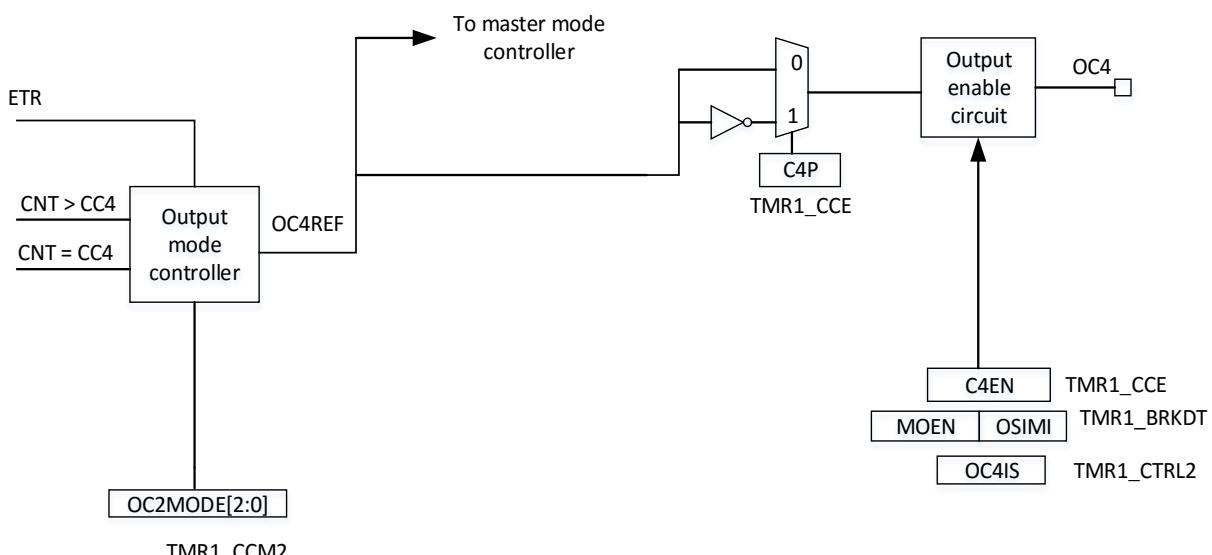


Figure 10-109 Capture/Compare Channel Output Stage (Channel 4)



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

In capture mode, captures are actually done in the shadow register, and then copied into the preload

register.

In compare mode, the content of the preload register is copied into the shadow register, and then the content of the shadow register is compared with the counter.

10.4.3.6 Input Capture Mode

In input capture mode, the capture/compare registers (TMRx_CC_x) latch the current counter value after a corresponding edge when the IC_x signal is detected. When a capture occurs, the corresponding CxIF flag (TMRx_STS register) is set. An interrupt or a DMA request can be sent if enabled. If a capture occurs when the CxIF flag is already high, the over-capture flag CxOF (TMRx_STS register) is set. CxIF can be cleared by writing CxIF = 0 or by reading the captured data stored in the TMRx_CC_x register. CxOF = 0 is cleared when writing CxOF = 0.

The following example shows how to capture the counter value to the TMRx_CC1 register during TI1 input rising edge:

- Select the active input: Write C1SEL = 01 in the TMRx_CC1 register since the TMRx_CC1 register must be linked to the TI1 input. The channel is configured as input as long as the C1SEL is not '00'. TMRx_CC1 register becomes read-only.
- Program the desired input filter duration according to the input signal (if the input is TI_x, the control bit of the input filter is the ICxDF bit in the TMRx_CCM_x register). If the input signal is not stable during five internal clock cycles at most, the filter duration should be configured longer than five clock cycles. Therefore, an actual edged transition on TI1 can be validated by 8 consecutive samples (at f_{CK_INT} frequency). That is, write IC1DF = 0011 in the TMRx_CCM1 register.
- Select the active transition edge on the TI1 channel by writing C1P = 0 (rising edge) in the TMRx_CCE register).
- Program the input prescaler. In this example, each valid transition is expected to be captured, so the prescaler is disabled (write IC1DIV = 00 in the TMRx_CCM1 register).
- Set C1EN = 1 in the TMRx_CCE register to allow the counter value to be captured into the capture register.
- If needed, enable the relevant interrupt request by setting the C1IE bit in the TMRx_DIE register, and the DMA request by setting the C1DE bit in the TMRx_DIE register.

When an input capture occurs:

- The counter value is sent to the TMRx_CC1 register on active transition.
- C1IF flag is set (interrupt flag). When at least two consecutive captures occur and the C1IF flag is not cleared, C1OF is also set.
- An interrupt is generated if the C1IE bit is set.
- A DMA request is generated if the C1DE bit is set.

In case of overcapture, it is recommended that the data is read before the overcapture flag. This is to avoid missing an overcapture information that is generated after reading the overcapture flag and before reading the data.

Note: Input capture interrupt and/or DMA requests can be generated by setting the corresponding CxG bit in the TMRx_EVEG register with software.

10.4.3.7 PWM Input Mode

This mode is a special case of input capture mode. The procedure is the same as input capture mode except for the following conditions:

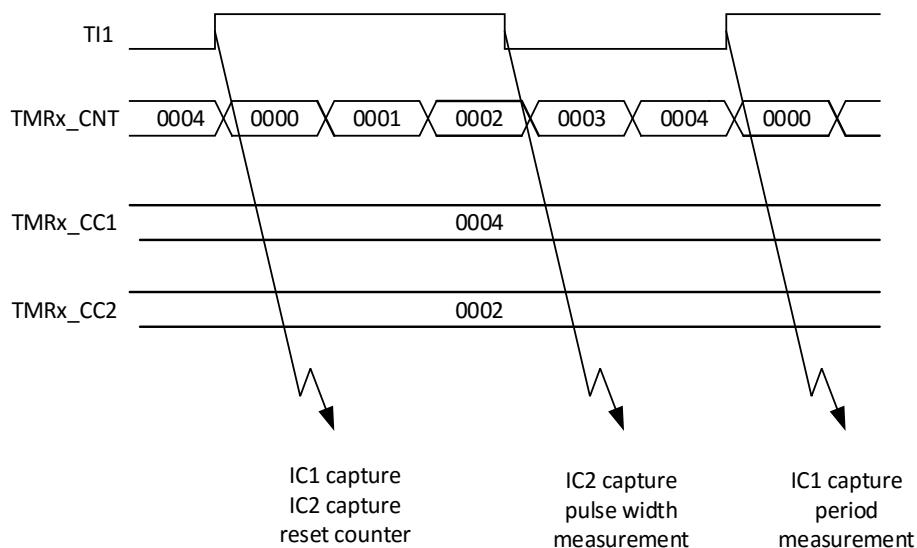
- Two IC_x signals are mapped on the same TI_x input.
- The two IC_x signals are active on edges but with opposite polarities.
- One of the two TI_xFP signals is used as trigger input signal and the slave mode controller is configured as reset mode.

For example, if users need to measure the period (TMRx_CC1 register) and the duty cycle (TMRx_CC2 register) of the PWM signal input on TI1, the procedure is as follows (depending on CK_INT frequency

and prescaler value):

- Select the active input for TMRx_CC1: Write C1SEL = 01 in the TMRx_CCM1 register (TI1 selected).
- Select the active polarity for TI1FP1 (used to capture data into TMRx_CC1 and clear the counter): Write C1P = 0 (active on rising edge).
- Select the active input for TMRx_CC2: Write C2SEL = 10 in the TMRx_CCM1 register (TI1 selected).
- Select the active polarity for TI1FP2 (capture data to TMRx_CC2): Write C2P = 1 (active on falling edge).
- Select the valid trigger input signal: Write TRGSEL = 101 in the TMRx_SMC register (TI1FP1 selected).
- Configure the slave mode controller as reset mode: Write SMSEL = 100 in the TMRx_SMC register.
- Enable the captures: Write C1EN = 1 and C2EN = 1 in the TMRx_CCE register.

Figure 10-110 PWM Input Mode Timing



Note: Since only TI1FP1 and TI2FP2 are connected to the slave mode controller, the PWM input mode can be used only with the TMRx_CH1/TMRx_CH2 signals.

10.4.3.8 Forced Output Mode

In output mode (CxSEL = 00 in the TMRx_CCMx register), output compare signal (OCxREF and the corresponding OCx/OCxN) can be forced to be active or inactive level directly by software, independent of any comparison between the output compare register and the counter.

To force an output compare signal (OCxREF/OCx) to its active level, users can write the corresponding OCxMODE = 101 in the TMRx_CCMx register. In this way, OCxREF is forced high (OCxREF is always active high), and at the same time OCx gets the opposite value of the CxP polarity bit.

For example: CxP = 0 (OCx active high), then OCx is forced to be high level.

The OCxREF signal can be forced low by writing OCxMODE = 100 in the TMRx_CCx register.

In this mode, the comparison between the TMRx_CCx shadow register and the counter is still ongoing, and the corresponding flag is also modified. Accordingly, the corresponding interrupt and DMA requests are still generated. This will be described in [Section 10.4.3.9](#).

10.4.3.9 Output Compare Mode

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

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

- Output the value defined by the output compare mode (the OCxMODE bit in the

TMRx_CCMx register) and the output polarity (the CxP bit in the TMRx_CCE register) to the corresponding pin. When being compared and matched, the output pin can keep its level (OCxMODE = 000), be set active (OCxMODE = 001), be set inactive (OCxMODE = 010), or toggle (OCxMODE = 011).

- Set the flag bit in the interrupt status register (the CxIF bit in the TMRx_STS register).
- An interrupt will be generated if the corresponding interrupt mask is set (the CxIE bit in the TMRx_DIE register).
- A DMA request will be generated if the corresponding enable bit is set (the CxDE bit in the TMRx_DIE register, the CDSEL bit in the TMRx_CTRL2 register for the DMA request selection).

The OCxPEN bit in the TMRx_CCMx register can be configured to choose whether the TMRx_CCx register uses the preload register.

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

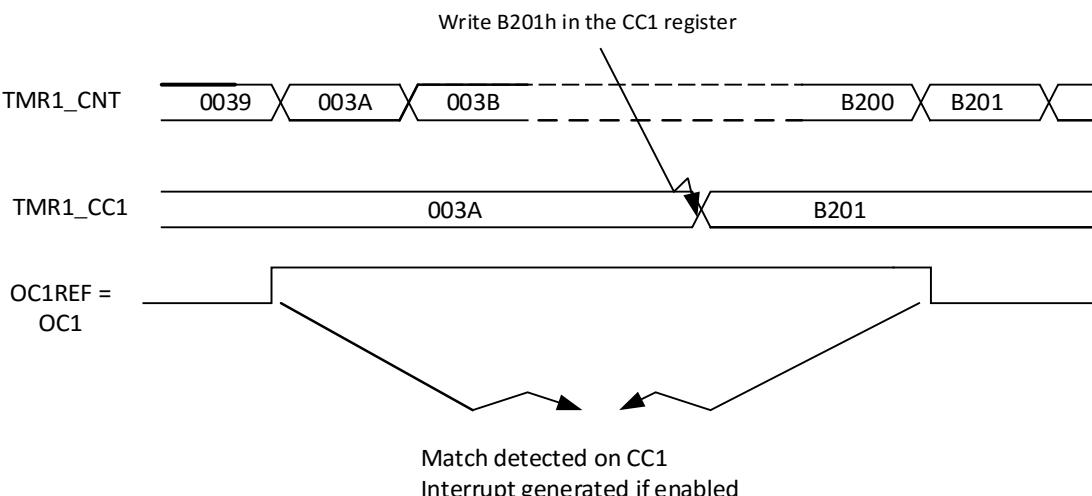
The timing resolution can reach one count of the counter. Output compare mode can also be used to output a single pulse (in one-pulse mode).

Output compare mode configuration procedure is as follows:

1. Select the counter clock (internal, external, and prescaler).
2. Write the corresponding data to the TMRx_AR and the TMRx_CCx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
 - When the counter and the CCx bit is matched, toggle the output pin of OCx, configure OCxMODE = '011'
 - Set OCxPEN = 0, disable the preload register
 - Set CxP = 0, select polarity as active high
 - Set CxEN = 1, enable output
5. Enable the counter by setting the CNTEN bit in the TMRx_CTRL1 register.

The TMRx_CCx register can be updated at any time by software to control the output waveform, under the condition that the preload register is not enabled (OCxPEN = '0', else the TMRx_CCx shadow register can only be updated at the next update event). An example is shown in Figure 10-111.

Figure 10-111 Output Compare Mode, Toggle on OC1



10.4.3.10 PWM Mode

Pulse width modulation mode can generate a signal with its frequency determined by the TMRx_AR register and its duty cycle determined by TMRx_CCx register.

Writing '110' (PWM mode 1) or '111' (PWM mode 2) in the OCxMODE bit in the TMRx_CCMx register can independently configure each OCx output channel and generate single-channel PWM. The

OCxPEN bit in the TMRx_CCMx register must be configured to enable the corresponding preload register. Finally, the ARPEN bit in the TMRx_CTRL1 register (in upcounting or center-aligned modes) must be set to enable the auto-reload preload register.

Since the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, users must initialize all the registers by setting the UEVG bit in the TMRx_EVEG register. OCx polarity is software programmable by setting the CxP bit in the TMRx_CCE register. It can be programmed as active high or active low. OCx output enable is controlled by the combination of the CxEN, CxNEN, MOEN, OSIMI, and OSIMR bits (in the TMRx_CCE and TMRx_BRKDT registers). Please refer to the description of TMRx_CCE register for more details.

In PWM mode (1 or 2), TMRx_CNT and TMRx_CCx are always compared (according to the direction of the counter) to confirm that $TMRx_CCx \leq TMRx_CNT$ or $TMRx_CNT \leq TMRx_CCx$.

The timer can generate the edge-aligned or center-aligned PWM signal according to the status of the CMSEL bit in the TMRx_CTRL1 register.

PWM edge-aligned mode

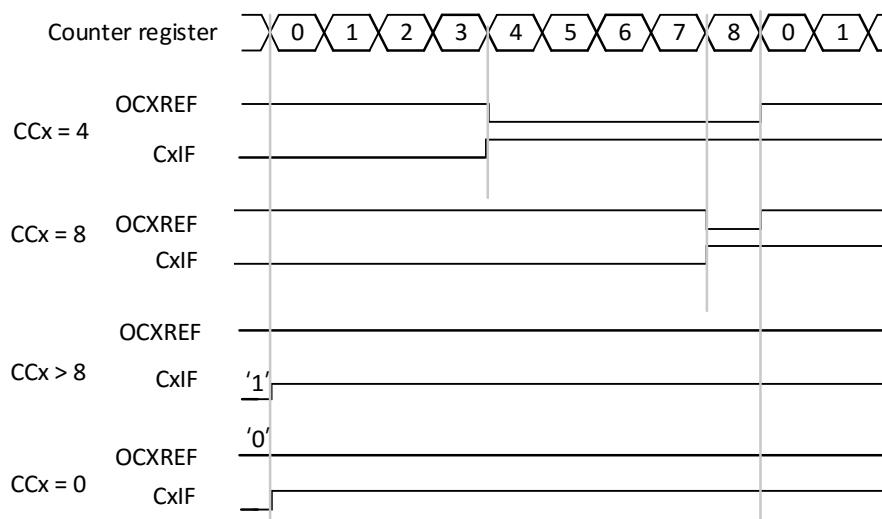
- Upcounting configuration

Upcounting is active when the DIR bit in the TMRx_CTRL1 register is low. Please refer to [Section 10.4.3.2](#).

The following is an example of PWM mode 1. The reference PWM signal, OCxREF, is high once $TMRx_CNT < TMRx_CCx$; else, it becomes low. If the compare value in TMRx_CCx is greater than the auto-reload value (in TMRx_AR), OCxREF is held at '1'.

If the compare value is 0, then OCxREF is held at '0'. Figure 10-112 shows an example of edge-aligned PWM waveforms with TMRx_AR = 8.

Figure 10-112 Edge-aligned PWM Waveforms (AR = 8)



- Downcounting configuration

Downcounting is active when the DIR bit in the TMRx_CTRL1 register is high. Please refer to [Section 10.4.3.2](#). In PWM mode 1, the reference signal, OCxREF, is low once $TMRx_CNT > TMRx_CCx$; else, it becomes high. If the compare value in TMRx_CCx is greater than the auto-reload value in TMRx_AR, then OCxREF is held at '1'. 0% PWM is not possible in this mode.

PWM center-aligned mode

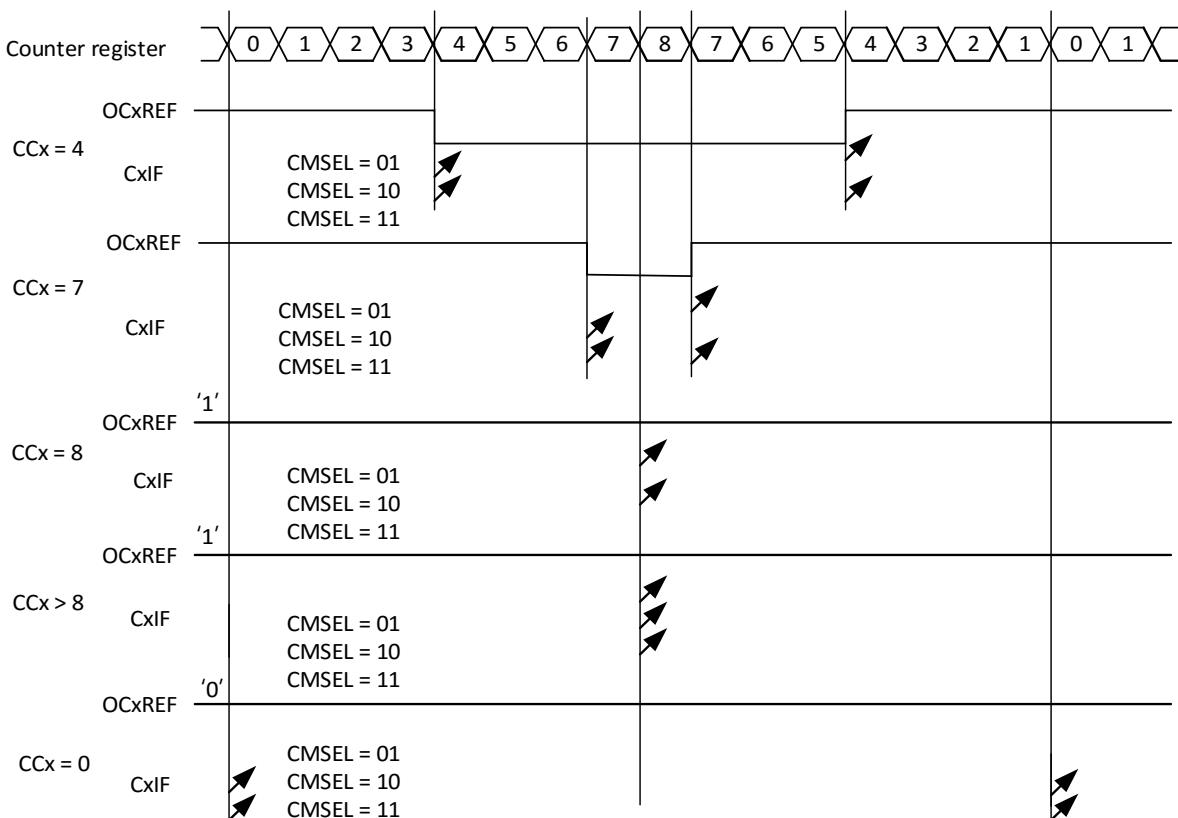
Center-aligned mode is active when the CMSEL bit in the TMRx_CTRL1 register is not '00' (all the other configurations have the same effect on the OCxREF/OCx signals). According to different CMSEL bit configurations, the compare flag is set when the counter counts up, counts down, or counts both up and down. The direction bit (DIR) in the TMRx_CTRL1 register is updated by hardware and must not be changed by software. Please refer to the center-aligned mode in [Section 10.4.3.2](#).

Figure 10-113 shows several center-aligned PWM waveforms in an example where:

- TMRx_AR = 8
- PWM mode 1
- The compare flag is set when the counter counts down, center-aligned mode 1 is

selected, and CMSEL = 01 in the TMRx_CTRL1 register.

Figure 10-113 Center-aligned PWM Waveforms (AR = 8)



Hints on using center-aligned mode:

- When entering the center-aligned mode, the current up/down configuration is used. This means whether the counter counts up or down depends on the current value of the DIR bit in the TMRx_CTRL1 register. In addition, the DIR and CMSEL bits cannot be changed by software at the same time.
- In center-aligned mode, writing to the counter while running is not recommended as it may lead to unexpected results. In particular:
 - If the value written to the counter is greater than the auto-reload value (TMRx_CNT > TMRx_AR), the direction will not be updated. For example, if the counter is counting up, it will keep counting up.
 - The direction is updated if 0 or the TMRx_AR value is written to the counter, but no update event UEV will be generated.

The safest way to use the center-aligned mode is to generate an update by software (setting the UEVG bit in the TMRx_EVEG register) before enabling the counter, and do not write the counter while it is running.

10.4.3.11 Complementary Output and Dead-time Insertion

The advanced-control timers (TMR1, TMR8, and TMR15) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs. This time is generally known as dead-time. Users can adjust dead-time according to the devices connected to the outputs and their characteristics (intrinsic delays of level-shifters, delays resulted from power switches, etc.)

Setting the CxP and CxNP bits in the TMRx_CCE register can select the polarity (main output OCx or complementary OCxN) for each output independently.

The complementary signals, OCx and OCxN, are activated by the combination of the following control bits: CxEN and CxNEN bits in the TMRx_CCE register, the MOEN, OCxIS, OCxNIS, OSIMI, and OSIMR bits in the TMRx_BRKDT and TMRx_CTRL 2 register. Please refer to [Table 10-14](#) for more details on output control bits for complementary OCx and OCxN channels with break feature. In particular, the

dead-time is activated when switching to the IDLE state (MOEN falling down to 0).

Setting CxEN and CxNEN bits at the same time will insert dead-time. The MOEN bit should also be set if there is break circuit. Every channel has a 10-bit dead-time generator. Reference signal, OCxREF, can generate two outputs, OCx and OCxN. If OCx and OCxN are active high:

- The OCx output signal is the same as the reference signal, but its rising edge is delayed compared to the reference signal rising edge.
- The OCxN output signal is the opposite of the reference signal, but its rising edge is delayed compared to the reference signal falling edge.

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

Figure 10-114 ~ Figure 10-116 show the relationship between the output signals of the dead-time generator and the reference signal, OCxREF. (Assuming that CxP = 0, CxNP = 0, MOEN = 1, CxEN = 1, and CxNEN = 1)

Figure 10-114 Complementary Output with Dead-time Insertion

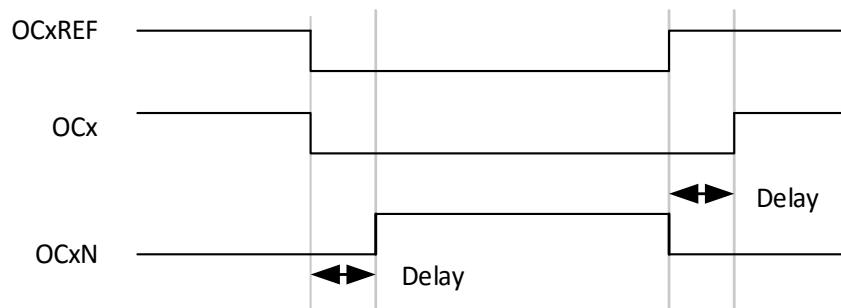


Figure 10-115 Dead-time Waveform Delay Which is Greater than Negative Pulse

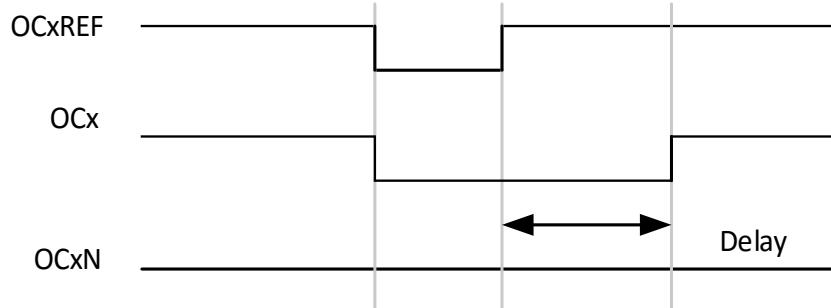
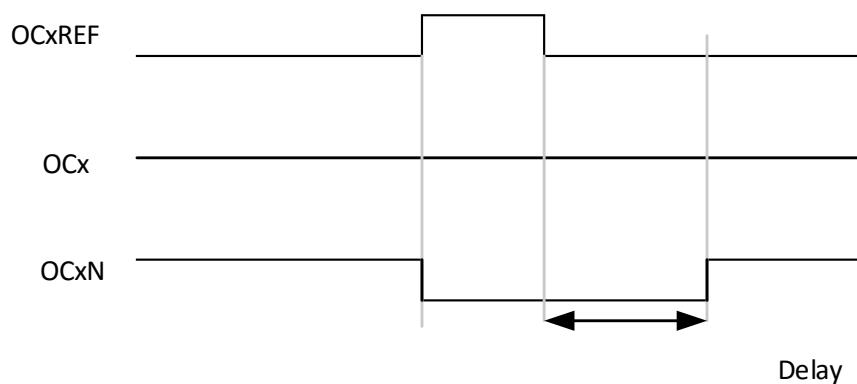


Figure 10-116 Dead-time Waveform Delay Which is Greater than Positive Pulse



The dead-time delay is the same for each channel, and is programmable with the DTGS bits in the TMRx_BRKDT register. Please refer to [Section 10.4.4.18](#) for delay calculation.

Re-directing OCxREF to OCx or OCxN

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

This function allows the user to send a specific waveform (such as PWM or static active level) on one output while the complementary output remains at its inactive level. Another usage is to put both outputs at inactive level, or at active level to be complementary with dead-time.

Note: When only OCxN is enabled ($CxEN = 0$, $CxNEN = 1$), it is not inverted and becomes active as soon as OCxREF is high. For example, if $CxNP = 0$, then $OCxN = OCxRef$. On the other hand, when both OCx and OCxN are enabled ($CxEN = CxNEN = 1$), OCx becomes active when OCxREF is high, whereas OCxN becomes active when OCxREF is low.

10.4.3.12 Using the Break Function

When using the break function, the output enable signals and inactive levels are modified according to the corresponding control bits (the MOEN, OSIMI, and OSIMR bits in the TMRx_BRKDT register, the OCxIS and OCxNIS bits in the TMRx_CTRL2 register). In any case, OCx and OCxN outputs cannot both be set to active level at the same time. Please refer to [Table 10-14](#) for more details.

The break source can be the break input pin or a clock failure event. A clock failure event is generated by the clock security system in the reset clock controller. For further information, please refer to [Section 3.2.7](#).

After system reset, the break circuit is disabled, and the MOEN bit is low. Users can enable the break function by setting the BRKEN bit in the TMRx_BRKDT register. The break input polarity can be selected by configuring the BRKP bit in the same register. BRKEN and BRKP can be modified at the same time. When the BRKEN and BRKP bits are written, a delay of 1 APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait for 1 APB clock period to correctly read the written bit.

Because MOEN falling edge can be asynchronous, a resynchronization circuit is inserted between the actual signal (acting on the outputs) and the synchronous control bit (in the TMRx_BRKDT register). The resynchronization circuit generates delays between the asynchronous and the synchronous signals. In particular, if $MOEN = 1$ when it is low, a delay (dummy instruction) must be inserted before reading it. This is because an asynchronous signal is written, but a synchronous signal is read.

When a break occurs (selected level occurs on the break input), there are following actions:

- The MOEN bit is cleared asynchronously, putting the outputs at inactive state, idle state, or at reset state (selected by the OSIMI bit). This feature functions even if the MCU oscillator is off.
- Once $MOEN = 0$, each output channel is driven with the level programmed in the OCxIS bit in the TMRx_CTRL2 register. If $OSIMI = 0$, the timer releases the enable output, else the enable output remains high.
- When complementary outputs are used:
 - The outputs are first put at reset state, namely, inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
 - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OCxIS and OCxNIS bits after a dead-time. Even in this case, OCx and OCxN cannot be driven to their active levels at the same time.
Please note that because of the resynchronization on MOEN, the dead-time duration is a bit longer than usual (around 2 ck_tim clock cycles).
 - If $OSIMI = 0$, the timer releases the enable output, else keeps the enable output. Or the enable output becomes high as soon as one of the CxEN and CxNEN bits become high.
- If the BRKIE bit in the TMRx_DIE register is set, an interrupt can be generated when the break status flag (the BRKIF bit in the TMRx_STS register) is set. A DMA request can be sent if the BDE bit in the TMRx_DIE register is set.
- If the AOEN bit in the TMRx_BRKDT register is set, the MOEN bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Otherwise, MOEN remains low until it is set again. This feature can be used for security purposes. The break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

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

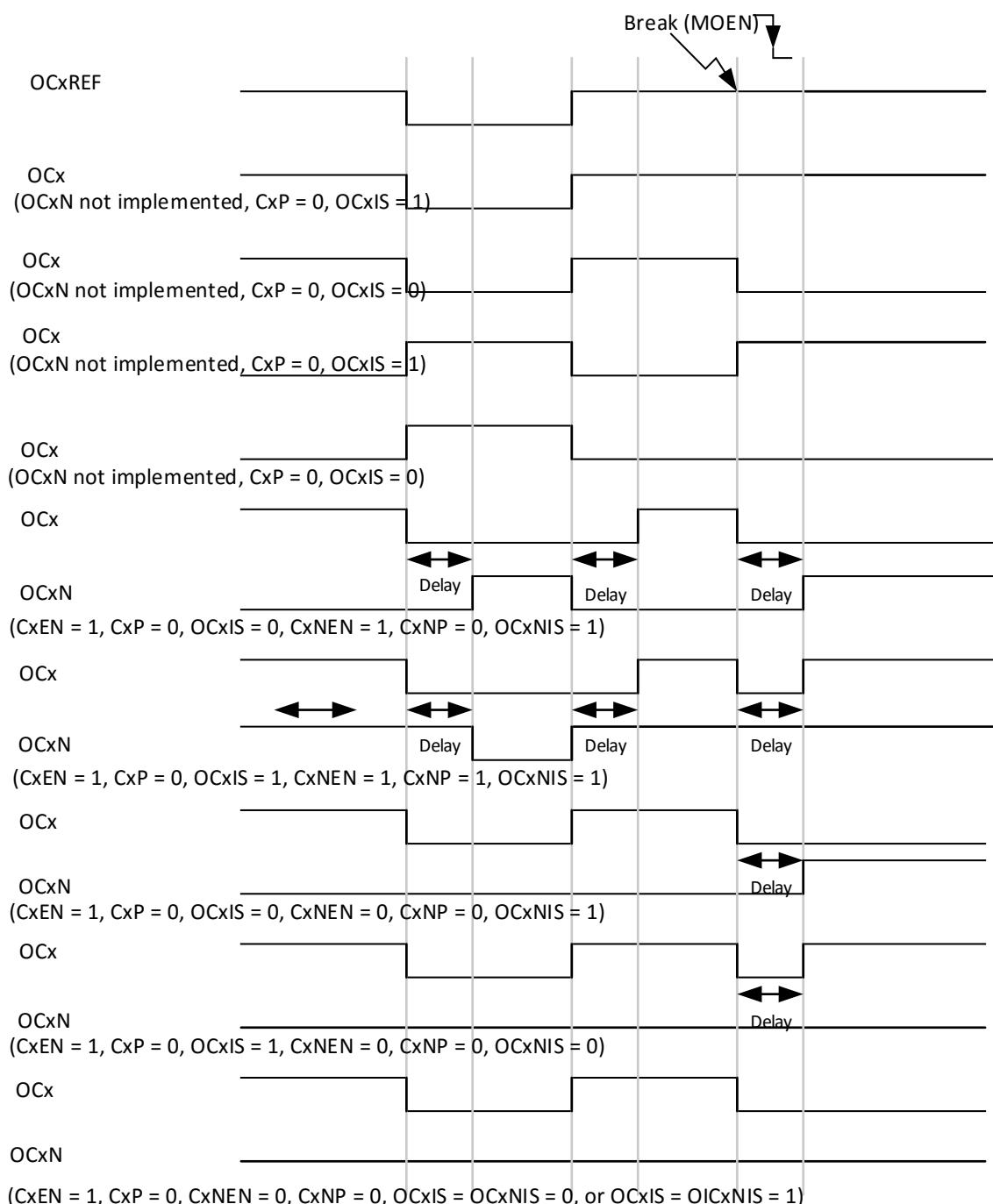
The break is generated by the BRK input which has a programmable polarity, and is enabled by the BRKEN bit in the TMRx_BRKDT register.

In addition to break input and output management, a write protection is implemented inside the break circuit to safeguard the application. It allows freezing the configuration of several parameters (dead-time duration, OCx/OCxN polarities and state when disabled, OCxMODE configurations, break enable and polarity).

Users can choose from three levels of protection through the LOCKC bits in the TMRx_BRKDT register. Please refer to [Section 10.4.4.18](#). The LOCKC bits can be written only once after an MCU reset.

Figure 10-117 shows an example of the output behavior in response to a break.

Figure 10-117 Outputs in Response to a Break



10.4.3.13 Clearing OCxREF Signal on an External Event

For a given channel, a high level of the ETRF input can lower the OCxREF signal if the corresponding OCxDIS bit in the TMRx_CCMx register is set. The OCxREF signal remains low until the next update event, UEV, occurs.

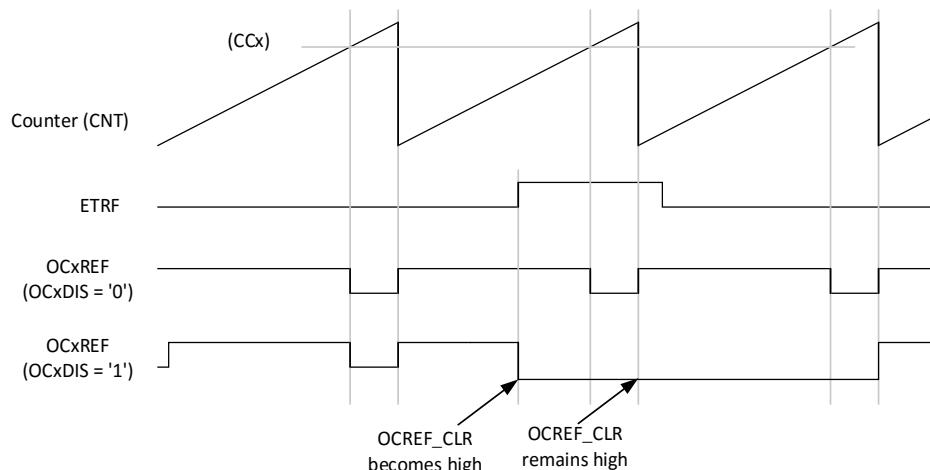
This function can only be used in output compare and PWM modes, and does not work in the forced output mode.

For example, the OCxREF signal can be connected to a comparator output for current control. In this case, ETR must be configured as follows:

1. The external trigger prescaler should be kept OFF: ETD[1:0] = '00' in the TMRx_SMC register.
2. The external clock mode 2 must be disabled: ECLKEN = '0' in the TMRx_SMC register.
3. The external trigger polarity (ETRGP) and the external trigger filter (ETDF) can be configured according to users' needs.

Figure 10-118 shows how OCxREF signal behaves to respond to different OCxDIS values when the ETRF input becomes high. In this example, the timer TMRx is programmed in PWM mode.

Figure 10-118 Clearing OCxREF in TMRx



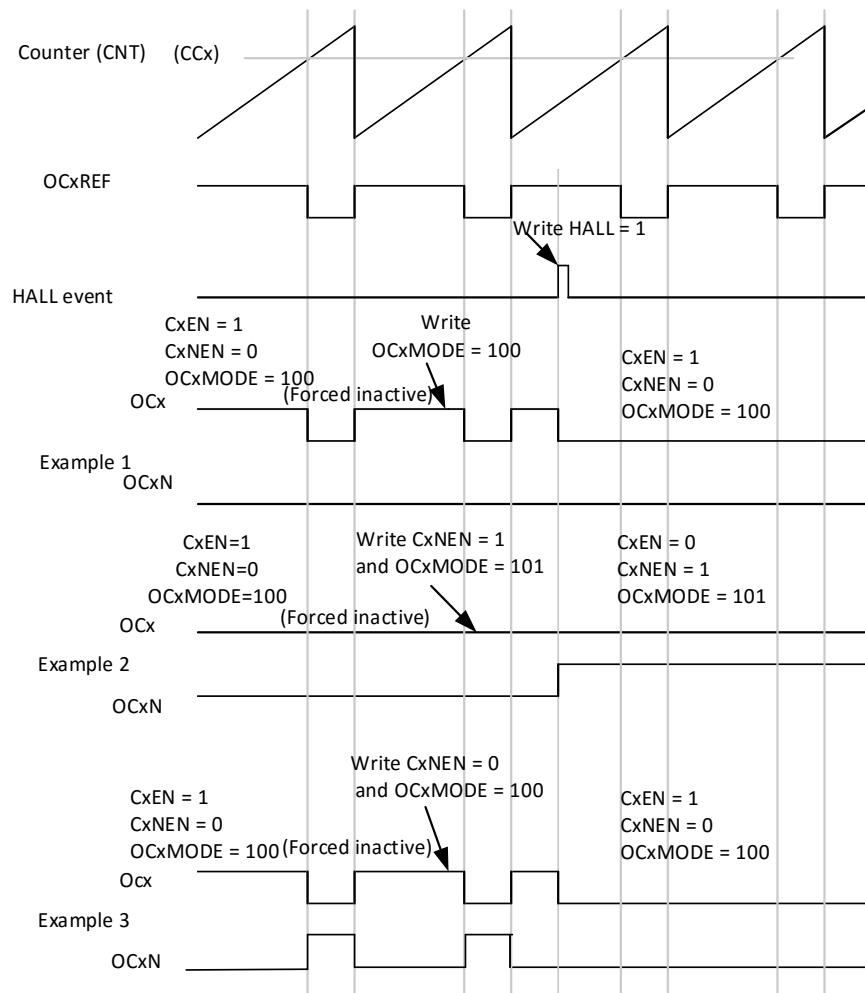
10.4.3.14 6-step PWM Output Generation

When complementary outputs are used on a channel, preload bits are the OCxMODE, CxEN, and CxNEN bits. The preload bits are transferred to the shadow bits at the HALL commutation event. In this way, users can program the configuration for the next step in advance and change the configuration of all the channels at the same time. HALL can be generated by setting the HALL bit in the TMRx_EVEG register through software or on TRGI rising edge by hardware.

A flag (the HALLIF bit in the TMRx_STS register) is set when the HALL event occurs. If the HALLIE bit in the TMRx_DIE register is set at this time, an interrupt will be generated. If the HALLDEB bit is set in the TMRx_DIE register, a DMA request will be generated.

Figure 10-119 shows the behavior of the OCx and OCxN outputs in 3 different configurations when a HALL event occurs.

Figure 10-119 6-step PWM Generation, Example of Using HALL (OSIMR = 1)



10.4.3.15 One-pulse Mode

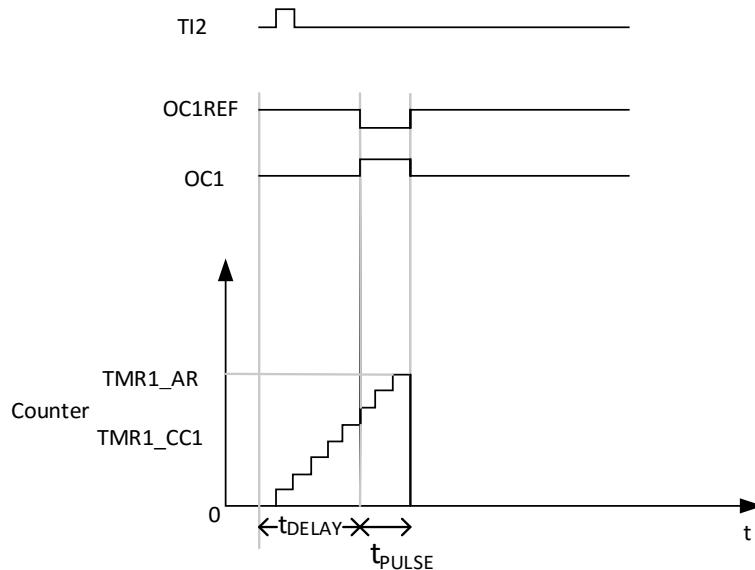
One-pulse mode (OPMODE) is quite different from the above-mentioned modes. It allows the counter to respond to a stimulus and to generate a pulse with a programmable length after a programmable delay.

The counter can be enabled through the slave mode controller, generating waveforms in output compare mode or PWM mode. One-pulse mode is selected by setting the OPMODE bit in the TMRx_CTRL1 register, and this can stop the counter automatically at the next update event UEV.

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

- In upcounting: $CNT < CCx \leq AR$ (especially, $0 < CCx$)
- In downcounting: $CNT > CCx$

Figure 10-120 One-pulse Mode Example



For example, if users want to generate a positive pulse with a length of t_{PULSE} on OC1 once a rising edge is detected on the TI2 input pin and after a delay t_{DELAY} .

If TI2FP2 is used as trigger 1:

- Map TI2FP2 to TI2 by writing C2SEL = '01' in the TMRx_CCM1 register.
- Write C2P = '0' in the TMRx_CCE register to enable TI2FP2 to detect a rising edge.
- Write TRGSEL = '110' in the TMRx_SMC register to make TI2FP2 the trigger of the slave mode controller (TRGI).
- Write SMSEL = '110' in the TMRx_SMC register (trigger mode), and TI2FP2 is used to enable the counter.

The OPMODE waveform is defined by the value written to the compare registers (the clock frequency and the counter prescaler should be taken into account).

- t_{DELAY} is defined by the value written to the TMRx_CC1 register.
- t_{PULSE} is defined by the comparison result of auto-reload value and the compare value (TMRx_AR - TMRx_CC1).
- Assume that users want to generate a waveform with a transition from '0' to '1' when a compare match occurs, and a transition from '1' to '0' when the counter reaches the preload value. First, set OC1MODE = '111' in the TMRx_CCM1 register to enter PWM mode 2. Enable the desired preload registers by setting OC1PEN = '1' in the TMRx_CCM1 register and the ARPEN bit in the TMRx_CTRL1 register. Afterwards, write compare value in the TMRx_CC1 register, write auto-reload value in the TMRx_AR register, modify the UEVG bit to generate an update event, and wait for an external trigger event on TI2. In this example, C1P = '0'.

In this example, the DIR and CMSEL bits in the TMRx_CTRL1 register should be low. Since only one pulse is needed, '1' must be written to the OPMODE bit in the TMRx_CTRL1 register to stop the counter at the next update event (when the counter rolls over to 0 from the auto-reload value).

Special case: OCx fast enable

In one-pulse mode, the counter is enabled by setting the CNTEN bit on the edge detection logic of TIx input pin. Then, the comparison between the counter and the compare value makes output toggle. However, since several clock cycles are needed for these operations, it limits the minimum delay t_{DELAY} users can get.

The OCxFEN bit in the TMRx_CCMx register can be set to output a waveform with the minimum delay. In this case, OCxREF (and OCx) are forced to respond to the stimulus, instead of relying on the comparison result. The waveform outputted is the same as the one being compared and matched. OCxFEN acts only if the channel is configured as PWM1 or PWM2 mode.

10.4.3.16 Encoder Interface Mode

To select the encoder interface mode, write SMSEL = 001 in the TMRx_SMC register if the counter is counting on TI2 edges only; write SMSEL = 010 if it is counting on TI1 edges only; write SMSEL = 011 if it is counting on both TI1 and TI2 edges.

Select the TI1 and TI2 polarity by programming the C1P and C2P bits in the TMRx_CCE register. Users can program the input filter as well when it is needed.

The two inputs, TI1 and TI2, are used as the interface of incremental encoder. Please refer to Table 10-11. If the counter is enabled (CNTEN = '1' in the TMRx_CTRL1 register), it is clocked by each valid transition on TI1FP1 or TI2FP2. TI1FP1 and TI2FP2 are the signals of TI1 and TI2 after input filter and polarity selection. If not filtered and not inverted, TI1FP1 = TI1 and TI2FP2 = TI2. Count pulses and direction signal are generated according to the transition sequence of the two inputs. The counter counts up or down according to the transition sequence of the two inputs, and at the same time the DIR bit in the TMRx_CTRL1 register is set accordingly by hardware.

The DIR bit is re-calculated at each transition on any input (TI1 or TI2), no matter the counter is counting on TI1 only, TI2 only, or both TI1 and TI2.

Encoder interface mode basically acts like an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value in the TMRx_AR register (0 to AR or ARR down to 0, depending on the direction). Hence, the TMRx_AR register must be configured before the counting starts. Likewise, features of the capture, comparator, prescaler, repetition counter, and trigger output work as usual. Encoder mode and external clock mode 2 are not compatible, so they cannot be used at the same time.

In this mode, the counter is modified automatically according to the speed and the direction of the incremental encoder. Therefore, its content always indicates the position of the encoder. The counting direction corresponds to the rotation direction of the connected sensor. Table 10-11 lists all the possible combinations, assuming that TI1 and TI2 do not switch at the same time.

Table 10-11 Counting Direction and Encoder Signals

Active Edge	Level on Opposite Signal (TI1FP1 to TI2, TI2FP2 to TI1)	TI1FP1 Signal		TI2FP2 Signal	
		Rising	Falling	Rising	Falling
Counting on TI1 only	High	Down	Up	No count	No count
	Low	Up	Down	No count	No count
Counting on TI2 only	High	No count	No count	Up	Down
	Low	No count	No count	Down	Up
Counting on TI1 and TI2	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

An external incremental encoder can be connected directly to MCU without external interface logic. However, comparators are normally used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output signal, which indicates the mechanical zero position, can be connected to an external interrupt input to trigger a counter reset.

Figure 10-121 provides an example of counter operation, showing counting signal generation and direction control. It also shows how input jitter is compensated when both edges are selected. Jitters might occur if the sensor is positioned near to one of the switching points. For example, assume that the configuration is the following:

- C1SEL = '01' (TMRx_CCM1 register, IC1FP1 is mapped on TI1.)
- C2SEL = '01' (TMRx_CCM1 register, IC2FP2 is mapped on TI2.)
- C1P = '0' (TMRx_CCE register, IC1FP1 not inverted, IC1FP1 = TI1)
- C2P = '0' (TMRx_CCE register, IC2FP2 not inverted, IC2FP2 = TI2)
- SMSEL = '011' (TMRx_SMC register, all inputs are active on both rising and falling edges.)
- CNTEN = '1' (TMRx_CTRL1 register, counter enabled)

Figure 10-121 Example of Counter Operation in Encoder Interface Mode

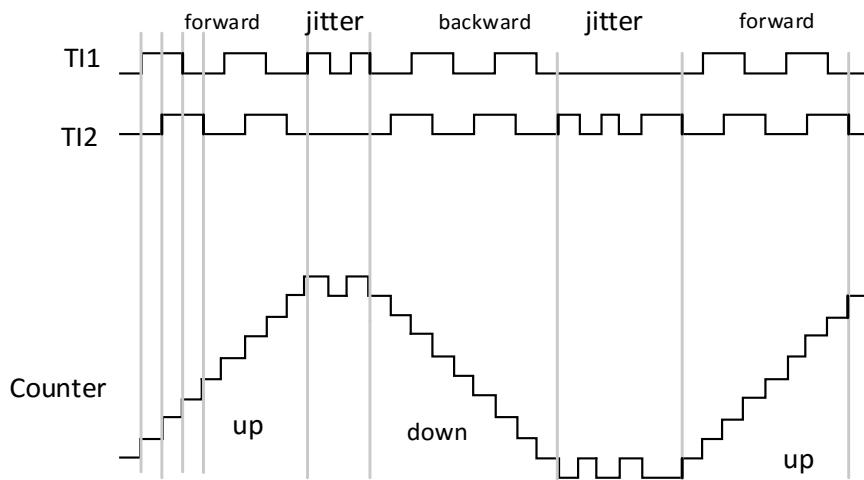
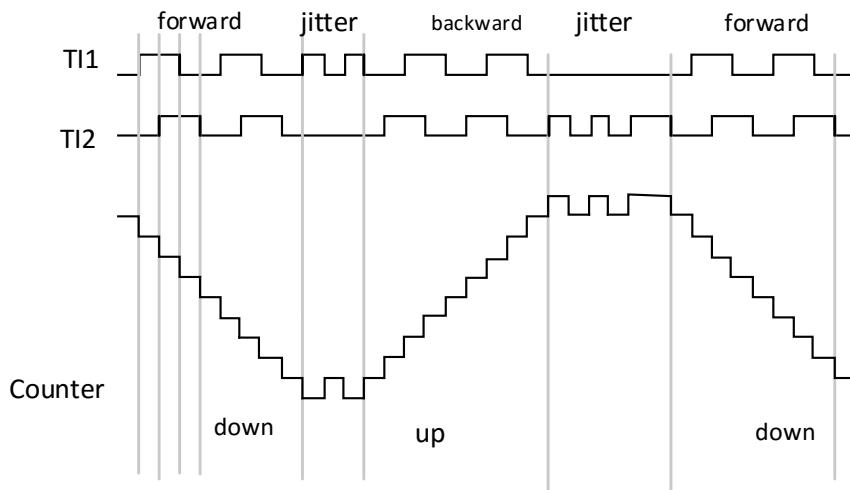


Figure 10-122 is an example of counter behavior when IC1FP1 polarity is inverted (the same configuration as the above example, except for C1P = '1').

Figure 10-122 Example of Encoder Interface Mode with IC1FP1 Polarity Inverted



The timer, when configured as encoder interface mode, provides information on the sensor's current position. Using the second timer configured in capture mode can measure the interval period between two encoder events to obtain the dynamic information (speed, acceleration, and deceleration). The encoder output which indicates the mechanical zero can be used for this purpose. Based on the interval between two events, the counter can also be read regularly. If possible, users can latch the counter value into the third input capture register (the capture signal must be periodic and can be generated by another timer). It is also possible to read its value through a DMA request generated by a real-time clock.

10.4.3.17 Timer Input XOR Function

The TI1SEL bit in the TMRx_CTRL2 register allows the input filter of channel 1 to be connected to the output of an XOR gate, which includes three input pins, TMRx_CH1, TMRx_CH2, and TMRx_CH3.

The XOR output can be used with all the timer input functions, such as trigger or input capture. An example of this feature used to interface Hall sensors is listed in the next section.

10.4.3.18 Interfacing with Hall Sensors

When using the advanced-control timer (TMR1, TMR8, or TMR15) to generate PWM signal which drives the motor, another timer general-purpose timer TMRx (TMR2, TMR3, TMR4, or TMR5) can function as “interfacing timer” to connect Hall sensors. Please refer to [Figure 10-123](#). The 3 timer input pins (CC1, CC2, and CC3) are connected to the TI1 input channel (selected by setting the TI1SEL bit in the TMRx_CTRL2 register) through an XOR gate, and the “interfacing timer” captures this signal.

The slave mode controller is configured in reset mode; the slave input is TI1F_ED. Each time when one

of the 3 inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TRC (see [Figure 10-106](#)). The captured value, which reflects the time delay between the 2 changes on the inputs, provides information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which can change the channel configuration of the advanced-control timer TMR1, TMR8, or TMR15 (by triggering a HALL event). The PWM signal generated by advanced-control timers can drive the motor.

Therefore, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer (TIM1, TIM8, or TMR15) through the TRGO output.

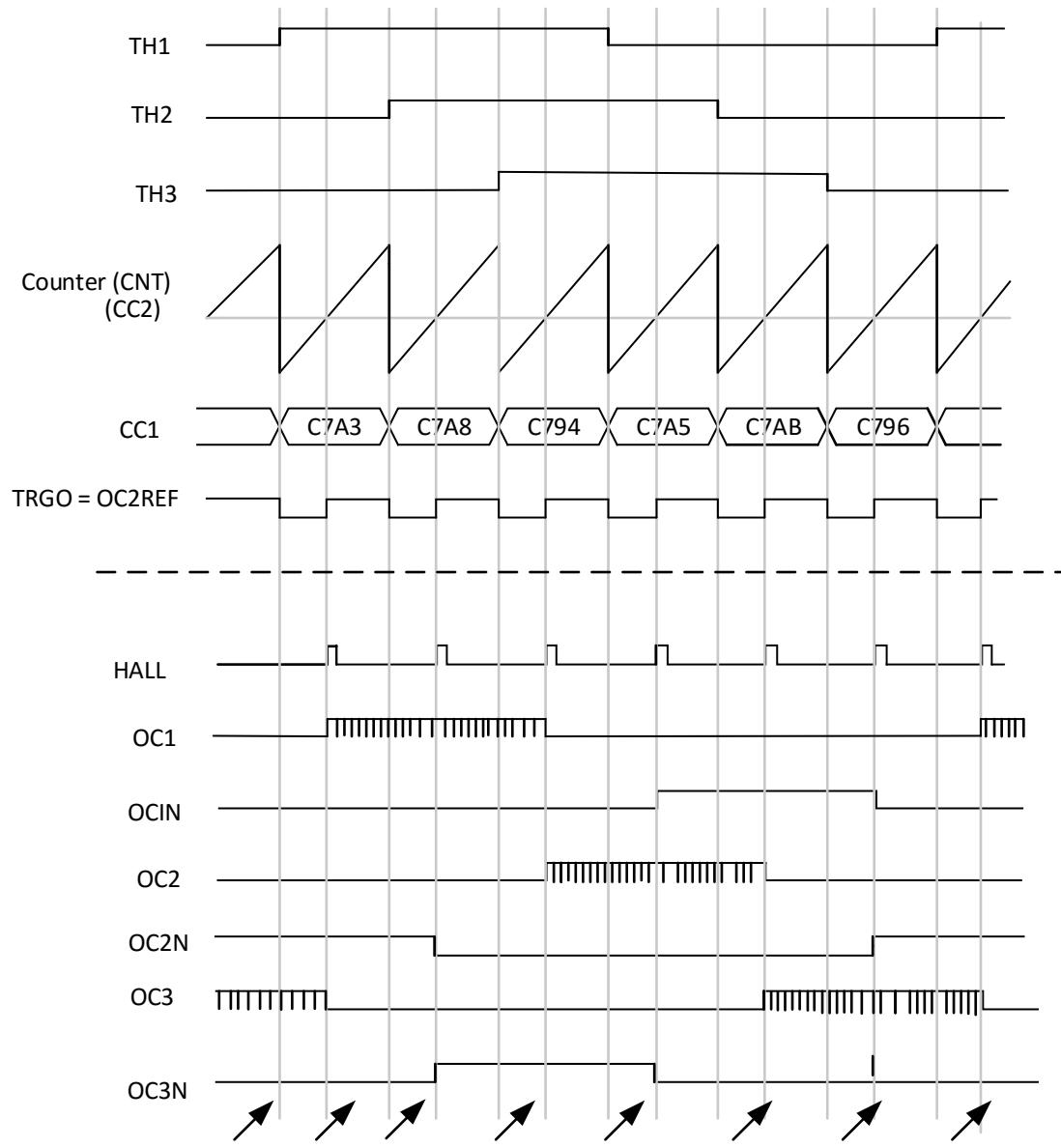
For example: To change the PWM configuration of the advanced-control timer TMRx at a given time after any change occurs on the Hall inputs connected to the TMRx timer:

- Configure 3 timer inputs ORed to the TI1 input by writing the TI1SEL bit in the TMRx_CTRL2 register to ‘1’.
- Program the time base: Set the TMRx_AR to the max. value (the counter must be cleared by the TI1 change.) Set the prescaler to get a maximum counter period longer than the time between 2 changes on the sensors.
- Program channel 1 in capture mode (TRC selected): Write C1SEL = 11 in the TMRx_CCM1 register. The digital filter can also be configured if needed.
- Program channel 2 in PWM 2 mode with the desired delay: Write OC2MODE = 111 and C2SEL = 00 in the TMRx_CCM1register.
- Select OC2REF as trigger output on TRGO: Write MMSEL = 101 in the TMRx_CTRL2 register.

In the advanced-control register TMR1, the correct ITR input must be trigger input. The timer is programmed to generate PWM signals, and the capture/compare control signals are preloaded (CPC = 1 in the TMRx_CTRL2 register), and at the same time the HALL event is controlled by the trigger input (CUSEL = 1 in the TMRx_CTRL2 register). The PWM control bits (CxEN, OCxMODE) for the next step are written after a HALL event. This can be done in an interrupt subroutine generated by the rising edge of OC2REF).

[Figure 10-123](#) shows the example.

Figure 10-123 Example of HALL Sensor Interface



Write CxEN, CxNEN, and OCxMODE for the next step

10.4.3.19 TMRx Timer and External Trigger Synchronization

The TMRx timer can be synchronized with an external trigger in several modes: reset mode, gated mode, and trigger mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized when a trigger input event occurs. In this case, if the UEVDIS bit of the TMRx_CTRL1 register is low, an update event UEV is generated. Then, all the preloaded registers (TMRx_AR, TMRx_CCx) will be updated.

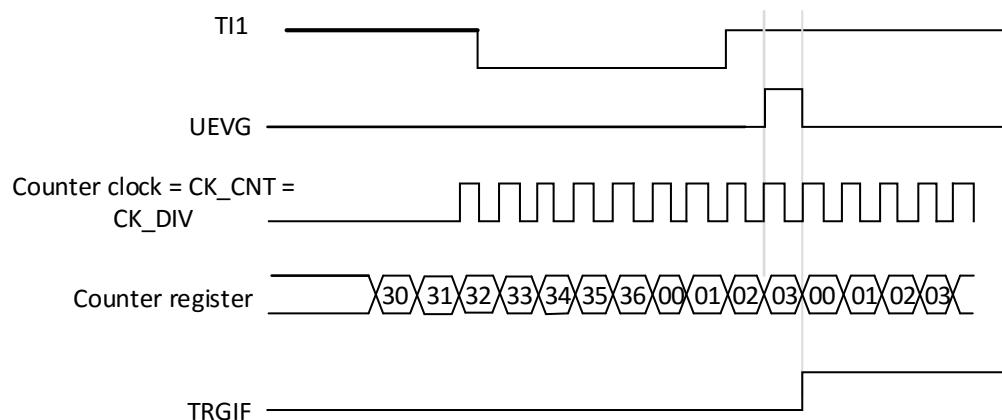
In the following example, the up counter is cleared due to a rising edge on TI1 input:

- Configure channel 1 to detect the rising edges on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bit selects the input capture source only, that is, C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 0 in the TMRx_CCE register to confirm the polarity (and detect rising edges only).

- Configure the timer as reset mode by writing SMSEL = 100 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Start the counter by writing CNTEN = 1 in the TMRx_CTRL1 register.

The counter starts counting on the internal clock, and runs normally until TI1 rising edge occurs; at this time, the counter is cleared and restarts from 0. In the meantime, the trigger flag (the TRGIF bit in the TMRx_STS register) is set, and an interrupt request, or a DMA request can be sent depending on the configuration of the TRGIE bit in the TMRx_DIE register (interrupt enable) and the TRGDE bits in TMRx_DIE register (DMA enable). Figure 10-124 shows the behavior when the auto-reload register TMRx_AR = 0x36. The delay between the rising edge on TI1 and the actual reset of the counter is determined by the resynchronization circuit on TI1 input.

Figure 10-124 Control Circuit in Reset Mode



Slave mode: Gated mode

The counter can be enabled according to the selected level of input.

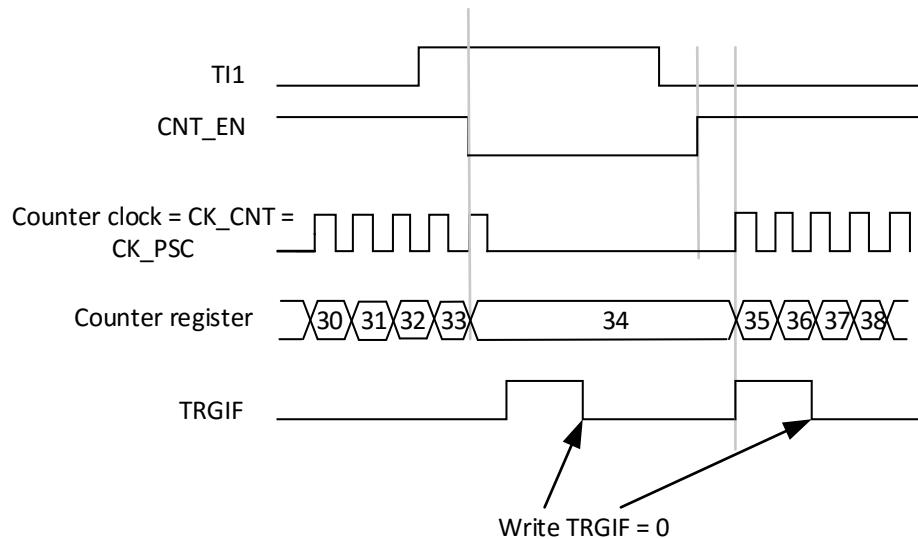
In the following example, the counter counts up only when TI1 input is low:

- Configure channel 1 to detect low levels on TI1. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C1SEL bits select the input capture source. Set C1SEL = 01 in the TMRx_CCM1 register. Write C1P = 1 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as gated mode by writing SMSEL = 101 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.
- Enable the counter by writing CNTEN = 1 in the TMRx_CTRL1 register. In gated mode, the counter cannot be enabled if CNTEN = 0, no matter what the trigger input level is.

The counter starts counting on the internal clock as long as TI1 is low and stops as soon as TI1 becomes high. The TRGIF flag in the TMRx_STS register is set both when the counter starts and stops.

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

Figure 10-125 Control Circuit in Gated Mode



Slave mode: Trigger mode

The counter can be enabled according to the selected level of input.

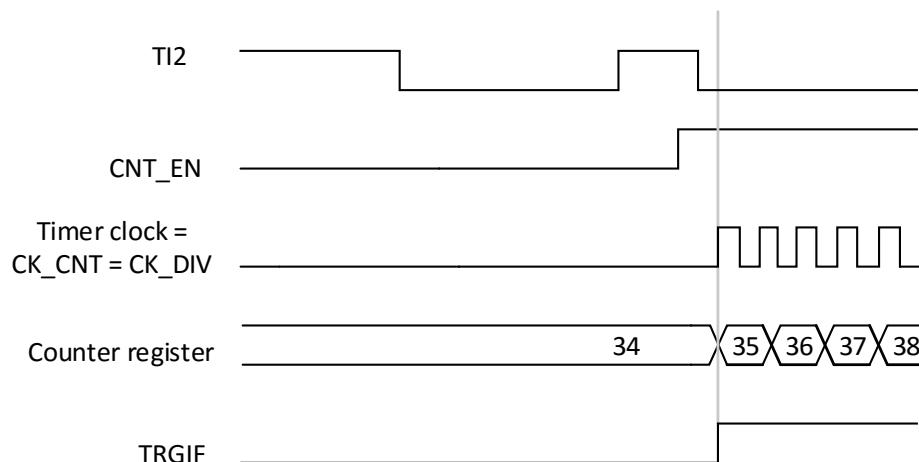
In the following example, the counter starts to count up at rising edge on TI2 input:

- Configure channel 2 to detect the rising edge on TI2. Configure the input filter duration (in this example, no filter is needed, so keep IC1DF = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The C2SEL bits are only used to select the input capture source. Set C2SEL = 01 in the TMRx_CCM1 register. Write C2P = 0 in the TMRx_CCE register to confirm the polarity (and detect low level only).
- Configure the timer as trigger mode by writing SMSEL = 110 in the TMRx_SMC register. Select TI2 as the input source by writing TRGSEL = 110 in the TMRx_SMC register.

The counter starts counting on the internal clock as long as a rising edge occurs on TI2, and at the same time the TRGIF flag is set.

The delay between the rising edge on TI2 and enabling counting is determined by the resynchronization circuit on TI2 input.

Figure 10-126 Control Circuit in Trigger Mode



Slave mode: External Clock Mode 2 + Trigger Mode

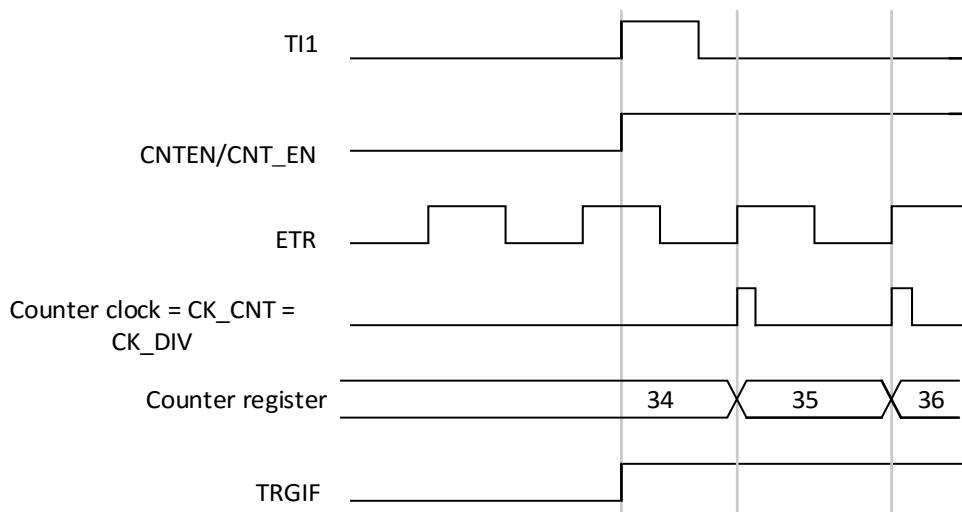
The external clock mode 2 can be used with another slave mode (except for external clock mode 1 and encoder mode). In this case, the ETR signal is used as external clock input, and another input can be trigger input in reset mode, gated mode, or trigger mode. Selecting ETR as TRGI through the TRGSEL bits in the TMRx_SMC register is not recommended.

In the following example, as soon as a rising edge of TI1 occurs, the counter counts up once at each rising edge of the ETR signal:

1. Configure the external trigger input circuit through the TMRx_SMC register:
 - ETDF = 0000: No filter
 - ETD = 00: Prescaler is disabled.
 - ETRGP = 0: Detect the rising edges on ETR and set ECLKEN = 1 to enable the external clock mode 2.
2. Configure channel 1 as follows to detect the rising edges on TI:
 - IC1DF = 0000: No filter
 - The capture prescaler is not used for triggering and does not need to be configured.
 - Set C1SEL = 01 in the TMRx_CCM1 register to select the input capture source.
 - Set C1P = 0 in the TMRx_CCE register to confirm polarity (and detect rising edge only).
3. Configure the timer as trigger mode by writing SMSEL = 110 in the TMRx_SMC register. Select TI1 as the input source by writing TRGSEL = 101 in the TMRx_SMC register.

A rising edge on TI1 enables the counter and sets the TRGIF flag. The counter then counts on ETR rising edges. The delay between the rising edge of the ETR signal and the actual reset of the counter is determined by the resynchronization circuit on ETRP input.

Figure 10-127 Control Circuit in External Clock Mode 2 + Trigger Mode



10.4.3.20 Timer Synchronization

All TMR timers are connected internally for synchronization or chaining. Please refer to [Section 10.2.3.15](#) for more details.

10.4.3.21 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the TMRx counter either continues to work normally or stops, depending on DBG_TMRx_STOP configuration in DBG module. For more details, please refer to [Section 22.2.2](#).

10.4.4 TMR1, TMR8, and TMR15 Registers Description

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

In Table 10-12, TMR1, TMR8, and TMR15 registers are mapped to a 16-bit addressable space.

Table 10-12 TMR1, TMR8, and TMR15 –Register Table and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
0x00	TMRx_CTRL1	Reserved														0	0	0	0	0	0	0	0	0	
	Reset Value															0	0	0	0	0	0	0	0	0	
0x04	TMRx_CTRL2	Reserved														0	0	0	0	0	0	0	0	0	
	Reset Value															0	0	0	0	0	0	0	0	0	
0x08	TMRx_SMC	Reserved														0	ETRGP	0	0	0	0	0	0	0	0
	Reset Value															0	ECLKEN	0	0	0	0	0	0	0	0
0x0C	TMRx_DIE	Reserved														0	TRGDE	0	0	0	0	0	0	0	0
	Reset Value															0	HALLDE	0	0	0	0	0	0	0	0
0x10	TMRx_STS	Reserved														0	C4OF	0	C4DE	0	ETD[1:0]	0	0	0	0
	Reset Value															0	C3OF	0	C3DE	0	ETDF[3:0]	0	0	0	0
0x14	TMRx_EVEG	Reserved														0	C1OF	0	C1DE	0	OC2IS	0	0	0	0
	Reset Value															0	Reserved	0	UEVDE	0	OC1IS	0	0	0	0
0x18	TMRx_CCM1 output compare mode	Reserved														0	OC2DIS	0	BRKIE	0	MSMODO	0	0	0	0
	Reset Value															0	OC2MODE[2:0]	0	HALLIF	0	TRGSEL[2:0]	0	0	0	0
	TMRx_CCM1 input capture mode	Reserved														0	OC4PE	0	OC2PEN	0	TRGIF	0	0	0	0
	Reset Value															0	OC4FEN	0	OC2FEN	0	HALLIF	0	0	0	0
0x1C	TMRx_CCM2 output compare mode	Reserved														0	C4SEL[1:0]	0	C2SEL[1:0]	0	C4G	0	0	0	0
	Reset Value															0	OC3DIS	0	IC1DF[3:0]	0	C4IE	0	0	0	0
	OC3MODE[2:0]	Reserved														0	OC4PEN	0	OC1PEN	0	IC1DIV[1:0]	0	0	0	0
	OC3FEN															0	OC3FEN	0	OC1FEN	0	C2G	0	0	0	0
	C3SEL[1:0]	Reserved														0	C3SEL[1:0]	0	C1SEL[1:0]	0	C1G	0	0	0	0
	Reset Value															0	UEVG	0	UEVIE	0	UEVIF	0	0	0	0

	TMRx_CCM2 input compare mode	Reserved		IC4DF[3:0]				IC4DIV[1:0]				C4SI[1:0]				IC3DF[3:0]				IC3DIV[1:0]				C3SEL[1:0]						
	Reset Value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0x20	TMRx_CCE	Reserved				C4P	C4EN	C3NP	C3NEN	C3P	C3EN	C2NP	C2NEN	C2P	C2NEN	C1NP	C1NEN	C1P	C1EN	C3DIV[1:0]	C3DF[3:0]	C3SEL[1:0]	C3EN	C2P	C2NEN	C1NP	C1NEN	C1P	C1EN	
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	TMRx_CNT	Reserved				CNT[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TMRx_DIV	Reserved				DIV[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TMRx_AR	Reserved				AR[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x30	TMRx_RC	Reserved				RC[7:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x34	TMRx_CC1	Reserved				CC1[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TMRx_CC2	Reserved				CC2[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	TMRx_CC3	Reserved				CC3[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	TMRx_CC4	Reserved				CC4[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	TMRx_BRKDT	Reserved				MOEN	AOEN	BRKP	BRKEN	OSIMR	OSIMI	LOCKC[1:0]	DTGS[7:0]																	
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	TMRx_DMAC	Reserved				DBLEN[4:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	TMRx_DMABA	Reserved				DMABA[15:0]																								
	Reset Value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

10.4.4.1 TMR1, TMR8, and TMR15 Control Register 1 (TMRx_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	res	CLKDIV [1:0]	ARP EN	CMSEL [1:0]	DIR	OPMODE	UEVRS	UEVDIS	CNTEN							
Bit 15:10 Reserved. Always read as 0.																															

Bit 9:8	CLKDIV[1:0]: Clock division Define the division ratio between the timer clock (CK_INT) frequency, dead-time, and sampling frequency used by the dead-time generator and digital filters (ETR, TIx). 00: $t_{DTS} = t_{CK_INT}$ 01: $t_{DTS} = 2 \times t_{CK_INT}$ 10: $t_{DTS} = 4 \times t_{CK_INT}$ 11: Reserved, do not use this configuration.
Bit 7	ARPEN: Auto-reload preload enable 0: TMRx_AR register is not buffered. 1: TMRx_AR register is buffered.
Bit 6:5	CMSEL[1:0]: Center-aligned mode selection 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR). 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set only when the counter is counting down. 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set only when the counter is counting up. 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured as output (CxSEL = 00 in the TMRx_CCMx register) are set both when the counter is counting up or down. <i>Note: It is not allowed to switch from edge-aligned mode to center-aligned mode when the counter is enabled (CNTEN = 1).</i>
Bit 4	DIR: Direction 0: The counter counts up. 1: The counter counts down. <i>Note: This bit is read-only when the counter is configured in center-aligned mode or encoder mode.</i>
Bit 3	OPMODE: One pulse mode 0: The counter does not stop at update event. 1: The counter stops at the next update event (The CNTEN bit is cleared).
Bit 2	UEVRS: Update request source This bit is set and cleared by software to select the UEV event sources. 0: Any of the following events generates an update interrupt or DMA request if update interrupt or DMA request is enabled: <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UEVG bit - An update is generated from the slave mode controller. 1: Only counter overflow/underflow generates an update interrupt or DMA request if update interrupt or DMA request is enabled.
Bit 1	UEVDIS: Update disable This bit is set and cleared by software to enable/disable UEV event generation. 0: UEV is enabled. An update event (UEV) is generated by any of the following events: <ul style="list-style-type: none"> - Counter overflow/underflow - Setting the UEVG bit - An update is generated from the slave mode controller. Buffered registers are loaded with their preload values. 1: UEV is disabled. Update events are not generated, and shadow registers (AR, DIV, and CCx) keep their values. The counter and the prescaler are reinitialized if the UEVG bit is set or the slave mode controller sends a hardware reset.
Bit 0	CNTEN: Counter enable 0: The counter is disabled. 1: The counter is enabled. <i>Note: External clock, gated mode, and encoder mode can work only after the CNTEN bit is previously set by software. Trigger mode can set the CNTEN bit automatically by hardware.</i>

10.4.4.2 TMR1, TMR8, and TMR15 Control Register 2 (TMRx_CTRL2)

Address offset: 0x04

Reset value: 0x0000

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	OC4 IS	OC3 NIS	OC3 IS	OC2 NIS	OC2 IS	OC1 NIS	OC1 IS	TI1SEL		MMSEL[2:0]	CDS EL	CUS EL	Reserve d	CPC		
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	

Bit 15	Reserved. Always read as 0.
Bit 14	OC4IS: Output idle state 4 (OC4 output). Please refer to the OC1IS bit.
Bit 13	OC3NIS: Output idle state 3 (OC3N output). Please refer to the OC1NIS bit.
Bit 12	OC3IS: Output idle state 3 (OC3 output). Please refer to the OC1IS bit.
Bit 11	OC2NIS: Output idle state 2 (OC2N output). Please refer to the OC1NIS bit.
Bit 10	OC2IS: Output idle state 2 (OC2 output). Please refer to the OC1IS bit.
Bit 9	OC1NIS: Output idle state 1 (OC1N output) 0: When MOEN = 0, OC1N = 0 after dead-time. 1: When MOEN = 0, OC1N = 1 after dead-time. <i>Note: This bit cannot be changed if the LOCKC bit (TMRx_BRKDT register) level 1, 2, or 3 is set.</i>
Bit 8	OC1IS: Output idle state 1 (OC1 output) 0: When MOEN = 0, if OC1N is done, OC1 = 0 after dead-time. 1: When MOEN = 0, if OC1N is done, OC1 = 1 after dead-time. <i>Note: This bit cannot be changed if the LOCKC bit (TMRx_BRKDT register) level 1, 2, or 3 is set.</i>
Bit 7	TI1SEL: TI1 selection 0: TMRx_CH1 pin is connected to TI1 input. 1: TMRx_CH1, TMRx_CH2, and TMRx_CH3 pins are connected to the TI1 input after XORed.
Bit 6:4	MMSEL[2:0]: Master mode selection The 3 bits select the synchronization information (TRGO) to be sent from master mode to slave timers. The possible combinations are as follows: 000: Reset – The UEVG bit in the TMRx_EVEG register is used as trigger output (TRGO). If the reset is generated by the trigger input (slave mode controller is in reset mode), the signal on TRGO is delayed compared with the actual reset. 001: Enable – The counter enable signal, CNT_EN, is used as trigger output (TRGO). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The counter enable signal is generated by a logic OR between CNTEN control bit and the trigger input signal in gated mode. When the counter enable signal is controlled by the trigger input, there is a delay on TRGO, unless the master/slave mode is selected (Please refer to the MSMODE bit description in the TMRx_SMC register). 010: Update – The update event is selected as trigger output (TRGO). For instance, a master timer can be used as a prescaler for a slave timer. 011: Compare Pulse – After a capture or a compare match occurs, the trigger output sends a positive pulse (TRGO) when the C1IF flag is to be set (even if it is already high). 100: Compare – OC1REF signal is used as trigger output (TRGO). 101: Compare – OC2REF signal is used as trigger output (TRGO). 110: Compare – OC3REF signal is used as trigger output (TRGO). 111: Compare – OC4REF signal is used as trigger output (TRGO).
Bit 3	CDSEL: Capture/Compare DMA selection 0: CCx DMA request is sent when CCx event occurs. 1: CCx DMA request is sent when update event occurs.
Bit 2	CUSEL: Capture/compare control update selection 0: When capture/compare control bits are preloaded (CPC = 1), they are updated only by setting the HALL bit. 1: When capture/compare control bits are preloaded (CPC = 1), they are updated by setting the HALL bit or a rising edge on TRGI. <i>Note: This bit only acts on channels that have complementary output.</i>
Bit 1	Reserved. Always read as 0.

Bit 0	CPC: Capture/compare preloaded control 0: The CxEN, CxNEN, and OCxMODE bits are not preloaded. 1: The CxEN, CxNEN and OCxMODE bits are preloaded. After being written, they are updated only when a HALL event occurs (the HALL bit is set or a rising edge is detected on TRGI, depending on the CUSEL bit). <i>Note: This bit only acts on channels that have complementary output.</i>
-------	--

10.4.4.3 TMR1, TMR8, and TMR15 Slave Mode Control Register (TMRx_SMC)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETR GP	ECL KEN	ETD[1:0]		ETDF[3:0]	MSM ODE	TRGSEL[2:0]	Res erve d		SMSEL[2:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw

Bit 15	ETRGP: External trigger polarity This bit selects whether ETR or inverted ETR is used for trigger operations. 0: ETR is non-inverted and active at high level or rising edge. 1: ETR is inverted and active at low level or falling edge.
Bit 14	ECLKEN: External clock enable This bit enables external clock mode 2. 0: External clock mode 2 is disabled 1: External clock mode 2 is enabled. The counter is clocked by any active edge on the ETRF signal. <i>Note 1: Setting the ECLKEN bit has the same effect as selecting external clock mode 1 with TRGI connected to ETRF (SMSEL = 111 and TRGSEL = 111).</i> <i>Note 2: It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode, and trigger mode. Nevertheless, TRGI must not be connected to ETRF in this case (TRGSEL bits must not be 111).</i> <i>Note 3: If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is ETRF.</i>
Bit 13:12	ETD[1:0]: External trigger divide External trigger signal ETRP frequency must be 1/4 of TMRxCLK frequency at most. A prescaler can be used to reduce ETRP frequency when inputting faster external clocks. 00: Prescaler is OFF. 01: ETRP frequency is divided by 2. 10: ETRP frequency is divided by 4. 11: ETRP frequency is divided by 8.
Bit 11:8	ETDF[3:0]: External trigger filter The bits define the frequency used to sample TETRP signal and the length of ETRP digital filter. In fact, the digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output: 0000: No filter, sampling is done at f_{DTS} 1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6 0001: $f_{SAMPLING} = f_{CK_INT}$, N = 2 1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8 0010: $f_{SAMPLING} = f_{CK_INT}$, N = 4 1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5 0011: $f_{SAMPLING} = f_{CK_INT}$, N = 8 1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6 0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6 1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8 0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8 1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5 0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6 1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6 0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8 1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8
Bit 7	MSMODE: Master/Slave mode 0: No effect 1: The event on the trigger input (TRGI) is delayed to allow a perfect synchronization between the current timer and its slaves (through TRGO). It is useful when synchronizing several timers on a single external event.

Bit 6:4	TRGSEL[2:0]: Trigger selection The bits select the trigger input used to synchronize the counter. 000: Internal Trigger 0 (ITR0) 100: TI1 Edge Detector (TI1F_ED) 001: Internal Trigger 1 (ITR1) 101: Filtered Timer Input 1 (TI1FP1) 010: Internal Trigger 2 (ITR2) 110: Filtered Timer Input 2 (TI2FP2) 011: Internal Trigger (ITR3) 111: External Trigger Input (ETRF) Please refer to Table 10-13 for the details on ITRx. <i>Note: These bits must be changed only when they are not used (e.g. when SMSEL = 000) to avoid wrong edge detections at transitions.</i>
Bit 3	Reserved. Always read as 0.
Bit 2:0	SMSEL[2:0]: Slave mode selection When external signals are selected, the active edge of the trigger signal (TRGI) is relevant to the selected external input polarity (Please refer to input control register and control register description.) 000: Slave mode is disabled - If CNTEN = 1, the prescaler is clocked directly by the internal clock. 001: Encoder mode 1 - Counter counts up/down on TI1FP1 edge depending on TI2FP2 level. 010: Encoder mode 2 - Counter counts up/down on TI2FP2 edge depending on TI1FP1 level. 011: Encoder mode 3 - Counter counts up/down on both TI1FP1 and TI2FP2 edges depending on the level of the other input. 100: Reset Mode - Rising edge of the selected trigger input (TRGI) reinitializes the counter and generates an update register signal. 101: Gated Mode - The counter clock is enabled when the trigger input (TRGI) is high. The counter stops (but is not reset) once the trigger becomes low. Both start and stop of the counter are controlled. 110: Trigger Mode - The counter starts at a rising edge of the trigger input TRGI (but it is not reset). Only the start of the counter is controlled. 111: External Clock Mode 1 - Rising edges of the selected trigger input (TRGI) clocks the counter. <i>Note: The gated mode must not be used if TI1F_EN is selected as the trigger input (TRGSEL = 100). This is because TI1F_ED outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger input signal.</i>

Table 10-13 TMRx Internal Trigger Connection

Slave Timer	ITR0 (TRGSEL = 000)	ITR1 (TRGSEL = 001)	ITR2 (TRGSEL = 010)	ITR3 (TRGSEL = 011)
TMR1	TMR5	TMR2	TMR3	TMR4
TMR8	TMR1	TMR2	TMR4	TMR5
TMR15	TMR8	TMR2	TMR4	TMR5

10.4.4.4 TMR1, TMR8, and TMR15 DMA/Interrupt Enable Register (TMRx_DIE)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve	TRG DE	HAL LDE	C4DE	C3D E	C2D E	C1D E	UEV DE	BRKI E	TRGI E	HALL IE	C4IE	C3IE	C2IE	C1IE	UEVI E

res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15	Reserved. Always read as 0.														
Bit 14	TRGDE: Trigger DMA request enable 0: Trigger DMA request is disabled. 1: Trigger DMA request is enabled.														

Bit 13	HALLDE: HALL DMA request enable 0: HALL DMA request is disabled. 1: HALL DMA request is enabled.
Bit 12	C4DE: Capture/Compare 4 DMA request enable 0: Capture/Compare 4 DMA request is disabled. 1: Capture/Compare 4 DMA request is enabled.
Bit 11	C3DE: Capture/Compare 3 DMA request enable 0: Capture/Compare 3 DMA request is disabled. 1: Capture/Compare 3 DMA request is enabled.
Bit 10	C2DE: Capture/Compare 2 DMA request enable 0: Capture/Compare 2 DMA request is disabled. 1: Capture/Compare 2 DMA request is enabled.
Bit 9	C1DE: Capture/Compare 1 DMA request enable 0: Capture/Compare 1 DMA request is disabled. 1: Capture/Compare 1 DMA request is enabled.
Bit 8	UEVDE: Update DMA request enable 0: Update DMA request is disabled. 1: Update DMA request is enabled.
Bit 7	BRKIE: Break interrupt enable 0: Break interrupt is disabled. 1: Break interrupt is enabled.
Bit 6	TRGIE: Trigger interrupt enable 0: Trigger interrupt is disabled. 1: Trigger interrupt is enabled.
Bit 5	HALLIE: HALL interrupt enable 0: HALL interrupt is disabled. 1: HALL interrupt is enabled.
Bit 4	C4IE: Capture/Compare 4 interrupt enable 0: Capture/Compare 4 interrupt is disabled. 1: Capture/Compare 4 interrupt is enabled.
Bit 3	C3IE: Capture/Compare 3 interrupt enable 0: Capture/Compare 3 interrupt is disabled. 1: Capture/Compare 3 interrupt is enabled.
Bit 2	C2IE: Capture/Compare 2 interrupt enable 0: Capture/Compare 2 interrupt is disabled. 1: Capture/Compare 2 interrupt is enabled.
Bit 1	C1IE: Capture/Compare 1 interrupt enable 0: Capture/Compare 1 interrupt is disabled. 1: Capture/Compare 1 interrupt is enabled.
Bit 0	UEVIE: Update interrupt enable 0: Update interrupt is disabled. 1: Update interrupt is enabled.

10.4.4.5 TMR1, TMR8, and TMR15 Status Register (TMRx_STS)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	C4O F	C3O F	C2O F	C1O F	Reserve d	BRK IF	TRG IF	HAL LIF	C4I F	C3I F	C2I F	C1I F	UEV IF		
res	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:13		Reserved. Always read as 0.													

Bit 12	C4OF: Capture/Compare 4 overcapture flag Please refer to C1OF description.
Bit 11	C3OF: Capture/Compare 3 overcapture flag Please refer to C1OF description.
Bit 10	C2OF: Capture/Compare 2 overcapture flag Please refer to C1OF description.
Bit 9	C1OF: Capture/Compare 1 overcapture flag This flag is set by hardware only when the corresponding channel is configured as input capture mode. It is cleared by writing '0'. 0: No overcapture is detected. 1: When the counter value is captured to the TMRx_CC1 register, C1IF flag is already set.
Bit 8	Reserved. Always read as 0.
Bit 7	BRKIF: Break interrupt flag This flag is set by hardware once the break input goes active. It can be cleared by software if the break input is inactive. 0: No break event occurs. 1: An active level is detected on the break input.
Bit 6	TRGIF: Trigger interrupt flag This flag is set by hardware on trigger event (When the slave mode controller is in all modes except for gated mode, active edge is detected on TRGI input, or any edge in the gated mode). It is cleared by software. 0: No trigger event occurs. 1: Trigger interrupt is pending.
Bit 5	HALLIF: HALL interrupt flag This flag is set by hardware on HALL event (when capture/compare control bits: CxEN, CxNEN, OCxMODE are updated). It is cleared by software. 0: No HALL event occurs. 1: HALL interrupt is pending.
Bit 4	C4IF: Capture/Compare 4 interrupt flag Please refer to C1IF description.
Bit 3	C3IF: Capture/Compare 3 interrupt flag Please refer to C1IF description.
Bit 2	C2IF: Capture/Compare 2 interrupt flag Please refer to C1IF description.
Bit 1	C1IF: Capture/Compare 1 interrupt flag If channel CC1 is configured as output mode: This flag is set by hardware when the counter value matches the compare value, but not in center-aligned mode (Please refer to the CMSEL bits in the TMRx_CTRL1). It is cleared by software. 0: No match 1: The TMRx_CNT value matches the TMRx_CC1 value. When the contents of TMRx_CC1 are greater than the contents of TMRx_AR, the C1IF bit goes high on the counter overflow in upcounting and up/down-counting modes, or on underflow in downcounting mode. If channel CC1 is configured as input mode: This bit is set by hardware on a capture. It is cleared by software or by reading the TMRx_CC1 register. 0: No input capture occurs. 1: The counter value is captured to the TMRx_CC1 register (An edge which matches the selected polarity is detected on IC1).

Bit 0	UEVIF: Update interrupt flag This bit is set by hardware on an update event. It is cleared by software. 0: No update event occurs. 1: Update interrupt is pending. This bit is set by hardware when the registers are updated: <ul style="list-style-type: none"> - If UEVDIS = 0 in the TMRx_CTRL1 register, when repetition counter value overflows or underflows (An update event is generated when the repetition counter = 0). - If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, an update event is generated when UEVG = 1 in the TMRx_EVEG register (when the counter CNT is reinitialized by software). - If UEVDIS = 0 and UEVRS = 0 in the TMRx_CTRL1 register, when the counter CNT is reinitialized by trigger events (Please refer to Section 10.4.4.3).
-------	---

10.4.4.6 TMR1, TMR8, and TMR15 Event Generation Register (TMRx_EVEG)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
									BRK G	TRG G	HALL G	C4G	C3G	C2G	C1G	UEV G

res	rw														
-----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bit 15:8	Reserved. Always read as 0.
Bit 7	BRKG: Break generation This bit is set by software to generate a break event. It is cleared automatically by hardware. 0: No action 1: A break event is generated. At this time, MOEN = 0 and BRKIF = 1. The corresponding interrupt and DMA are generated if enabled.
Bit 6	TRGG: Trigger generation This bit is set by software to generate a trigger event. It is cleared automatically by hardware. 0: No action 1: TRGIF = 1 in the TMRx_STS register, the corresponding interrupt and DMA is generated if enabled.
Bit 5	HALLG: Capture/Compare update generation This bit is set by software. It is cleared automatically by hardware. 0: No action 1: When CPC = 1, the CxEN, CxNEN, and OCxMODE bits are allowed to be updated. <i>Note: This bit only acts on channels that have complementary output.</i>
Bit 4	C4G: Capture/compare 4 generation Please refer to C1G description.
Bit 3	C3G: Capture/compare 3 generation Please refer to C1G description.
Bit 2	C2G: Capture/compare 2 generation Please refer to C1G description.
Bit 1	C1G: Capture/compare 1 generation This bit is set by software to generate a capture/compare event. It is cleared automatically by hardware. 0: No action 1: A capture/compare event is generated on channel 1 CC1: If channel CC1 is configured as output: Set C1IF = 1. The corresponding interrupt and DMA request is generated if enabled. If channel CC1 is configured as input: The current value of the counter is captured to the TMRx_CC1 register. Set C1IF = 1. The corresponding interrupt and DMA request is generated if enabled. If C1IF is already '1', set C1OF = 1.

Bit 0	UEVG: Update generation This bit can be set by software, and it is automatically cleared by hardware. 0: No action 1: Re-initialize the counter and generate an update. Please note that the prescaler counter is also cleared, but the prescaler ratio is not affected. The counter is cleared if the center-aligned mode is selected, or if DIR = 0 (upcounting), else it takes the TMRx_AR value if DIR = 1 (downcounting).
-------	--

10.4.4.7 TMR1, TMR8, and TMR15 Capture/Compare Mode Register 1 (TMRx_CCM1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by the corresponding CxSEL bits. All the other bits of this register have different functions in input and output modes. For a given bit, OCxx describes its function when the channel is in output mode, while ICxx describes its function when the channel is in input mode. Attention must be given to the fact that the same bit can have different definitions for the input stage and for the output stage.

Output compare mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 DIS	OC2MODE[2:0]	OC2 PEN	OC2 FEN	C2SEL[1:0]	OC1 DIS	OC1MODE[2:0]	OC1 PEN	OC1 FEN	C1SEL[1:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15	OC2DIS: Output compare 2 clear enable
Bit 14:12	OC2MODE[2:0]: Output compare 2 mode
Bit 11	OC2PEN: Output compare 2 preload enable
Bit 10	OC2FEN: Output compare 2 fast enable
Bit 9:8	C2SEL[1:0]: Capture/Compare 2 selection This bit defines the channel direction (input/output), and the selection of input pin: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C2SEL can be written only when the channel is disabled (C2NEN = 0 in the TMRx_CCE register).</i>
Bit 7	OC1DIS: Output compare 1 clear enable 0: OC1REF is not affected by ETRF input. 1: Once high level is detected on ETRF input, clear OC1REF = 0.

Bit 6:4	<p>OC1MODE[2:0]: Output compare 1 mode</p> <p>The bits define the behavior of the output reference signal, OC1REF, and OC1REF determines the OC1 and OC1N values. OC1REF is active high, while active levels of OC1 and OC1N are determined by the C1P and C1NP bits.</p> <p>000: Frozen. The comparison between the output compare register TMRx_CC1 and the counter TMRx_CNT has no effect on OC1REF.</p> <p>001: Set channel 1 to active level on match. OC1REF is forced high when the value of the counter TMRx_CNT matches the capture/compare register1 (TMRx_CC1).</p> <p>010: Set channel 1 to inactive level on match. OC1REF is forced low when the value of the counter TMRx_CNT matches the capture/compare register1 (TMRx_CC1).</p> <p>011: Toggle. Toggle OC1REF level when TMRx_CC1 = TMRx_CNT.</p> <p>100: Force inactive level. OC1REF is forced low.</p> <p>101: Force active level. OC1REF is forced high.</p> <p>110: PWM mode 1— In upcounting, channel 1 is active once TMRx_CNT < TMRx_CC1, else inactive; in downcounting, channel 1 is inactive once TMRx_CNT > TMRx_CC1 (OC1REF = 0), else active (OC1REF = 1).</p> <p>111: PWM mode 2—In upcounting, channel 1 is inactive once TMRx_CNT < TMRx_CC1, else active; in downcounting, channel 1 is active once TMRx_CNT > TMRx_CC1, else inactive.</p> <p><i>Note 1: Once the LOCK level is set to 3 (the LOCKC bit in the TMRx_BRKDT register) and C1SEL = 00 (this channel configured as output), the bit cannot be changed.</i></p> <p><i>Note 2: In PWM mode 1 or PWM mode 2, OC1REF level changes only when the comparison result changes or when the output compare mode switches from frozen mode to PWM mode .</i></p>
Bit 3	<p>OC1PEN: Output compare 1 preload enable</p> <p>0: Preload function of TMRx_CC1 is disabled. TMRx_CC1 can be written at any time, and the new value takes effect immediately.</p> <p>1: Preload function of TMRx_CC1 is enabled. Read/Write operations only access the preload register. TMRx_CC1 preload value is sent to the active register at each update event.</p> <p><i>Note 1: Once the LOCK level is set to 3 (the LOCKC bit in the TMRx_BRKDT register) and C1SEL = 00 (this channel configured as output), the bit cannot be changed.</i></p> <p><i>Note 2: The PWM mode can be used without validating the preload register only in one-pulse mode (OPMODE = '1' in the TMRx_CTRL1 register). Else, the behavior is not guaranteed.</i></p>
Bit 2	<p>OC1FEN: Output compare 1 fast enable</p> <p>This bit is used to accelerate the CC output's response to trigger the input event.</p> <p>0: CC1 behaves normally with the counter and CC1 values even when the trigger is ON. When an active edge occurs on the trigger input, the minimum delay to activate CC1 output is 5 clock cycles.</p> <p>1: An active edge on the trigger input acts like a compare match. Therefore, OC is set to the compare level which is irrelevant to the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles.</p> <p>This bit only acts when channel is configured as PWM1 or PWM2 mode.</p>
Bit 1:0	<p>C1SEL[1:0]: Capture/Compare 1 selection</p> <p>The bits define the channel direction (input/output), and input pin selection:</p> <p>00: CC1 channel is configured as output.</p> <p>01: CC1 channel is configured as input, IC1 is mapped on TI1.</p> <p>10: CC1 channel is configured as input, IC1 is mapped on TI2.</p> <p>11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i></p>

Input capture mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2DF[3:0]			IC2DIV[1:0]		C2SEL[1:0]		IC1DF[3:0]			IC1DIV[1:0]		C1SEL[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:12	IC2DF[3:0]: Input capture 2 filter																
Bit 11:10	IC2DIV[1:0]: Input capture 2 prescaler																
Bit 9:8	<p>C2SEL[1:0]: Capture/Compare 2 selection The bits define the channel direction (input/output), and input pin selection: 00: CC2 channel is configured as output. 01: CC2 channel is configured as input, IC2 is mapped on TI2. 10: CC2 channel is configured as input, IC2 is mapped on TI1. 11: CC2 channel is configured as input, IC2 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C2SEL can be written only when the channel is disabled (C2EN = '0' in the TMRx_CCE register).</i></p>																
Bit 7:4	<p>IC1DF[3:0]: Input capture 1 filter The bits define the frequency used to sample ETRP signal and the length of ETRP digital filter. In fact, the digital filter is an event counter which records N consecutive events that are needed to generate a transition on the output:</p> <table> <tr> <td>0000: No filter, sampling is done at f_{DTS}</td> <td>1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$</td> </tr> <tr> <td>0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$</td> <td>1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$</td> </tr> <tr> <td>0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$</td> <td>1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$</td> </tr> <tr> <td>0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$</td> <td>1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$</td> </tr> <tr> <td>0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$</td> <td>1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$</td> </tr> <tr> <td>0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$</td> <td>1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$</td> </tr> <tr> <td>0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$</td> <td>1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$</td> </tr> <tr> <td>0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$</td> <td>1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$</td> </tr> </table>	0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$	0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$	1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$	0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$	1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$	0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$	1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$	0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$	1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$	0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$	1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$	0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$	1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$	0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$	1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$
0000: No filter, sampling is done at f_{DTS}	1000: $f_{SAMPLING} = f_{DTS}/8, N = 6$																
0001: $f_{SAMPLING} = f_{CK_INT}, N = 2$	1001: $f_{SAMPLING} = f_{DTS}/8, N = 8$																
0010: $f_{SAMPLING} = f_{CK_INT}, N = 4$	1010: $f_{SAMPLING} = f_{DTS}/16, N = 5$																
0011: $f_{SAMPLING} = f_{CK_INT}, N = 8$	1011: $f_{SAMPLING} = f_{DTS}/16, N = 6$																
0100: $f_{SAMPLING} = f_{DTS}/2, N = 6$	1100: $f_{SAMPLING} = f_{DTS}/16, N = 8$																
0101: $f_{SAMPLING} = f_{DTS}/2, N = 8$	1101: $f_{SAMPLING} = f_{DTS}/32, N = 5$																
0110: $f_{SAMPLING} = f_{DTS}/4, N = 6$	1110: $f_{SAMPLING} = f_{DTS}/32, N = 6$																
0111: $f_{SAMPLING} = f_{DTS}/4, N = 8$	1111: $f_{SAMPLING} = f_{DTS}/32, N = 8$																
Bit 3:2	<p>IC1DIV[1:0]: Input capture 1 prescaler The bits define CC1 input (IC1) prescaler ratio. Once C1EN = '0' (in the TMRx_CCE register), the prescaler is reset. 00: No prescaler. Capture is done each time when an edge is detected on the capture input. 01: Capture is done once every 2 events. 10: Capture is done once every 4 events 11: Capture is done once every 8 events.</p>																
Bit 1:0	<p>C1SEL[1:0]: Capture/Compare 1 selection The bits define the channel direction (input/output), and input pin selection: 00: CC1 channel is configured as output. 01: CC1 channel is configured as input, IC1 is mapped on TI1. 10: CC1 channel is configured as input, IC1 is mapped on TI2. 11: CC1 channel is configured as input, IC1 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C1SEL can be written only when the channel is disabled (C1EN = '0' in the TMRx_CCE register).</i></p>																

10.4.4.8 TMR1, TMR8, and TMR15 Capture/Compare Mode Register 2 (TMRx_CCM2)

Address offset: 0x1C

Reset value: 0x0000

Please refer to the above-mentioned CCM1 Registers Description.

Output compare mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 DIS	OC4MODE[2:0]		OC4 PEN	OC4 FEN	C4SEL[1:0]	OC3 DIS	OC3MODE[2:0]		OC3 PEN	OC3 FEN	C3SEL[1:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15		OC4DIS: Output compare 4 clear enable													

Bit 14:12	OC4MODE[2:0]: Output compare 4 mode
Bit 11	OC4PEN: Output compare 4 preload enable
Bit 10	OC4FEN: Output compare 4 fast enable
Bit 9:8	<p>C4SEL[1:0]: Capture/Compare 4 selection The bits define the channel direction (input/output), and input pin selection: 00: CC4 channel is configured as output. 01: CC4 channel is configured as input, IC4 is mapped on TI4. 10: CC4 channel is configured as input, IC4 is mapped on TI3. 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: CC4 can be written only when the channel is disabled (C4EN = '0' in the TMRx_CCE register).</i></p>
Bit 7	OC3DIS: Output compare 3 clear enable
Bit 6:4	OC3MODE[2:0]: Output compare 3 mode
Bit 3	OC3PEN: Output compare 3 preload enable
Bit 2	OC3FEN: Output compare 3 fast enable
Bit 1:0	<p>C3SEL[1:0]: Capture/Compare 3 selection The bits define the channel direction (input/output), and input pin selection: 00: CC3 channel is configured as output. 01: CC3 channel is configured as input, IC3 is mapped on TI3. 10: CC3 channel is configured as input, IC3 is mapped on TI4. 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C3SEL can be written only when the channel is disabled (C3EN = '0' in the TMRx_CCE register).</i></p>

Input capture mode

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				IC4DF[3:0]		IC4DIV[1:0]	C4SEL[1:0]		IC3DF[3:0]		IC3DIV[1:0]		C3SEL[1:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:12	IC4DF[3:0]: Input capture 4 filter
Bit 11:10	IC4DIV[1:0]: Input capture 4 prescaler
Bit 9:8	<p>C4SEL[1:0]: Capture/Compare 4 selection The bits define the channel direction (input/output), and input pin selection: 00: CC4 channel is configured as output. 01: CC4 channel is configured as input, IC4 is mapped on TI4. 10: CC4 channel is configured as input, IC4 is mapped on TI3. 11: CC4 channel is configured as input, IC4 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register).</p> <p><i>Note: C4SEL can be written only when the channel is disabled (C4EN = '0' in the TMRx_CCE register).</i></p>
Bit 7:4	IC3DF[3:0]: Input capture 3 filter
Bit 3:2	IC3DIV[1:0]: Input capture 3 prescaler

Bit 1:0	C3SEL[1:0]: Capture/Compare 3 selection The bits define the channel direction (input/output), and input pin selection: 00: CC3 channel is configured as output. 01: CC3 channel is configured as input, IC3 is mapped on TI3. 10: CC3 channel is configured as input, IC3 is mapped on TI4. 11: CC3 channel is configured as input, IC3 is mapped on TRC. This mode works only when the internal trigger input is selected (by the TRGSEL bit in the TMRx_SMC register). <i>Note: C3SEL can be written only when the channel is disabled (C3EN = '0' in the TMRx_CCE register).</i>
---------	--

10.4.4.9 TMR1, TMR8, and TMR15 Capture/Compare Enable Register (TMRx_CCE)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	C4P	C4EN	C3NP	C3EN	C3P	C3EN	C2NP	C2EN	C2P	C2EN	C1NP	C1NEN	C1P	C1EN	
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:14	Reserved. Always read as 0.
Bit 13	C4P: Capture/Compare 4 output polarity Please refer to C1P description.
Bit 12	C4EN: Capture/Compare 4 output enable Please refer to C1EN description.
Bit 11	C3NP: Capture/Compare 3 complementary output polarity Please refer to C1NP description.
Bit 10	C3NEN: Capture/Compare 3 complementary output enable Please refer to C1NEN description.
Bit 9	C3P: Capture/Compare 3 output polarity Please refer to C1P description.
Bit 8	C3EN: Capture/Compare 3 output enable Please refer to C1EN description.
Bit 7	C2NP: Capture/Compare 2 complementary output polarity Please refer to C1NP description.
Bit 6	C2NEN: Capture/Compare 2 complementary output enable Please refer to C1NEN description.
Bit 5	C2P: Capture/Compare 2 output polarity Please refer to C1P description.
Bit 4	C2EN: Capture/Compare 2 output enable Please refer to C1EN description.
Bit 3	C1NP: Capture/Compare 1 complementary output polarity CC1 channel is configured as output: 0: OC1N is active high. 1: OC1N is active low. CC1 channel is configured as input: C1NP is combined with C1P to define TI1FP1/TI2FP1 polarity (Please refer to C1P description). <i>Note: This bit cannot be modified once LOCK level is set as '2' or '3' (the LOCKC bits in the TMRx_BRKDT register) and C1SEL = 00 (channel is configured as output).</i>

Bit 2	C1NEN: Capture/Compare 1 complementary output enable 0: OFF— OC1N disables output, so OC1N level depends on the values of the MOEN, OSIMI, OSIMR, OC1IS, OC1NIS, and C1EN bits. 1: ON— OC1N signal outputs to the corresponding output pin, and its output level depends on the values of MOEN, OSIMI, OSIMR , OC1IS, OC1NIS, and C1EN bits.
Bit 1	C1P: Capture/Compare 1 output polarity CC1 channel is configured as output: 0: OC1N is active high. 1: OC1N is active low. CC1 channel is configured as input: The CC1NP/CC1P bits select TI1FP1 and TI2FP1 polarity signal as trigger or capture signal. 00: Not inverted/Rising edge. Circuit is sensitive to TIxFP1 rising edge (capture, reset in trigger, external clock or trigger mode). TIxFP1 is not inverted (gated mode in trigger). 01: Inverted/Falling edge. Circuit is sensitive to TIxFP1 falling edge (capture, reset in trigger, external clock or trigger mode). TIxFP1 is inverted (gated mode in trigger). 10: Reserved 11: Not inverted/Both edges. Circuit is sensitive to both TIxFP1 rising and falling edge (capture, reset in trigger, external clock or trigger mode). TIxFP1 is not inverted (gated mode in trigger). <i>Note: This bit cannot be modified once LOCK level is set as '2' or '3' (the LOCKC bits in the TMRx_BRKDT register).</i>
Bit 0	C1EN: Capture/Compare 1 output enable CC1 channel is configured as output: 0: OFF— OC1 output is disabled. So OC1 output level depends on the values of the MOEN, OSIMI, OSIMR, OC1IS, OC1NIS, and C1NEN bits. 1: ON— OC1 signal outputs on the corresponding output pin, and its output level output level depends on the values of the MOEN, OSIMI, OSIMR, OC1IS, OC1NIS, and C1NEN bits. CC1 channel is configured as input: This bit determines whether the counter value can be captured to TMRx_CC1 register. 0: Capture is disabled. 1: Capture is enabled.

Table 10-14 Complementary Output Channel OCx and OCxN Control Bits with Break Function

Control Bit					Output State ^(Note)	
MOEN bit	OSIMI bit	OSIMR bit	CxEN bit	CxNEN bit	OCx Output State	OCxN Output State
1	X	0	0	0	Output disabled (not driven by the timer) OCx = 0, OCx_EN = 0	Output disabled (not driven by the timer) OCxN = 0, OCxN_EN = 0
		0	0	1	Output disabled (not driven by the timer) OCx = 0, OCx_EN = 0	OCxREF + polarity, OCxN = OCxREF xor CxNP, OCxN_EN = 1
		0	1	0	OCxREF + polarity, OCx = OCxREF xor CxP, OCx_EN = 1	Output disabled (not driven by the timer) OCxN = 0, OCxN_EN = 0
		0	1	1	OCxREF + polarity+ dead-time, OCx_EN = 1	OCxREF inverted + polarity + dead-time, OCxN_EN = 1
		1	0	0	Output disabled (not driven by the timer) OCx = CxP, OCx_EN = 0	Output disabled (not driven by the timer) OCxN = CxNP, OCxN_EN = 0
		1	0	1	Off-state (output enabled with inactive state) OCx = CxP, OCx_EN = 1	OCxREF + polarity, OCxN = OCxREF xor CxNP, OCxN_EN = 1

		1	1	0	OCxREF + polarity, OCx = OCxREF xor CxP, OCx_EN = 1	Off-state (output enabled with inactive state) OCxN = CxNP, OCxN_EN = 1
		1	1	1	OCxREF + polarity + dead-time, OCx_EN = 1	OCxREF inverted + polarity + dead-time, OCxN_EN = 1
0	X	0	0	0	Output disabled (not driven by the timer)	
		0	1	0	Asynchronously: OCx = CxP, OCx_EN = 0, CxN = CxNP, OCxN_EN = 0;	
		0	0	1	If the clock is present: After a dead-time, OCx = OC1IS, OCxN = OCxNIS, assuming that OC1IS and OCxNIS do not correspond to OCx and OCxN active level.	
		0	1	1	Asynchronously: OCx = CxP, OCx_EN = 1, OCxN = CxNP, OCxN_EN = 1;	
		1	0	0	If the clock is present: After a dead-time, OCx = OC1IS, OCxN = OCxNIS, assuming that OC1IS and OCxNIS do not correspond to OCx and OCxN active level.	
		1	1	0	Asynchronously: OCx = CxP, OCx_EN = 1, OCxN = CxNP, OCxN_EN = 1;	
		1	0	1	If the clock is present: After a dead-time, OCx = OC1IS, OCxN = OCxNIS, assuming that OC1IS and OCxNIS do not correspond to OCx and OCxN active level.	
		1	1	1	Asynchronously: OCx = CxP, OCx_EN = 1, OCxN = CxNP, OCxN_EN = 1;	

Note 1: If the two outputs of a channel is not used (CxEN = CxNEN = 0), OC1IS, OCxNIS, CxP, and CxNP must be cleared.

Note 2: The state of the external I/O pins connected to the complementary OCx and OCxN channels depends on the OCx and OCxN channel state and the GPIO and AFIO registers.

10.4.4.10 TMR1, TMR8, and TMR15 Counter (TMRx_CNT)

Address offset: 0x24

Reset value: 0x0000

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

10.4.4.11 TMR1, TMR8, and TMR15 Prescaler (TMRx_DIV)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0		DIV[15:0]: Prescaler value The counter clock frequency CK_CNT is fCK_DIV/(DIV[15:0]+1). DIV contains the value loaded in the active prescaler register at each update event. Update events include the counter cleared by the UEVG bit in the TMR_EVEG register, or cleared by the slave controller working in reset mode.													

10.4.4.12 TMR1, TMR8, and TMR15 Auto-reload Register (TMRx_AR)

Address offset: 0x2C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0	AR[15:0]: Auto-reload value AR contains the value to be loaded to the actual auto-reload register. When auto-reload value is null, the counter does not work. Please refer to Section 10.4.3.1 for more details on AR update and behavior.														

10.4.4.13 TMR1, TMR8, and TMR15 Repetition Counter Register (TMRx_RC)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RC[7:0]							
res								rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:8	Reserved. Always read as 0.														
Bit 7:0	RC[7:0]: Repetition counter value These bits allow users to set the update rate of the compare registers (i.e. periodic transfers from preload registers to active registers) after preload registers are enabled. The update interrupt generation rate is also affected, if the update interrupt generation is enabled. Each time the down counter RC_CNT reaches zero, an update event is generated, and it restarts counting from RC value. As RC_CNT is reloaded with RC value only at the repetition update event U_RC, any write to the TMRx_RC register is not taken into account until the next repetition update event. It means in PWM mode, (RC+1) corresponds to: <ul style="list-style-type: none"> — The number of PWM periods in edge-aligned mode — The number of half PWM period in center-aligned mode 														

10.4.4.14 TMR1, TMR8, and TMR15 Capture/Compare Register 1 (TMRx_CC1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0	CC1[15:0]: Capture/Compare channel 1 value If CC1 channel is configured as output: CC1 is the value to be loaded in the actual capture/compare 1 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM1 register (the OC1PEN bit). Else, the preload value is loaded in the active capture/compare 1 register when an update event occurs. The active capture/compare register involves in the comparison with the counter TMRx_CNT, and generates the output on OC1. If CC1 channel is configured as input: CC1 is the counter value transferred by the last input capture 1 event (IC1).														

10.4.4.15 TMR1, TMR8, and TMR15 Capture/Compare Register 2 (TMRx_CC2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CC2[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 15:0		CC2[15:0]: Capture/Compare channel 2 value If CC2 channel is configured as output: CC2 is the value to be loaded in the actual capture/compare 2 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM2 register (the OC2PEN bit). Else, the preload value is loaded in the active capture/compare 2 register when an update event occurs. The active capture/compare register involves in the comparison with the counter, TMRx_CNT, and generates output on OC2. If CC2 channel is configured as input: CC2 is the counter value transferred by the last input capture 2 event (IC2).														

10.4.4.16 TMR1, TMR8, and TMR15 Capture/Compare Register 3 (TMRx_CC3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CC3[15:0]																
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 15:0		CC3[15:0]: Capture/Compare channel 3 value If CC3 channel is configured as output: CC3 is the value to be loaded in the actual capture/compare 3 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM3 register (the OC3PEN bit). Else, the preload value is loaded in the active capture/compare 3 register when an update event occurs. The active capture/compare register involves in the comparison with the counter, TMRx_CNT, and generates output on OC3. If CC3 channel is configured as input: CC3 is the counter value transferred by the last input capture 3 event (IC3).														

10.4.4.17 TMR1, TMR8, and TMR15 Capture/Compare Register 4 (TMRx_CC4)

Address offset: 0x40

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:0	CC4[15:0]: Capture/Compare channel 4 value If CC4 channel is configured as output: CC4 is the value to be loaded in the actual capture/compare 4 register (preload value). The written value is loaded immediately to the active register if the preload feature is not selected in the TMRx_CCM4 register (the OC4PEN bit). Else, the preload value is loaded in the active capture/compare 4 register when an update event occurs. The active capture/compare register involves in the comparison with the counter, TMRx_CNT, and generates output on OC4. If CC4 channel is configured as input: CC4 is the counter value transferred by the last input capture 4 event (IC4).
----------	---

10.4.4.18 TMR1, TMR8, and TMR15 Break and Dead-time Register (TMRx_BRKDT)

Address offset: 0x44

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MO EN	AO EN	BR KP	BRK EN	OSI MR	OSI MI	LOCKC[1:0]									DTGS[7:0]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: The AOEN, BRKP, BRKEN, OSIMI, OSIMR, and DTGS[7:0] bits can be write-locked according to LOCK configuration. It is necessary to configure all of them during the first write access to the TMRx_BRKDT register.

Bit 15	MOEN: Main output enable This bit is cleared asynchronously by hardware once the break input is active. It is cleared by software or set automatically depending on the AOEN bit configuration. It acts only on the channels configured as output. 0: OC and OCN outputs are disabled or forced to idle state. 1: OC and OCN outputs are enabled if their corresponding enable bits are set (the CxEN, CxNEN bits in the TMRx_CCE register). Please refer to Section 10.4.4.9 for more details on OC/OCN enable.
Bit 14	AOEN: Automatic output enable 0: MOEN can be set only by software. 1: MOEN can be set by software or set automatically at the next update event (if the break input is not active). <i>Note: This bit cannot be modified once LOCK level is set as '1' (the LOCKC bits in the TMRx_BRKDT register).</i>
Bit 13	BRKP: Break polarity 0: Break input is active low. 1: Break input is active high. <i>Note 1: This bit cannot be modified once LOCK level is set as '1' (the LOCKC bits in the TMRx_BRKDT register).</i> <i>Note 2: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</i>
Bit 12	BRKEN: Break enable 0: Break inputs (BRK and CFD clock failure event) is disabled. 1: Break inputs (BRK and CFD clock failure event) is enabled. <i>Note 1: This bit cannot be modified once LOCK level is set as '1' (the LOCKC bits in the TMRx_BRKDT register).</i> <i>Note 2: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.</i>

Bit 11	OSIMR: Off-state selection for Run mode This bit is used when MOEN = 1 on channels having a complementary output. The OSIMR is not present in the timer without complementary output. Please refer to OC/OCN enable description for more details (Section 10.4.4.9). 0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal=0). 1: When the timer is inactive, once CxEN = 1 or CxNEN = 1, first enable OC/OCN and output the inactive level. Then, set OC/OCN enable output signal = 1 <i>Note: This bit cannot be modified once LOCK level is set as '2' (the LOCKC bits in the TMRx_BRKDT register).</i>
Bit 10	OSIMI: Off-state selection for Idle mode This bit is used when MOEN = 0 on channels configured as outputs. Please refer to OC/OCN enable description for more details (Section 10.4.4.9). 0: When the timer is inactive, OC/OCN outputs are disabled (OC/OCN enable output signal = 0). 1: When the timer is inactive, once CxEN = 1 or CxNEN = 1, OC/OCN first outputs the idle levels and OC/OCN enable output signal = 1. <i>Note: This bit cannot be modified once LOCK level is set as '2' (the LOCKC bits in the TMRx_BRKDT register).</i>
Bit 9:8	LOCKC[1:0]: Lock configuration These bits offer a write protection against software errors. 00: LOCK is OFF, no write protection to the register. 01: LOCK Level 1, the DTGS, BRKEN, BRKP, AOEN bits in the TMRx_BRKDT register and the OCXIS/OCXNIS bits in the TMRx_CTRL2 register cannot be written. 10: LOCK Level 2, all bits in LOCK Level 1 and CC polarity bits (the CxP/CCNxP bits in TMRx_CCE register, if the related channel is configured as output through the CxSEL bits) as well as OSIMR/OSIMI bits cannot be written. 11: LOCK Level 3, all bits in LOCK Level 2 and CC control bits (the OCxMODE/OCxPEN bits in TMRx_CCMx register if the related channel is configured as output through the CxSEL bits) cannot be written. <i>Note: The LOCKC bits can be written only once after the reset. Once the TMRx_BRKDT register is written, its content is frozen until the next reset.</i>
Bit 7:0	DTGS[7:0]: Dead-time generator setup The bits define the duration of the dead-time inserted between the complementary outputs. Assume that DT represents this duration: DTGS[7: 5] = 0xx => DT = DTGS[7:0] × Tdtg, Tdtg = T _{DTS} . DTGS[7: 5] = 10x => DT = (64+DTGS[5:0]) × Tdtg, Tdtg = 2 × T _{DTS} . DTGS[7: 5] = 110 => DT = (32+DTGS[4:0]) × Tdtg, Tdtg = 8 × T _{DTS} . DTGS[7: 5] = 111 => DT = (32+DTGS[4:0]) × Tdtg, Tdtg = 16 × T _{DTS} . Example: If T _{DTS} = 125 ns (8 MHz), possible dead-time values are: 0 ~ 15875 ns by the steps of 125 ns, 16 us ~ 31750 ns by the steps of 250 ns, 32 us ~ 63 us by the steps of 1 u, 64 us ~ 126 us by the steps of 2 us <i>Note: The bits cannot be modified once LOCK (the LOCKC bits in the TMRx_BRKDT register) level is set as '1', '2', or '3'.</i>

10.4.4.19 TMR1, TMR8, and TMR15 DMA Control Register (TMRx_DMAC)

Address offset: 0x48

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	DBLEN[4:0]				Reserved	ADDR[4:0]				res	rw	rw	rw	rw	rw	
res	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 15:13		Reserved. Always read as 0.														

Bit 12:8	<p>DBLEN[4:0]: DMA burst length The bits define the number of DMA transfers in burst mode (the timer recognizes a burst transfer when a read or a write access is done to the TMRx_DMABA register). Namely, defining the number of transfer, and the transfer can be half-word (two-byte) or byte: 00000: 1 transfer 00001: 2 transfers 00010: 3 transfers 10001: 18 transfers Example: The following transfer is intended: DBLEN = 7, ADDR = TMR2_CTRL1 - If DBLEN = 7, ADDR = TMR2_CTRL1 represents the address to be transferred, the transfer address is obtained from the following formula: (TMRx_CTRL1 address) + ADDR + (DMA index), where DMA index = DBLEN. When (TMRx_CTRL1 address) + ADDR + 7, the address to be written or read out data will be obtained, and data transfer will occur at slave address (TMRx_CTRL1 address) + 7 registers starting from ADDR. According to the configuration of DMA data length, the following conditions may occur: - If data configuration is half-word (16-bit), it will be transferred to all 7 registers. - If data configuration is byte, it will still be transferred to all 7 registers: The first register contains the first MSB byte, the second register contains the first LSB byte, and so on. Therefore, users must define DMA transfer data width for the timer.</p>
Bit 7:5	Reserved. Always read as 0.
Bit 4:0	<p>ADDR[4:0]: DMA base address The bits define the base address for DMA transfers in burst mode (when read/write access is done to the TMRx_DMAR register). ADDR is defined as an offset starting from the address of the TMRx_CTRL1 register. 00000: TMRx_CTRL1, 00001: TMRx_CTRL2, 00010: TMRx_SMC, </p>

10.4.4.20 TMR1, TMR8, and TMR15 DMA Address in Burst Mode (TMRx_DMABA)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMABA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0															
DMABA[15:0]: DMA register for burst accesses A read or write to the TMRx_DMAR register accesses the register located at the following address: TMRx_CTRL1 address + ADDR + DMA index, where: TMRx_CTRL1 address is the address of the control register 1(TMRx_CTRL1). “ADDR” is the base address defined in the TMRx_DMAR register. “DMA index” is the offset automatically controlled by the DMA, and is determined by the DBLEN bit configured in the TMRx_DMAR register.															

11 Watchdog

11.1 Window Watchdog (WWDG)

11.1.1 WWDG Introduction

The window watchdog is normally used to detect software malfunctions generated by external interference or by unforeseen logical conditions, which cause the program to abandon its normal sequence. Unless the values of the downcounter are refreshed before the CNTR6 bit becomes '0', the watchdog circuit generates an MCU reset when reaching the programmed time period. An MCU reset is also generated if the 7-bit downcounter value (in the control register) is refreshed before the downcounter reaches the window register value. This means that the downcounter must be refreshed in a limited window.

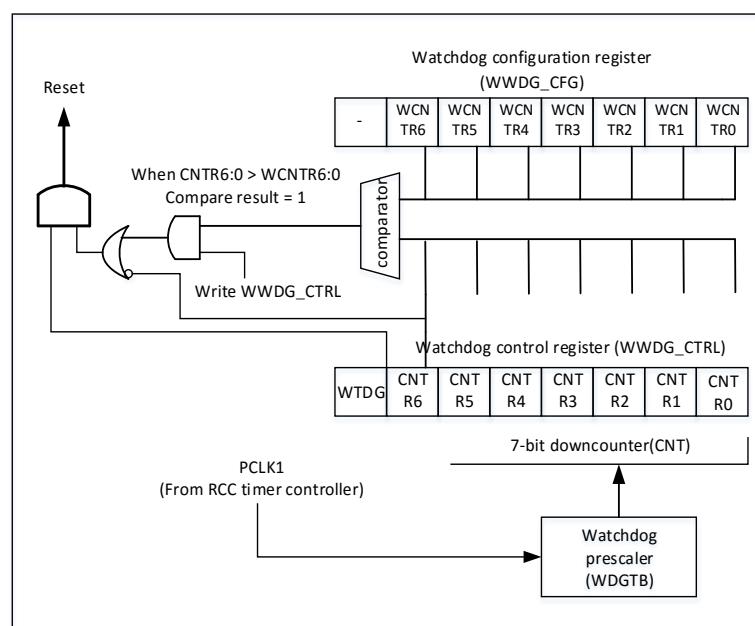
11.1.2 WWDG Main Features

- Programmable free-running downcounter
- Reset under the following conditions:
 - When the downcounter value is less than 0x40 (If the watchdog is enabled.)
 - When the downcounter is reloaded outside the window (if the watchdog is enabled.)
- If the watchdog is activated and the interrupt is enabled, early wakeup interrupt (EWIEN) will be generated when the downcounter reaches 0x40. It can be used to reload the counter to avoid WWDG reset.

11.1.3 WWDG Function Overview

If the watchdog is enabled (the EN bit in the WWDG_CTRL register is set), and when the 7-bit (CNTR[6:0]) downcounter rolls over from 0x40 to 0x3F (the CNTR6 bit is cleared), a reset is generated. If the counter is reloaded when the software value of the counter is greater than the value in the window register, a reset is generated.

Figure 11-1 Watchdog Block Diagram



The program must write in the WWDG_CTRL register regularly during normal operation to prevent an MCU reset. The write operation can be done only when the counter value is less than the window register value. The value stored in the WWDG_CTRL register must fall between 0xFF and 0xC0:

- Enable the watchdog.

After system reset, the watchdog is always disabled. Setting the EN bit in the WWDG_CTRL register can enable the watchdog. Afterwards, it cannot be disabled

unless a reset occurs.

- Control the downcounter.

The downcounter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the CNTR6 bit must be set to prevent an immediate reset. The CNTR[5:0] bits contain the number of counting before the watchdog generates a reset. The delay time varies between the minimum and the maximum value due to the unknown status of the prescaler value when writing to the WWDG_CTRL register. The configuration register (WWDG_CFG) contains the maximum value of the window: To prevent a reset, the downcounter must be reloaded when its value is less than the window register value and greater than 0x3F. Figure 11-2 shows the window register operation. Another way to reload the counter is to use early wakeup interrupt (EWIEN). Set the WEI bit in the WWDG_CFG register to enable the interrupt. When the downcounter reaches 0x40, the interrupt is generated. The corresponding interrupt service program (ISTS) can be used to load the counter to prevent WWDG reset. The interrupt can be cleared by writing '0' in the WWDG_STS register.

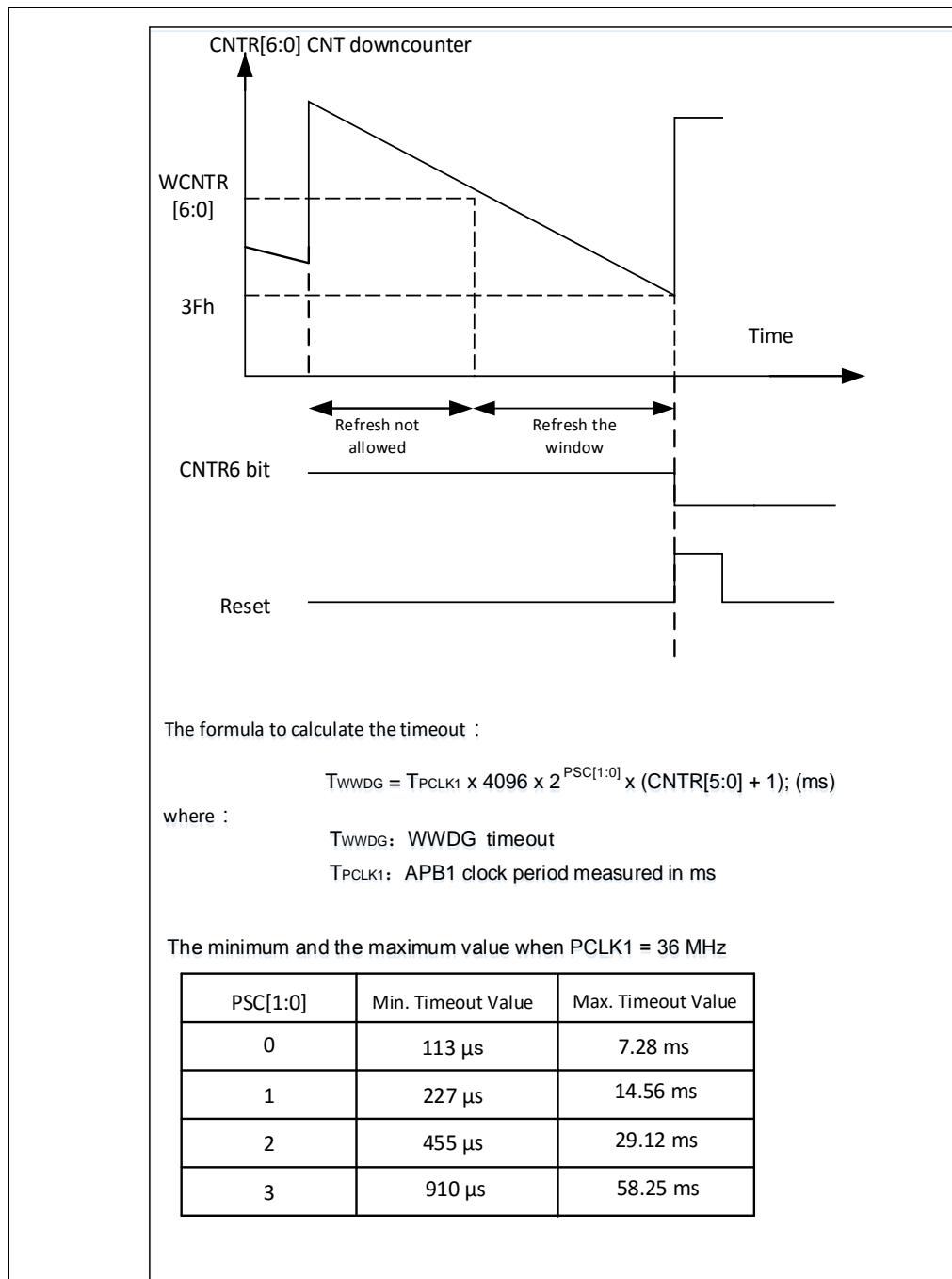
Note: The CNTR6 bit can be used to generate a software reset (Set the EN bit and clear the CNTR6 bit).

11.1.4 How to Program Watchdog Timeout

[Figure 11-2](#) provides the formula to calculate the window watchdog timeout.

Warning: When writing to the WWDG_CTRL register, always set the CNTR6 bit as '1' to avoid an immediate reset.

Figure 11-2 Window Watchdog Timing Diagram



11.1.5 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the WWDG counter may keep working or stop working, based on the status of the DBG_WWDG_STOP configuration bit in the debug module. Please refer to [Section 22.2.2](#) for more details.

11.1.6 Register Description

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

Table 11-1 WWDG Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	WWDG_CTL	Reserved																							EN	CNTR[6:0]							
																										0	1	1	1	1	1	1	1
0x04	WWDG_CFG	Reserved																							EWIEN	WCNTR[6:0]							
																										0	0	0	1	1	1	1	1
0x08	WWDG_STS	Reserved																							EWIF								
																										0							

11.1.6.1 Control Register (WWDG_CTRL)

Address offset: 0x00

Reset value: 0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Reserved																				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Reserved				EN	CNTR[6:0]															
res																rs																				
Bit 31:8																Reserved.																				
Bit 7																EN: Activation bit This bit is set by software, but can only be cleared by hardware after a reset. When EN = 1, the watchdog can generate a reset. 0: The watchdog is disabled. 1: The watchdog is enabled.																				
Bit 6:0																CNTR[6:0]: 7-bit counter (MSB to LSB) These bits are used for storing the counter values of the watchdog. Every (4096x 2^{PSC}) PCLK1 cycle minus 1. When the counter value changes from 40h to 3Fh (CNTR6 becomes '0'), a watchdog reset is generated.																				

11.1.6.2 Configuration Register (WWDG_CFG)

Address offset: 0x04

Reset value: 0x7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16												
Reserved																											
res																											
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved						EWIEN	PSC[1:0]	WCNTR[6:0]																			
res				rs		rw		rw																			
Bit 31:8	Reserved.																										
Bit 9	EWIEN: Early wakeup interrupt If this bit is set, an interrupt will be generated when the counter value reaches 40h. The interrupt can only be cleared by hardware after reset.																										
Bit 8:7	PSC[1:0]: Timer base The time base of the prescaler can be configured as follows: 00: CK Counter Clock (PCLK1 is divided by 4096) is divided by 1. 01: CK Counter Clock (PCLK1 is divided by 4096) is divided by 2. 10: CK Counter Clock (PCLK1 is divided by 4096) is divided by 4. 11: CK Counter Clock (PCLK1 is divided by 4096) is divided by 8.																										
Bit 6:0	WCNTR[6:0]: 7-bit window value These bits contain the window values used to compare with the downcounter.																										

11.1.6.3 Status Register (WWDG_STS)

Address offset: 0x08

Reset value: 0x00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
Reserved																			
res																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
Reserved												EWIF							
res												rs							
Bit 31:1	Reserved.																		
Bit 0	EWIF: Early wakeup interrupt flag This bit is set by hardware when the counter value reaches 40h. It must be cleared by software. Writing '1' to this bit has no effect. This bit is also set if interrupt is not enabled.																		

11.2 Independent Watchdog (IWDG)

11.2.1 Introduction

AT32F403 devices have two embedded watchdogs which provide a higher safety level, timing accuracy, and flexibilities. Both watchdog peripherals (independent watchdog and window watchdog) can detect and resolve malfunctions caused by software failures. When the counter reaches a given timeout value, they can trigger an interrupt (window watchdog only) or system reset.

The independent watchdog (IWDG) is clocked by its own dedicated low-speed clock (LSI) and thus stays active even if the main clock fails. The window watchdog (WWDG) is driven by a clock prescaler from the APB1 clock. It can detect abnormally late or early application behaviors through a configurable time-window.

IWDG best suits the occasions which require the watchdog to run completely independent of the main application, and have lower timing accuracy requirements. WWDG best suits the applications which require the watchdog to react within an accurate timing window.

For further information on the window watchdog, please refer to [Section 11.1](#).

11.2.2 IWDG Main Features

- Free-running downcounter
- Clocked by an independent RC oscillator (Can work in Stop mode and Standby mode)
- Reset generated when the counter counts to 0x000, after the watchdog is activated.

11.2.3 IWDG Function Overview

[Figure 11-3](#) is the block diagram showing the functions of the independent watchdog module. The independent watchdog is enabled by writing 0xCCCC into the key register (IWDG_KEY). At this time, the counter starts downcounting from the reset value, 0xFFFF. When the counter counts to the end, 0x000, a reset signal (IWDG_RESET) is generated.

Anytime when the key register, IWDG_KEY, is written with 0xAAAA, the value in IWDG_RLD will be reloaded to the counter to avoid a watchdog reset.

11.2.3.1 Hardware Watchdog

If users enable the “hardware watchdog” function through option bytes, the watchdog will be automatically enabled after system power-on reset. If the corresponding value is not written to the key register by software before the counting ends, a system reset will be generated.

11.2.3.2 Register Access Protection

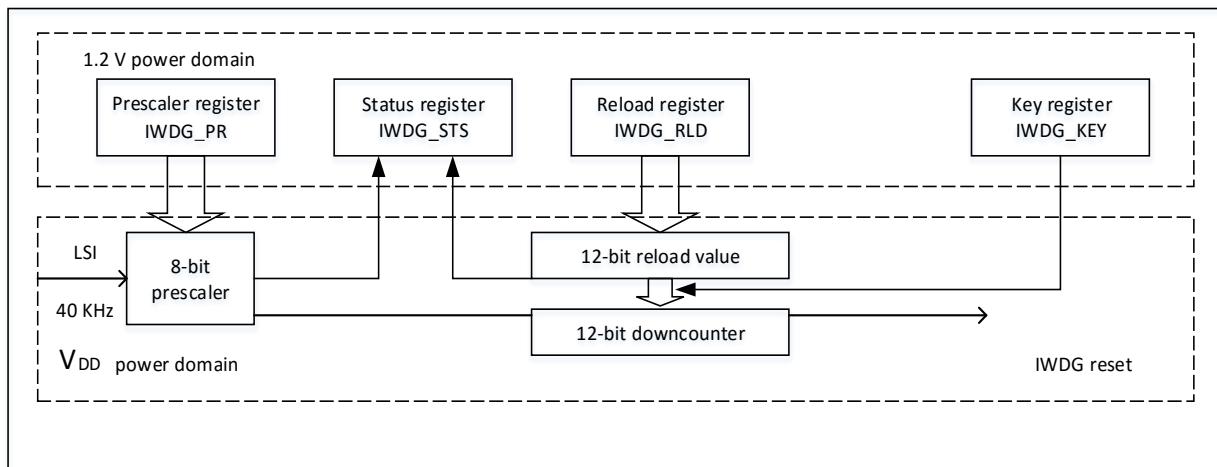
The IWDG_PR and IWDG_RLD registers are write-protected. To modify the values in the two registers, users must write 0x5555 to the IWDG_KEY register. If a different value is written to this register, the sequence will be interrupted, and the register will be re-protected. Reload operation (write 0xAAAA) also enables the write-protection.

The status register indicates the prescaler value and whether the downcounter is being updated.

11.2.3.3 Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the IWDG counter will keep working or stop according to the status of the DBG_IWDG_STOP configuration bit in the debug module. Please refer to the sections about debug module for more details.

Figure 11-3 Block Diagram of Independent Watchdog



Note: The watchdog is implemented in VDD power domain, which means that it can work normally in Stop mode and Standby mode.

Table 11-2 Watchdog Timeout Period (40 kHz Input Clock (LSI)) (Note)

Prescaler Divider	PR[2:0] Bits	Min. Timeout (ms) RLD[11:0] = 0x000	Max. Timeout (ms) RLD[11:0] = 0xFFFF
/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

Note: These timings are given based on the 40 kHz clock. In fact, the frequency of the MCU internal RC still varies between 30 kHz and 60 kHz. In addition, even the frequency of the RC oscillator is accurate, the actual timing still depends on the phase difference between the APB interface clock and the RC oscillator clock. Therefore, there is always an uncertain complete RC cycle.

A relatively accurate watchdog timeout period can be obtained through the calibration to LSI. Please refer to [Section 3.2.5](#) for more details on LSI calibration.

11.2.4 IWDG Register Description

About the abbreviations used on the register descriptions, please refer to [Table 1-4](#). These peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KEY	Reserved															KEY[15:0]																
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x04	IWDG_PR	Reserved																										PR[2:0]					
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x08	IWDG_RLD	Reserved															RLD[11:0]																
	Reset Value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1					
0x08	IWDG_STS	Reserved																										RLDF	PSCF				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

11.2.4.1 Key Register (IWDG_KEY)

Address offset: 0x00

Reset value: 0x0000 0000 (Reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	Reserved													
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	KEY[15:0]													
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:16				Reserved. Always read as 0.																									
Bit 15:0				KEY[15:0]: Key value (Write-only, the value being read is 0x0000.) These bits must be written 0xAAAA by software at regular intervals; otherwise, a watchdog reset will be generated when the counter turns 0. Writing 0x5555 enables accesses to the IWDG_PR and IWDG_RLD registers. (Please refer to Section 11.2.3.2) Writing 0xCCCC enables the watchdog (except the hardware watchdog is selected).																									

11.2.4.2 Prescaler Register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:3		Reserved. Always read as 0.													
Bit 2:0		PR[2:0]: Prescaler divider These bits are write-protected, please refer to Section 11.2.3.2 . Users can select the prescaler divider for the counter clock by setting these bits. To modify the prescaler divider, the PSCF bit in the IWDG_STS register must be 0. 000: Prescaler divider = 4 100: Prescaler divider = 64 001: Prescaler divider = 8 101: Prescaler divider = 128 010: Prescaler divider = 16 110: Prescaler divider = 256 011: Prescaler divider = 32 111: Prescaler divider = 256 Note: Write accesses to this register will return the prescaler value from VDD power domain. If write operation is in process, the value being read may be invalid. Therefore, the value being read is valid only when the PSCF bit in the IWDG_STS register is 0.													

11.2.4.3 Reload Register (IWDG_RLD)

Address offset: 0x08

Reset value: 0x0000 0FFF (Reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	Rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		RLD[11:0]													
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved. Always read as 0.													
Bit 11:0		RLD[11:0]: Watchdog counter reload value These bits are write-protected, please refer to Section 11.2.3.2 . They are used to define the reload value of the watchdog counter. Every time when the IWDG_KEY register is written with 0xAAAA, the reload value will be sent to the counter. Afterwards, the counter starts downcounting from this value. Watchdog timeout period can be calculated through this reload value and the clock prescaler value. Please refer to Table 11-2 . This register can be modified only when the RLDF bit in the IWDG_STS register is 0. Note: Read accesses to this register will return the reload value from VDD power domain. If write operation is in process, the value being read may be invalid. Therefore, the value being read is valid only when the RLDF bit in the IWDG_STS register is 0.													

11.2.4.4 Status Register (IWDG_STS)

Address offset: 0x0C

Reset value: 0x0000 0000 (Not reset by Standby mode)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r
Bit 31:2		Reserved.													
Bit 1		RLDF: Watchdog counter reload value update This bit is set by hardware to indicate that the reload value update is in process. When the reload update is done in VDD domain, this bit is cleared by hardware (five 40 kHz RC cycles are needed at most). The reload value can be updated only after the RLDF bit is cleared.													
Bit 0		PSCF: Watchdog prescaler value update This bit is set by hardware to indicate that the prescaler value update is in process. When the prescaler value update is done in VDD domain, this bit is cleared by hardware (five 40 kHz RC cycles are needed at most). The prescaler value can be updated only after the PSCF bit is cleared.													

Note: If several reload values or prescaler values are used in the application, preload values can be changed only after the RLDF bit is cleared, and the prescaler values can be changed only after the PSCF bit is cleared. However, after prescaler and/or reload value are updated, code operation can be continued without waiting for the RLDF or PSCF reset. (This write operation is still executed and completed even in low-power mode.)

12 Real-time Clock (RTC)

12.1 RTC Introduction

The real-time clock is an independent timer. The RTC module has a set of continuously running counters which can provide a clock-calendar function with relevant software configurations. Modifying the counter values can reset the current time/date of the system.

The RTC module and clock configuration system (the RCC_BDC register) are in Backup domain, which means that the RTC setting and time still remain the same after system reset or wakeup from Standby mode.

After system reset, accesses to the backup registers and RTC is disabled. This is to avoid possible undesired write accesses to Backup domain (BRKP). To enable accesses to the backup registers and RTC, proceed as follows:

- Set the PWREN and BKPN bits in the RCC_APB1EN register to enable power and backup interface clocks.
- Set the DBP bit in the PWR_CTRL register to enable accesses to the backup register and RTC.

12.2 Main Features

- Programmable prescaler ratio: Division factor up to 2^{20}
- 32-bit programmable counter for long-term measurement
- Two separate clocks: PCLK1 for APB1 interface and RTC clock (The frequency of RTC clock must be at least four times slower than the PCLK1 clock.)
- 3 RTC clock sources to choose from:
 - HSE clock divided by 128
 - LSE oscillator clock
 - LSI oscillator clock (Please refer to [Section 3.2.8.](#))
- 2 independent reset types:
 - APB1 interface is reset by system reset.
 - The RTC core (prescaler, alarm, counter, and divider) is reset only by Backup domain reset. (Please refer to [Section 3.1.3.](#))
- 3 dedicated maskable interrupts:
 - Alarm interrupt to generate a software programmable alarm interrupt
 - Second interrupt to generate a programmable periodic interrupt signal (up to 1 second)
 - Overflow interrupt to indicate that the internal programmable counter overflows and rolls over to zero.

12.3 Function Overview

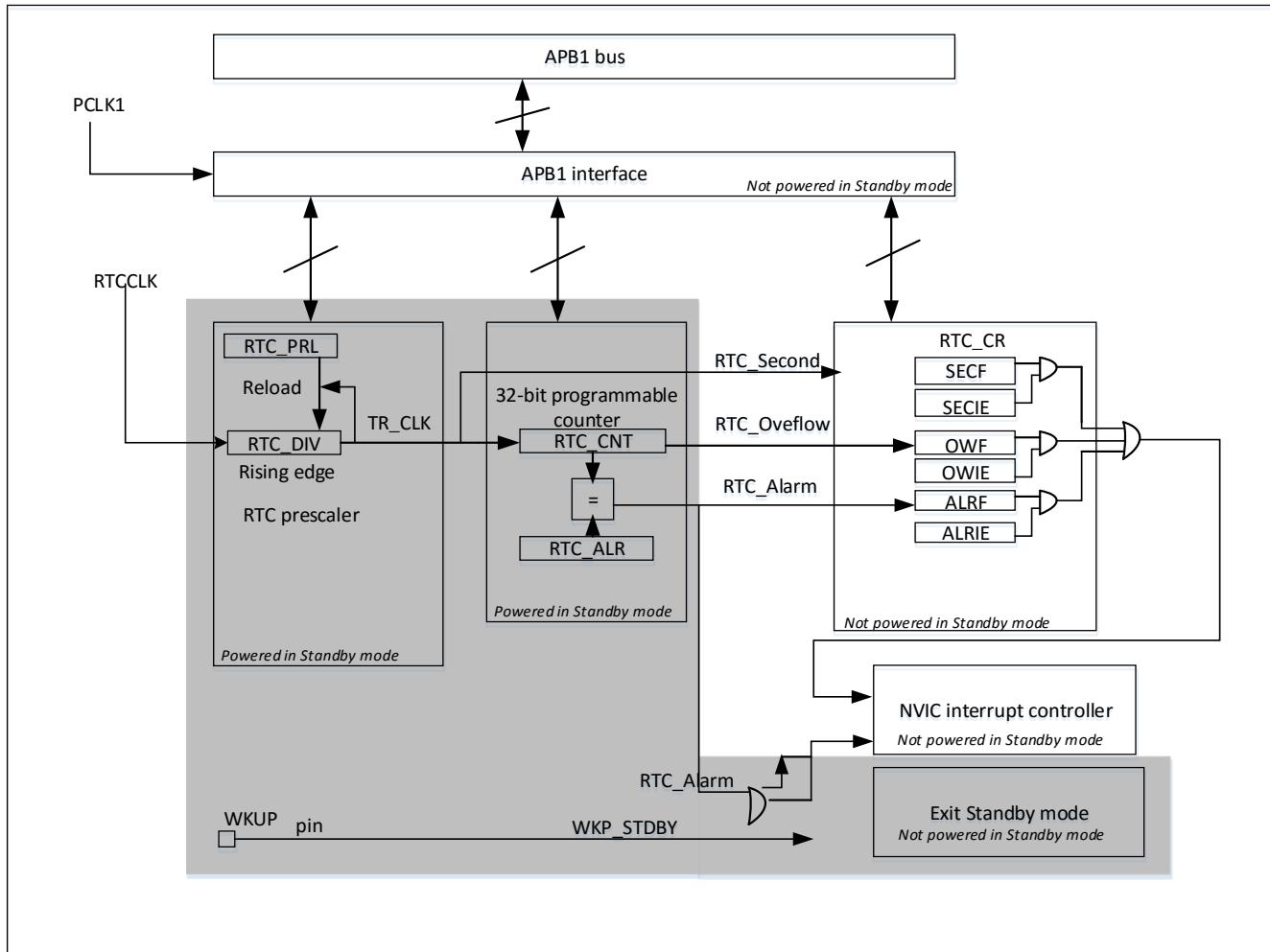
12.3.1 Introduction

RTC consists of two main units (Please refer to Figure 12-1). The first one (APB1 Interface) is used to interface with the APB1 bus. This unit also contains a set of 16-bit registers accessible from the APB1 bus for read or write operation (Please refer to [Section 12.4](#)). The APB1 interface is clocked by the APB1 bus clock in order to interface with the APB1 bus.

The other unit (RTC core) consists of a set of programmable counters divided into two main blocks. The first block is the RTC prescaler block, which can be programmed to generate the RTC time base, LN_CLK, of which length is up to 1 second. RTC prescaler block includes a 20-bit programmable divider (RTC prescaler). If the corresponding enable bits are set in the RTC_CTRL register, RTC generates an interrupt (second interrupt) every LN_CLK period. The second block is a 32-bit programmable counter that can be initialized to the current system time. The system time is incremented by the LN_CLK period

and is compared with the programmable time stored in the ALA register. An alarm interrupt is generated during the comparison and match if the corresponding bits are set in the RTC_CTRL control register.

Figure 12-1 Simplified RTC Block Diagram



12.3.2 Reset Process

All system registers are asynchronously reset by system reset or power reset except for the DIV, ALA, CNT, and DIVCNT registers.

The DIV, ALA, CNT, and DIVCNT registers are reset only by Backup domain reset signal. Please refer to [Section 3.1.3](#) for more details.

12.3.3 Reading RTC Registers

The RTC core is completely independent of the RTC APB1 interface. Software accesses the RTC prescaler value, counter value, and alarm value through the APB1 interface. However, the relevant readable registers are updated only at rising edge of the RTC clock, which is resynchronized by the RTC APB1 clock. This is also true for the RTC flags.

This suggests that, before the first internal update of the registers, the RTC register value read from APB1 may be corrupted (usually read as 0) if the APB1 interface is previously disabled, and the read operation occurs immediately after the APB1 interface is re-activated. The following conditions may lead to this situation:

- A system reset or a power reset occurs.
- The system just wakes up from Standby mode (Please refer to [Section 2.3.2](#)).
- The system just wakes up from Stop mode (Please refer to [Section 2.3.2](#)).

In all the above cases, the RTC core keeps running while the APB1 interface is disabled (reset, not clocked, or not powered). Consequently, when reading the RTC registers, if the RTC APB1 interface is disabled previously, the software must first wait for the RSYNF bit (register synchronized flag) in the

RTC_CTRL register to be set by hardware.

Note: The RTC APB1 interface is not affected by low-power modes such as WFI and WFE.

12.3.4 RTC Register Configuration

To write the DIV, CNT, ALA registers, the CMF bit in the RTC_CTRL register must be set to make RTC enter configuration mode.

In addition, write accesses to any RTC register must be proceeded after the previous write operation is completed. Users can check the RTF status bit in the RTC_CTRL register to see if the RTC register is being updated. A new value can be written to the RTC registers only when the value of the RTF status bit is '1'.

The configuration procedure is as follows:

1. Poll the RTF bit and wait until the RTF value becomes '1'.
2. Set the CMF bit to enter configuration mode.
3. Write to one or more RTC registers.
4. Clear the CMF flag bit to exit configuration mode
5. Poll the RTF bit and wait until the RTF value becomes '1' to ensure that the write operation is completed. Write operations are proceeded only when the CMF flag bit is cleared. It takes at least three RTCCLK cycles.

12.3.5 RTC Flag Assertion

In each RTC core clock cycle, the RTC second flag (PACEF) is asserted before the RTC counter is updated. The RTC overflow flag (OVF) is asserted in the last RTC clock cycle before the counter reaches 0x0000. The RTC_Alarm and RTC alarm flag (ALAF) are asserted in the last RTC clock cycle before the counter reaches the alarm register value + 1 (ALA + 1). Write accesses to the RTC alarm must be synchronized with the RTC second flag by using one of the following sequences:

- Use the RTC alarm interrupt and update the RTC alarm and/or the RTC counter in the interrupt routine.
- Wait for the PACEF bit to be set in the RTC control register, and then update the RTC alarm and/or the RTC counter.

Figure 12-2 Example of RTC Second and Alarm Waveform, PR = 0003, ALARM = 00004

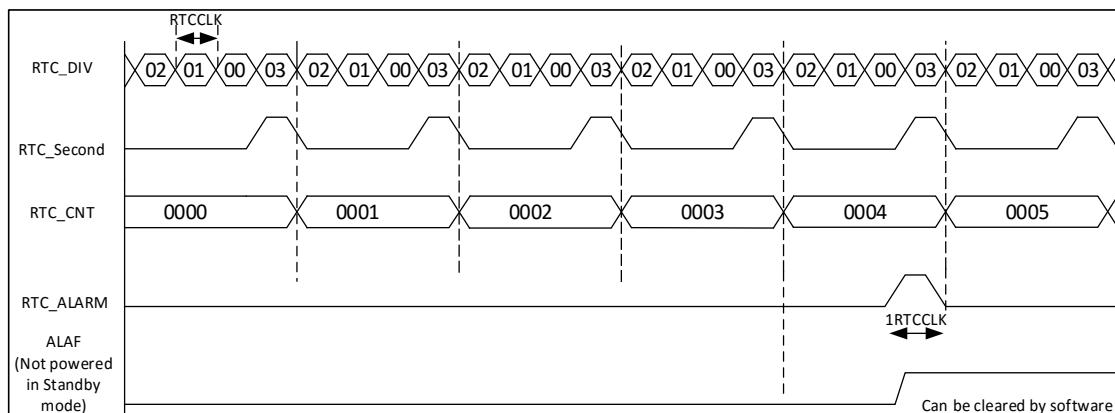
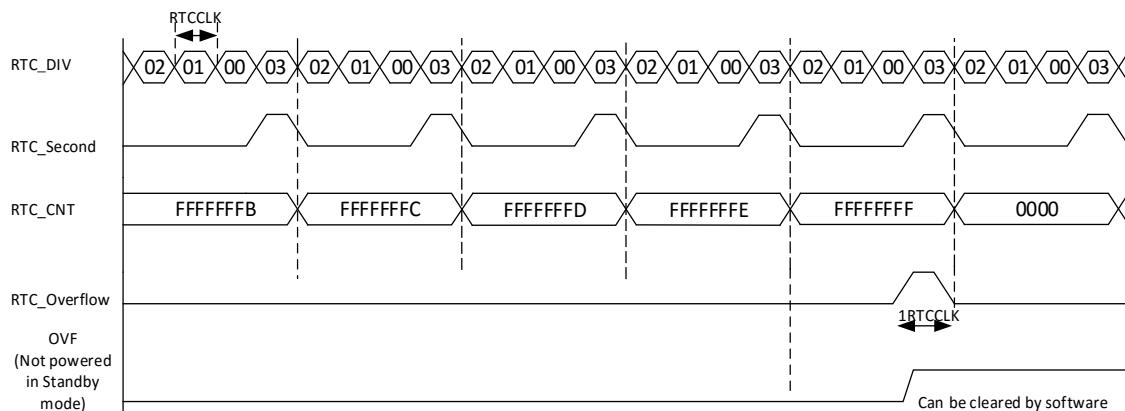


Figure 12-3 Example of RTC Overflow Waveform, PR = 0003



12.4 RTC Register Description

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

The RTC register is a 16-bit addressable register, and the detailed descriptions are as follows:

Table 12-1 RTC- Register Map and Reset Values

Offset	register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
000h	RTC_CTRLH	Reserved																									OVIEN	ALAIEN	PACEIEN						
		Reset Value																																	
004h	RTC_CTRLR	Reserved																								RTF	CMF	RSYNF							
		Reset Value																																	
008h	RTC_DIVH	Reserved																									DIV[19:16]								
		Reset Value																																	
00Ch	RTC_DIVL	Reserved												DIV[15:0]																					
		Reset Value												0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																					
010h	RTC_DIVC_NTH	Reserved												DIVCNT[31:16]																					
		Reset Value												0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																					
014h	RTC_DIVC_NTL	Reserved												DIVCNT [15:0]																					
		Reset Value												0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																					
018h	RTC_CNTH	Reserved												CNT[31:16]																					
		Reset Value												0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																					
01Ch	RTC_CNTL	Reserved												CNT[15:0]																					
		Reset Value												0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0																					
020h	RTC_ALAH	Reserved												ALA[31:16]																					
		Reset Value												1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1																					
024h	RTC_ALAL	Reserved												ALA[15:0]																					
		Reset Value												1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1																					

12.4.1 RTC Control Register High (RTC_CTRLH)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
														OVien	ALAIEN	PACEIEN
								res						rW	rW	rW
Bit 15:3																
Bit 2	OVien : Overflow interrupt enable 0: Overflow interrupt is masked (disabled). 1: Overflow interrupt is enabled.															
Bit 1	ALAIEN : Alarm interrupt enable 0: Alarm interrupt is masked (disabled). 1: Alarm interrupt is enabled.															
Bit 0	PACEIEN : Second interrupt enable 0: Second interrupt is masked (disabled). 1: Second interrupt is enabled.															

These bits are used to mask interrupt requests.

Note: All interrupts are masked after a system reset, so writing to the RTC register can ensure that there is no pending interrupt request after initialization. Write accesses to the RTC_CTRLH register are not allowed when the peripheral is processing the previous write access (The flag bit, RTF, = 0).
RTC function is controlled by this control register. Some write accesses to the bits must be completed through a specific configuration process (Please refer to [Section 12.3.4](#)).

12.4.2 RTC Control Register Low (RTC_CTRLL)

Address offset: 0x04

Reset value: 0x0020

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
														RTF	CMF	RSYNF	OVF	ALAF	PACEF
								res						r	rW	rc w0	rc w0	rc w0	rc w0
Bit 15:6																			
Bit 5	RTF : RTC operation OFF The RTC block uses this bit to indicate the status of the last access done to its register, showing whether the access is completed. If the bit is '0', no write access to any of the RTC registers is allowed. This bit is read-only. 0: The last write access to the RTC register is in process. 1: The last write access to the RTC register is completed.																		
Bit 4	CMF : Configuration flag This bit must be set by software to enter configuration mode, and then allows data to be written to the CNT , ALA , or DIVCNT registers. Write accesses are proceeded only when this bit is set and cleared by software again. 0: Exit configuration mode (Start to update the RTC register) 1: Enter configuration mode																		

Bit 3	RSYNF: Registers synchronized flag This bit is set by hardware every time when the CNT register and the DIVCNT register are updated or cleared by software. After APB1 reset or APB1 clock stops, this bit must be cleared by software. Before beginning any read operation, users must wait until this bit is set by hardware, to ensure that CNT, ALA, or DIVCNT are already synchronized. 0: Registers are not yet synchronized. 1: Registers are already synchronized.
Bit 2	OVF: Overflow flag This bit is set by hardware at 32-bit programmable counter overflow. If OVIEN = 1 in the RTC_CTRLH register, the interrupt is generated. This bit can be cleared only by software. Writing '1' to this bit has no effect. 0: No overflow 1: 32-bit programmable counter overflow occurs.
Bit 1	ALAF: Alarm flag When the 32-bit programmable counter reaches the threshold value programmed by the ALA register, this bit is set by hardware. If ALAIEN = 1 in the RTC_CTRLH register, the interrupt is generated. This bit can be cleared only by software. Writing '1' to this bit has no effect. 0: Alarm is not detected. 1: Alarm is detected.
Bit 0	PACEF: Second flag When the 32-bit programmable prescaler overflows, this bit is set by hardware; at the same time, the RTC counter + 1. Therefore, this flag provides a periodic signal (usually 1 second) for the resolution-programmable RTC counter. If PACEIEN = 1 in the RTC_CTRLH register, the interrupt is generated. This bit can be cleared only by software. Writing '1' to this bit has no effect. 0: Second flag condition is not met. 1: Second flag condition is met.

RTC function is controlled by this control register. The RTC_CTRL register cannot be written if the last write operation is not completed yet (Please refer to [Section 12.3.4](#)) for more details.

- Note:**
1. All flag bits remain pending until the corresponding RTC_CTRL request bits are reset by software, which means that all the interrupt requests are accepted.
 2. All interrupts are disabled at reset; the RTC register can be written if there is no pending interrupt request.
 3. When APB1 clock is not running, the OVF, ALAF, PACEF, and RSYNF bits are not updated.
 4. The OVF, ALAF, PACEF, and RSYNF bits can only be set by hardware and be cleared by software.
 5. If ALAF = 1 and ALAIEN = 1, the RTC global interrupt is enabled. If EXTI line 17 interrupt is enabled in the EXTI controller, RTC global interrupt and RTC alarm interrupt are enabled.
 6. If ALAF = 1, the RTC alarm interrupt is enabled if EXTI line 17 interrupt mode is set in the EXTI controller. When the EXTI line 17 event mode is set in the EXTI controller, a pulse is generated on this line (no RTC alarm interrupt is generated).

12.4.3 RTC Prescaler Load Register (RTC_DIVH/RTC_DIVL)

The prescaler load registers can store the counting value of RTC prescaler period. They are protected by the RTF bit in the RTC_CTRL register. Write accesses are allowed only when the RTF bit is '1'.

RTC prescaler load register high (RTC_DIVH)

Address offset: 0x08

Write-only (Please refer to [Section 12.3.4](#).)

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										DIV[19:16]					
res											W	W	W	W	W
Bit 15:4	Reserved														
Bit 3:0	DIV[19:16]: RTC prescaler reload value high These bits are used to define the frequency of the counter clock based on the following formula: $f_{LN_CLK} = f_{RTCCLK}/(DIV[19:0] + 1)$														

RTC prescaler load register low (RTC_DIVL)

Address offset: 0x0C

Write-only (Please refer to [Section 12.3.4.](#))

Reset value: 0x8000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV[15:0]															
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W
Bit 15:0	DIV[15:0]: RTC prescaler reload value low These bits are used to define the frequency of the counter clock based on the following formula: $f_{LN_CLK} = f_{RTCCLK}/(DIV[19:0] + 1)$ Note: Using 0 is not recommended, or the correct RTC interrupt and flag cannot be generated.														

Note: If the frequency of the input clock (f_{RTCCLK}) is 32.768 kHz, writing 7FFFh to this register obtains a signal of which period is 1 second.

12.4.4 RTC Prescaler Divider Register (RTC_DIVCNTH/RTC_DIVCNTL)

In every LN_CLK cycle, the counter value in the RTC prescaler is reset as the value of the DIVCNT register. Users can obtain the current value of the prescaler counter by reading the DIVCNT register without stopping the divider counter, thus gaining accurate time measurement. This register is read-only, and its value is reload by hardware after the values in the DIVCNT or CNT register change.

RTC prescaler divider register high (RTC_DIVCNTH)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										DIVCNT[19:16]					
res											r	r	r	r	r
Bit 15:4	Reserved														
Bit 3:0	DIVCNT[19:16]: RTC clock divider high														

RTC prescaler divider register low (RTC_DIVCNTL)

Address offset: 0x14

Reset value: 0x8000

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

12.4.5 RTC Counter Register (RTC_CNTH/RTC_CNTL)

The RTC core has a 32-bit programmable counter which can be accessed through two 16-bit registers; the count counts based on the LN_CLK time reference generated by the prescaler. The CNT registers store the counting value of the counter. They are write-protected by the RTF bit in the RTC_CTRL register. A write operation is allowed only if the RTF value is '1'. A write operation on the high (RTC_CNTH) or low (RTC_CNTL) registers directly loads the corresponding programmable counter and reloads the RTC prescaler. When reading the registers, the current counting value in the counter (system time) is returned.

RTC counter register high (RTC_CNTH)

Address offset: 0x18

Reset value: 0x0000

RTC counter register low (RTC_CNTL)

Address offset: 0x1C

Reset value: 0x0000

12.4.6 RTC Alarm Register (RTC_ALAH/RTC_ALAL)

When the value of the programmable counter is equal to the 32-bit value in ALA, an alarm event is triggered, and RTC alarm interrupt is generated. This register is write-protected by the RTF bit in the RTC_CTRL register. A write operation is allowed only if the RTF value is '1'.

RTC alarm register high (RTC_ALAH)

Address offset: 0x20

Write-only (Please refer to [Section 12.3.4.](#))

Reset value: 0xFFFF

Bit 15:0	ALA[31:16]: RTC alarm high This register stores the high part of the alarm time written by software. Before writing to this register, it is necessary to enter configuration mode (Please refer to Section 12.3.4).
----------	--

RTC alarm register low (RTC_ALAL)

Address offset: 0x24

Write-only (Please refer to [Section 12.3.4](#).)

Reset value: 0xFFFF

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

Bit 15:0

ALA[15:0]: RTC alarm lowThis register stores the low part of the alarm time written by software. Before writing to this register, it is necessary to enter configuration mode (Please refer to [Section 12.3.4](#)).

13 Analog-to-Digital Converter (ADC)

13.1 ADC Introduction

The 12-bit ADC is a successive approximation analog-to-digital converter. It has 23 channels to measure the 21 external and 2 internal signal sources. The analog signal of each channel can be converted by the ADC in single, continuous, scan, or discontinuous mode. ADC output can be stored in the 16-bit data register in a left-aligned or right-aligned way.

The analog watchdog allows the application to detect whether the input voltage exceeds the user-defined high/low thresholds.

An on-chip hardware oversample scheme improves performances while off-loading related computational burden from the MCU.

The ADC input clock is generated from the PCLK2 clock divided by a prescaler, and it cannot exceed 28 MHz. Please refer to [Figure 3-2](#).

13.2 ADC Main Features

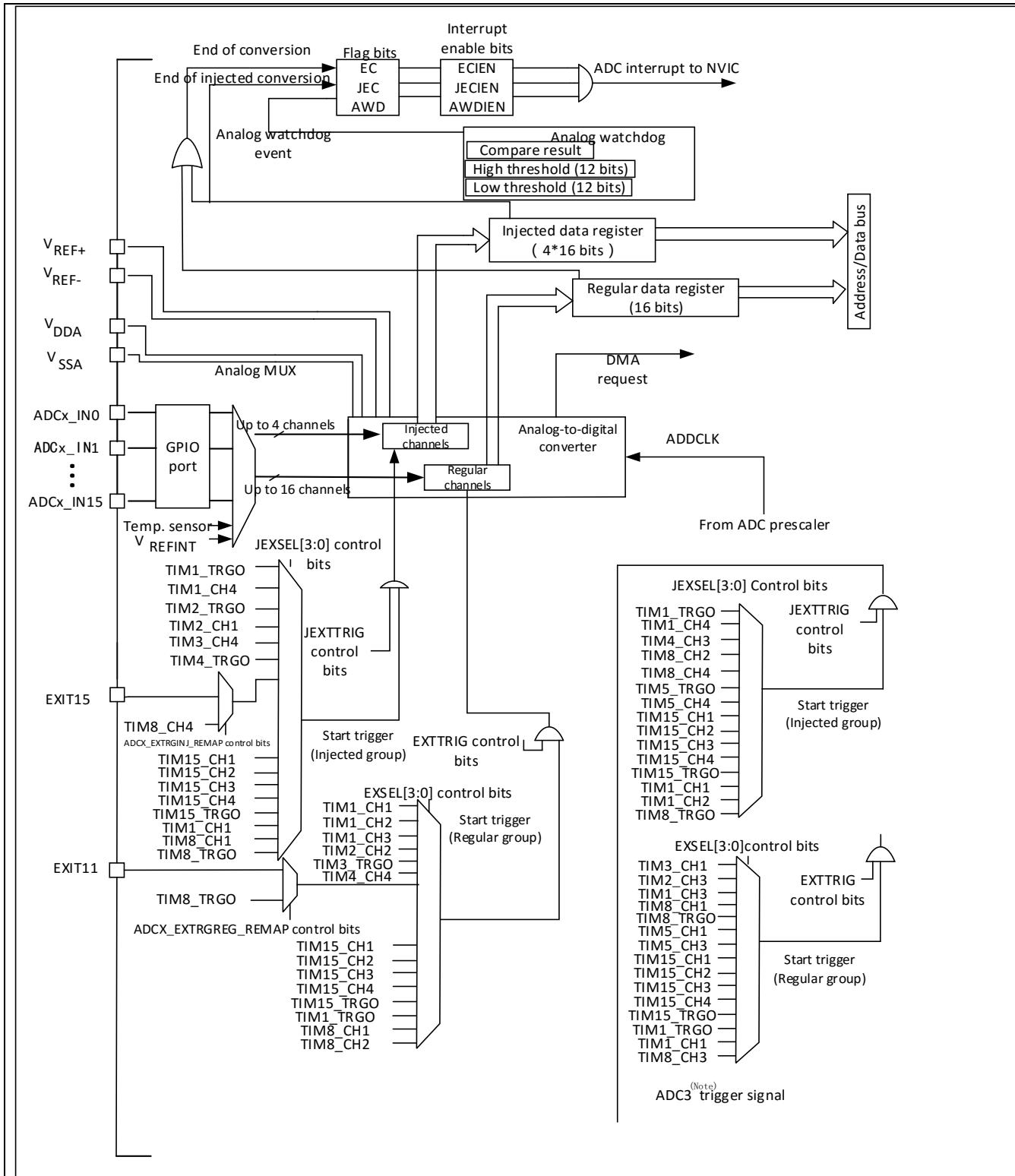
- 12-bit resolution
- Interrupt generation at the end of conversion, the end of injected conversion, and analog watchdog event
- Single and continuous conversion modes
- Automatic scan mode from channel 0 to channel n
- Self-calibration: 312 ADC clock cycles
- Data alignment with built-in data coherency
- Channel by channel programmable sampling interval
- External trigger option for both regular and injected conversion
- Discontinuous mode
- Dual mode (on devices with 2 ADCs or more)
- ADC conversion time
 - 0.5 μ s at 28 MHz clock (0.56 μ s at 200 MHz clock)
- ADC supply requirement: 2.6 V ~ 3.6 V
- ADC range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$
- DMA request generation at regular channel conversion

Note: If V_{REF-} pin is available (depends on the package), it must be tied to V_{SSA} .

13.3 ADC Function Overview

Figure 13-1 is the block diagram of an ADC module, and [Table 13-1](#) lists the description of ADC pins.

Figure 13-1 Single ADC Block Diagram



Note: The regular conversion and injected conversion trigger of ADC3 are different from those of ADC1 and ADC2.

Table 13-1 ADC Pins

Name	Signal Type	Description
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the ADC, $2.6 \text{ V} \leq V_{REF+} \leq V_{DDA}$
V_{DDA} ^(Note)	Input, analog supply	Analog power supply equal to V_{DD} and: $2.6 \text{ V} \leq V_{DDA} \leq V_{DD}$ (3.6 V)
V_{REF-}	Input, analog reference negative	The lower/negative reference voltage for the ADC, $V_{REF-} = V_{SSA}$
V_{SSA} ^(Note)	Input, analog supply ground	Ground for analog power supply equal to V_{SS}
ADCx_IN[15:0]	Analog input signal	21 analog input channels

Note: V_{DDA} and V_{SSA} have to be connected to V_{DD} and V_{SS} respectively.

13.3.1 ADC Switch

Setting the ADON bit in the ADC_CTRL2 register can activate the ADC. When setting the ADON bit for the first time, it will wake up the ADC from power-down mode.

Conversion starts when ADON bit is set for the second time after ADC power-on delay time (t_{STAB}).

Clearing the ADON bit can stop conversion and put the ADC in power-down mode. In this mode, the ADC hardly consumes power (only a few μA).

13.3.2 ADC Clock

The ADCCLK clock provided by the clock controller is synchronous with PCLK2 (APB2 clock). The RCU controller provides a dedicated programmable prescaler for the ADC clock. Please refer to the section of reset and clock control (RCC).

13.3.3 Channel Selection

There are 21 multiplexed channels. Conversions can be categorized into two groups: regular and injected. A group conversion consists of a sequence of conversions which are done on any channels in any orders. For instance, conversion can be done in the following order: channel 3, channel 8, channel 2, channel 2, channel 0, channel 2, channel 2, channel 15.

- **Regular group** is composed of up to 16 conversions. The regular channels and their conversion sequences must be selected in the ADC_RSQx registers. The total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_RSQ1 register.
- **Injected group** is composed of up to 4 conversions. The injected channels and their conversion sequences must be selected in the ADC_JSQ register. The total number of conversions in the injected group must be written in the LEN[1:0] bits in the ADC_JSQ register.

If the ADC_RSQx register or the ADC_JSQ register is modified during a conversion, the current conversion is reset, and a new start pulse will be sent to the ADC to convert the new selected group.

Temperature sensor/VREFINT internal channels

The temperature sensor is connected to channel ADC1_IN16, and the internal reference voltage V_{REFINT} is connected to ADC1_IN17. These two internal channels can be selected and converted as injected or regular channels.

Note: The temperature sensor and V_{REFINT} are only available on the master ADC1 peripheral.

13.3.4 Single Conversion Mode

In single conversion mode, the ADC performs only one conversion. This mode is started either by setting the ADON bit in the ADC_CTRL2 register (for a regular channel only) or by external trigger (for both regular and injected channels), the CON bit is 0 in the meantime.

Once the conversion of the selected channel is completed:

- If a regular channel is converted:
 - The converted data is stored in the 16-bit ADC_RDOR register.

- The EC (End of Conversion) flag is set.
- An interrupt is generated if the ECIEN bit is set.
- If an injected channel is converted:
 - The converted data is stored in the 16-bit ADC_JDOR1 register.
 - The JEC (End of Conversion Injected) flag is set.
 - An interrupt is generated if the JECIEN bit is set.

Then, the ADC stops.

13.3.5 Continuous Conversion Mode

In continuous conversion mode, the ADC starts another conversion as soon as it finishes the previous one. This mode is started either by external trigger or by setting the ADON bit in the ADC_CTRL2 register. The CON bit is 1 in the meantime.

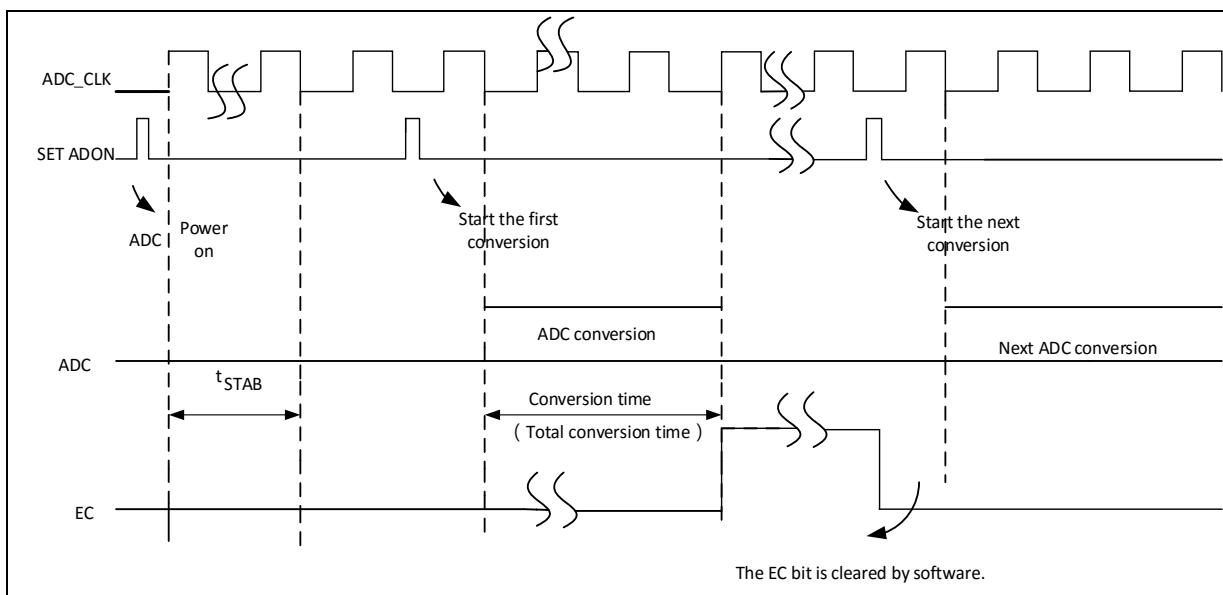
After each conversion:

- If a regular channel is converted:
 - The converted data is stored in the 16-bit ADC_RDOR register.
 - The EC (End of Conversion) flag is set.
 - An interrupt is generated if the ECIEN bit is set.
- If an injected channel is converted:
 - The converted data is stored in the 16-bit ADC_JDOR1 register.
 - The JEC (End of Conversion Injected) flag is set.
 - An interrupt is generated if the JECIEN bit is set.

13.3.6 Timing Diagram

As shown in Figure 13-2, the ADC needs a stabilization time, t_{STAB} , before it starts converting accurately. After the start of ADC conversion and 14 clock cycles, the EC flag is set, and the 16-bit ADC data register contains the result of the conversion.

Figure 13-2 Timing Diagram



13.3.7 Analog Watchdog

The AWD analog watchdog status bit is set if the analog voltage converted by the ADC is below the low threshold or above the high threshold. These thresholds are programmed in the 12 least significant bits of the ADC_WHTR and ADC_WLTR registers. The corresponding interrupt can be enabled by setting the AWDIEN bit in the ADC_CTRL1 register.

The threshold value is independent of the alignment selected by the DALIGN bit in the ADC_CTRL2

register. The comparison is done before the alignment (Please refer to [Section 13.3.12](#)). The analog watchdog can be enabled on one or more channels by configuring the ADC_CTRL1 register as shown in [Table 13-2](#).

Figure 13-3 Analog Watchdog Guarded Area

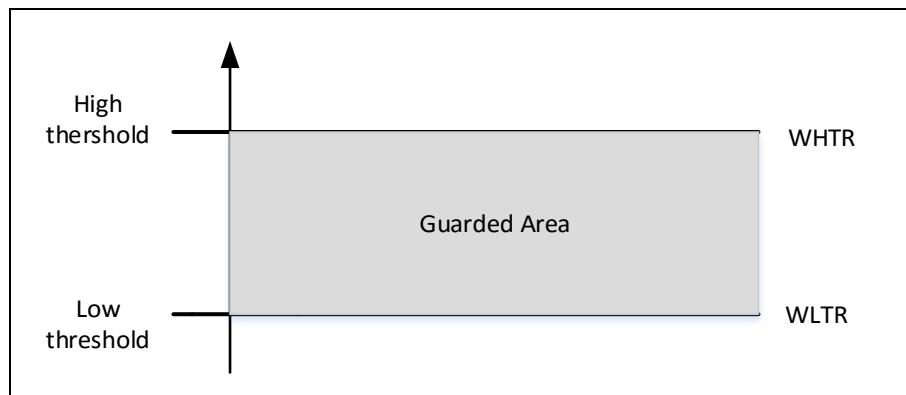


Table 13-2 Analog Watchdog Channel Selection

Channels to be guarded by analog watchdog	ADC_CTRL1 Register Control Bits		
	AWDSGE bit	AWDEN bit	JAWDEN bit
None	Any value	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single ^(Note) injected channel	1	0	1
Single ^(Note) regular channel	1	1	0
Single ^(Note) regular or injected channel	1	1	1

Note: Selected by the AWDCS [4:0] bits.

13.3.8 Scan Mode

This mode is used to scan a group of analog channels.

Scan mode can be selected by setting the SCN bit in the ADC_CTRL1 register. Once this bit is set, the ADC scans all the channels selected by the ADC_RSQX registers (for regular channels) or in the ADC_JSQ (for injected channels). A single conversion is performed on each channel of each group. After each conversion ends, the next channel of the same group is converted automatically. If the CON bit is set, conversion does not stop at the last channel in the selected group, but continues again from the first channel in the selected group.

If the DMAEN bit is set, and the DMA controller transfers the converted data of regular group channels to SRAM after each DOR register update. On the other hand, the converted data of injected channel is always stored in the ADC_JDORx register.

13.3.9 Injected Channel Management

Triggered injection

Triggered injection is enabled by clearing the JAUT bit in the ADC_CTRL1 register and setting the SCN bit.

1. Start the conversion of a group of regular channels either by external trigger or by setting the ADON bit in the ADC_CTRL2 register.
2. If an external injected trigger occurs during the regular group channel conversion, the current conversion is reset, and the injected channel sequence is converted in single scan mode.

3. Then, the regular group channel conversion resumes from the last interruption. If a regular event occurs during an injected conversion, the injected conversion will not be interrupted, and the regular sequence will be performed at the end of the injected sequence. [Figure 13-4](#) shows the timing diagram.

Note: When using triggered injection, the interval between trigger events must be longer than the injection sequence. For instance, if the sequence length is 28 ADC clock cycles (that is, two conversions with a 1.5 clock-period sampling time), the minimum interval between triggers must be 29 ADC clock cycles.

Auto-injection

If the JAUT bit is set, the injected group channels are automatically converted after the regular group channels. This can be used to convert up to 20 conversion sequences programmed in the ADC_RSQx and ADC_JSQ registers.

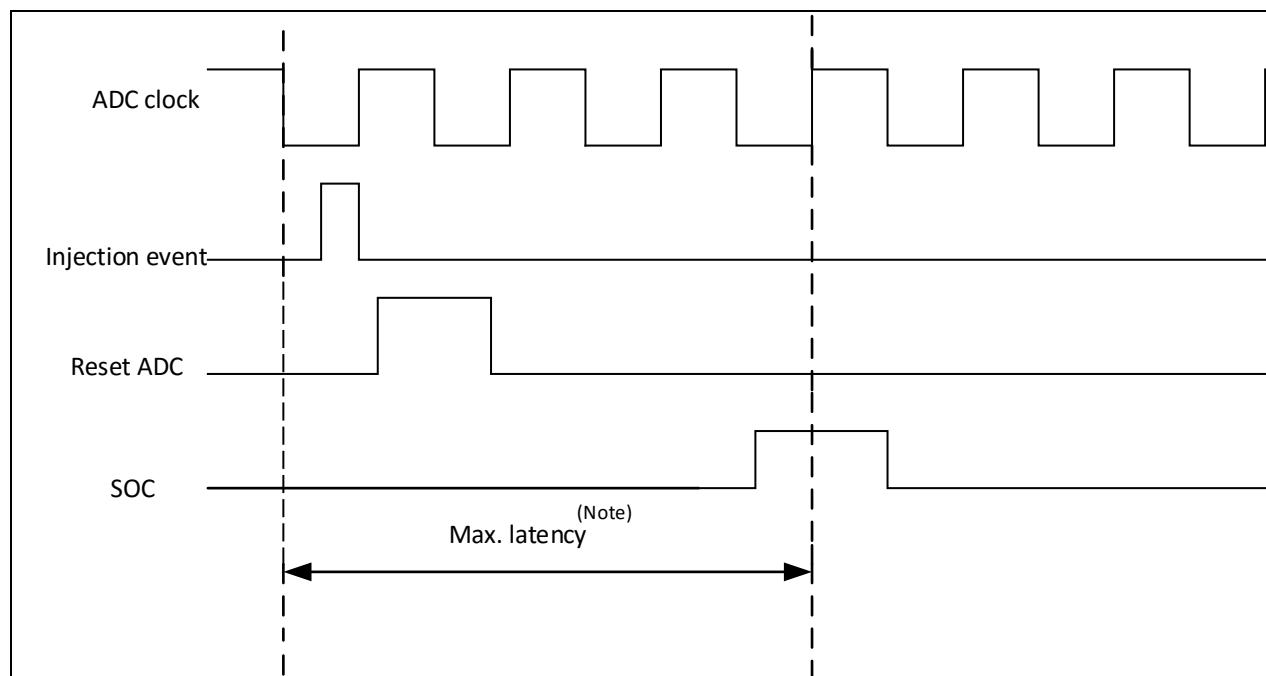
In this mode, external trigger on injected channels must be disabled.

If the CON bit is also set in addition to the JAUT bit, sequences of regular channels to injected channels are continuously converted.

For ADC clock prescaler ratios ranging from 4 to 8, an interval of 1 ADC clock period is automatically inserted when switching from regular conversion to injected sequence, and vice versa. When the ADC clock prescaler ratio is set to 2, the delay is 2 ADC clock periods.

Note: It is not possible to use both auto-injected and discontinuous modes at the same time.

Figure 13-4 Injected Conversion Latency



Note: The maximum latency value can be found in the section of electrical characteristics in the AT32F403 data sheets.

13.3.10 Discontinuous Mode

Regular group

This mode is enabled by setting the RDISEN bit in the ADC_CTRL1 register. It can be used to convert n ($n \leq 8$) conversions of a short sequence, and this conversion is a part of the conversion sequence selected by the ADC_RSQx registers. The value, n, is specified by the DISN[2:0] bits in the ADC_CTRL1 register.

An external trigger signal starts the next n conversions programmed in the ADC_RSQx register, until all the conversions in the sequence are done. The total sequence length is defined by the LEN [3:0] bits in the ADC_RSQ1 register.

Example:

$n = 3$, channels to be converted = 0, 1, 2, 3, 6, 7, 9, 10

First trigger: The converted sequence are 0, 1, and 2. An EC event is generated.

Second trigger: The converted sequence are 3, 6, and 7. An EC event is generated.

Third trigger: The converted sequence are 9 and 10. An EC event is generated.

Fourth trigger: The converted sequence are 0, 1, and 2. An EC event is generated.

Note: *When a regular group is converted in discontinuous mode, no rollover will occur.*

When all the sub groups are converted, the next trigger starts the conversion of the first sub-group. In the above example, the fourth trigger reconverts the first sub-group channels 0, 1, and 2.

Injected group

This mode is enabled by setting the JDISEN bit in the ADC_CTRL1 register. It can be used to convert the sequence selected in the ADC_JSQ register, channel by channel, after an external trigger event.

An external trigger signal starts the next channel sequence conversion selected in the ADC_JSQ register, until all the conversions in the sequence are done. The total sequence length is defined by the JLEN [1:0] bits in the ADC_JSQ register.

Example:

$n = 1$, channels to be converted = 1, 2, 3

First trigger: Channel 1 is converted.

Second trigger: Channel 2 is converted.

Third trigger: Channel 3 is converted, and EC and JEC events are generated.

Fourth trigger: Channel 1 is converted.

Note: 1. *When all the injected channels are converted, the next trigger starts the conversion of the first injected channel. In the above example, the fourth trigger reconverts the first injected channel 1.*

2. *It is not possible to use both auto-injected and discontinuous modes at the same time.*

3. *Users should avoid setting discontinuous mode for both regular and injected groups at the same time. Discontinuous mode must be enabled only for one group conversion.*

13.3.11 Calibration

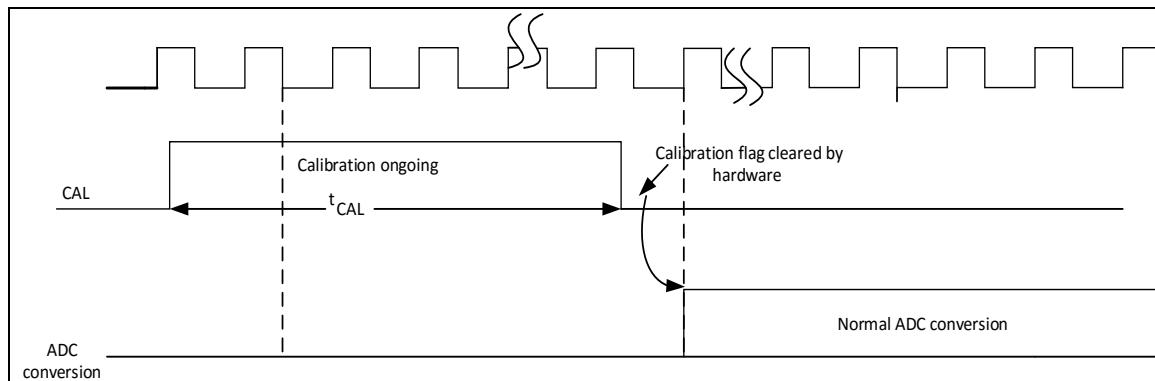
The ADC has a built-in self-calibration mode. Calibration significantly reduces accuracy errors resulted from internal capacitor bank variations. During calibration, an error-correction code (digital value) is calculated for each capacitor, and this code is used to remove the errors generated on each capacitor during the subsequent conversions.

Calibration is enabled by setting the CAL bit in the ADC_CTRL2 register. Once calibration is done, the CAL bit is reset by hardware, and conversion can be performed normally. It is recommended that users calibrate the ADC once at power-on. The calibration codes are stored in the ADC_RDOR as soon as the calibration phase ends.

Note: 1. *It is recommended that users perform a calibration after each power-up.*

2. *Before starting a calibration, the ADC must be at power-on state (ADON bit = '1') for at least two ADC clock cycles.*

Figure 13-5 Calibration Timing Diagram



13.3.12 Data Alignment

The DALIGN bit in the ADC_CTRL2 register selects the alignment of data stored after conversion. Data can be left-aligned or right-aligned, as shown in [Figure 13-6](#) and [Figure 13-7](#).

The converted data value of injected group channels is decreased by the offset defined in the ADC_JOFSx registers, so the result can be a negative value. The SEXT bit is the extended sign value.

For regular group channels, offset does not need to be subtracted, so only twelve bits are significant.

Figure 13-6 Right-alignment of Data

Injected group

SEXT	SEXT	SEXT	SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
------	------	------	------	-----	-----	----	----	----	----	----	----	----	----	----	----

Regular group

0	0	0	0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
---	---	---	---	-----	-----	----	----	----	----	----	----	----	----	----	----

Figure 13-7 Left-alignment of Data

Injected group

SEXT	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0
------	-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---

Regular group

D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	0	0	0	0
-----	-----	----	----	----	----	----	----	----	----	----	----	---	---	---	---

13.3.13 Programmable Channel Sample Time

The ADC samples the input voltage by using several ADC_CLK cycles. The sampling cycle number can be modified through the SMP[2:0] bits in the ADC_SMPT1 and ADC_SMPT2 registers. Each channel can be sampled with a different sample time.

The total conversion time is calculated as follows:

$$T_{CONV} = \text{Sampling time} + 12.5 \text{ cycles}$$

For example:

With an ADCCLK = 14 MHz and a sampling time of 1.5 cycles:

$$T_{CONV} = 1.5 + 12.5 = 14 \text{ cycles} = 1 \mu\text{s}$$

13.3.14 Conversion on External Trigger

Conversion can be triggered by an external event (e.g. timer capture, EXTI line). If the EXTRIG control bit is set, then external events can trigger a conversion. The EXSEL[3:0] and JEXSEL[3:0] control bits allow the application to select one event out of the 16 possible events to trigger conversion for the regular and injected groups.

Note: When an external trigger signal is selected for ADC regular or injected conversion, only its rising edge can start the conversion.

Table 13-3 ADC1 and ADC2 Used in External Trigger for Regular Channels

Source	Type	EXSEL[3:0]
TMR1_CC1 event	Internal signal from on-chip timers	0000
TMR1_CC2 event		0001
TMR1_CC3 event		0010
TMR2_CC2 event		0011
TMR3_TRGO event		0100
TMR4_CC4 event		0101
EXTI line 11/TMR8_TRGO event ^(Note)	External pin/Internal signal from on-chip timers	0110
SWSTR	Software control bit	0111
TMR15_CC1 event	Internal signal from on-chip timers	1000
TMR15_CC2 event		1001
TMR15_CC3 event		1010
TMR15_CC4 event		1011
TMR15_TRGO event		1100
TMR1_TRGO event		1101
TMR8_CC1 event		1110
TMR8_CC2 event		1111

Note: For regular channels, the selection of the EXTI line 11 or TMR8_TRGO as external trigger event is done through the configuration of the ADC1_ETRGREG_REMAP bit and the ADC2_ETRGREG_REMAP bit in ADC1 and ADC2.

Table 13-4 ADC1 and ADC2 Used in External Trigger for Injected Channels

Source	Type	JEXSEL[3:0]
TMR1_TRGO event	Internal signal from on-chip timers	0000
TMR1_CC4 event		0001
TMR2_TRGO event		0010
TMR2_CC1 event		0011
TMR3_CC4 event		0100
TMR4_TRGO event		0101
EXTI line 15/TMR8_CC4 event ^(Note)	External pin/Internal signal from on-chip timers	0110
JSWSTR	Software control bit	0111
TMR15_CC1 event	Internal signal from on-chip timers	1000
TMR15_CC2 event		1001
TMR15_CC3 event		1010
TMR15_CC4 event		1011
TMR15_TRGO event		1100

TMR1_CC1 event		1101
TMR8_CC1 event		1110
TMR8_TRGO event		1111

Note: For injected channels, the selection of the EXTI line 15 or TMR8_CC4 as external trigger event is done through the configuration of the ADC1_ETRGINJ_REMAP bit and the ADC2_ETRGINJ_REMAP bit in the ADC1 and ADC2.

Table 13-5 ADC3 Used in External Trigger for Regular Channels

Source	Type	EXSEL[3:0]
TMR3_CC1 event	Internal signal from on-chip timers	0000
TMR2_CC3 event		0001
TMR1_CC3 event		0010
TMR8_CC1 event		0011
TMR8_TRGO event		0100
TMR5_CC1 event		0101
TMR5_CC3 event		0110
SWSTR	Software control bit	0111
TMR15_CC1 event	Internal signal from on-chip timers	1000
TMR15_CC2 event		1001
TMR15_CC3 event		1010
TMR15_CC4 event		1011
TMR15_TRGO event		1100
TMR1_TRGO event		1101
TMR1_CC1 event		1110
TMR8_CC3 event		1111

Table 13-6 ADC3 Used in External Trigger for Injected Channels

Source	Type	EXSEL[3:0]
TMR1_TRGO event	Internal signal from on-chip timers	0000
TMR1_CC4 event		0001
TMR4_CC3 event		0010
TMR8_CC2 event		0011
TMR8_CC4 event		0100
TMR5_TRGO event		0101
TMR5_CC4 event		0110
JSWSTR	Software control bit	0111
TMR15_CC1 event	Internal signal from on-chip timers	1000
TMR15_CC2 event		1001

TMR15_CC3 event	1010
TMR15_CC4 event	1011
TMR15_TRGO event	1100
TMR1_CC1 event	1101
TMR1_CC2 event	1110
TMR8_TRGO event	1111

Software trigger events can be generated by setting the SWSTR or JSWSTR bit in the ADC_CTRL2 register.

A regular group conversion can be interrupted by an injected trigger.

13.3.15 DMA Request

Since the converted values of regular channels are stored in the only one data register, it is necessary to use DMA for conversion of more than one regular channel. This avoids the loss of data which is already stored in the ADC_RDOR register.

A DMA request is generated only at the end of a regular channel conversion, which transfers its converted data from the ADC_RDOR register to the destination location designated by the user.

Note: This DMA function is available only for ADC1 and ADC3. ADC2-converted data can be transferred by using the DMA function of ADC1 through dual ADC mode.

13.3.16 Dual ADC Mode

In devices with two ADCs or more, dual ADC mode is available (Please refer to [Figure 13-8](#)).

In dual ADC mode, the start of conversion is triggered alternately or simultaneously by the ADC1 master and the ADC2 slave, depending on the mode selected by the DUALMOD[2:0] bits in the ADC1_CTRL1 register.

Note: In dual mode, when configuring conversion to be triggered by an external event, users must set the trigger for the master only and set software trigger for the slave. This prevents spurious triggers to start unwanted slave conversion. However, external triggers must be enabled on both master and slave ADCs.

There are six possible modes:

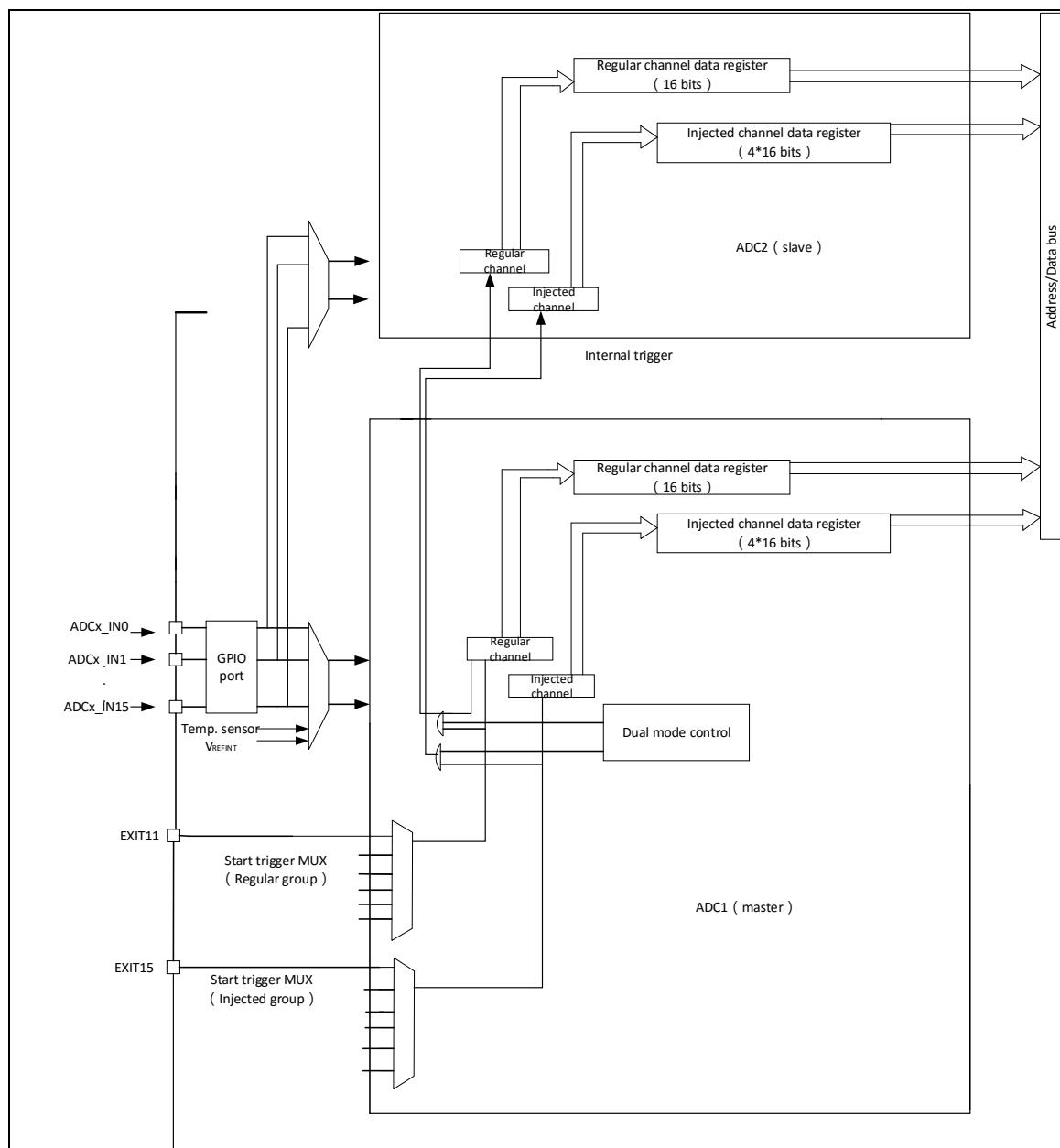
- Injected simultaneous mode
- Regular simultaneous mode
- Fast interleaved mode
- Slow interleaved mode
- Alternate trigger mode
- Independent mode

It is also possible to use the above modes combined in the following ways:

- Injected simultaneous mode + Regular simultaneous mode
- Regular simultaneous mode + Alternate trigger mode
- Injected simultaneous mode + Interleaved mode

Note: In dual ADC mode, to read the slave-converted data on the master data register, the DMAEN bit must be enabled, even if DMA is not used to transfer regular channel data.

Figure 13-8 Dual ADC Block Diagram (Note)



Note: 1. External trigger signal acts on ADC2, but it is not shown in Figure 13-8.
 2. In some dual ADC modes, the complete ADC1 data register (ADC1_RDOR) contains both ADC1 and ADC2 regular converted data.

13.3.16.1 Injected Simultaneous Mode

This mode converts an injected channel group. The external trigger comes from the injected group mux of ADC1 (selected by the JEXSEL[3:0] bits in the ADC1_CTRL2 register), and it provides a simultaneous trigger for ADC2 at the same time.

Note: Do not convert the same channel on two ADCs (no overlapping sampling times for two ADCs on the same channel).

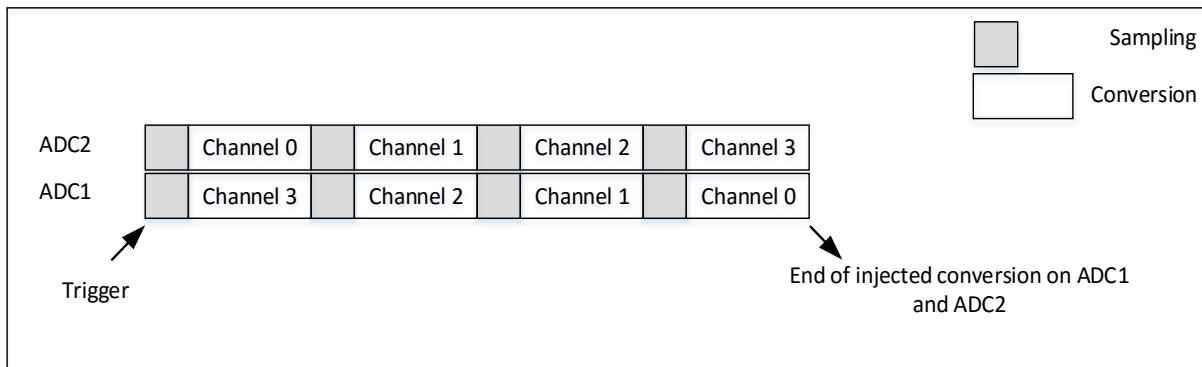
At the end of ADC1 or ADC2 conversion:

- The converted data is stored in the ADC_JDORx registers of each ADC interface.
- A JEC interrupt is generated (if enabled on any of the ADC interfaces) when the ADC1/ADC2 injected channels are all converted.

Note: In simultaneous mode, users must convert sequences with exactly the same sampling time, or ensure that the trigger interval is longer than the longer sequence within the two sequences. Otherwise, ADC conversion with the shorter

sequence might be restarted while the conversion of the longer sequence is not yet completed.

Figure 13-9 Injected Simultaneous Mode on 4 Channels



13.3.16.2 Regular Simultaneous Mode

This mode is performed on a regular channel group. The external trigger comes from the regular group mux of ADC1 (selected by the EXSEL[3:0] bits in the ADC1_CTRL2 register), and it provides a simultaneous trigger for ADC2 at the same time.

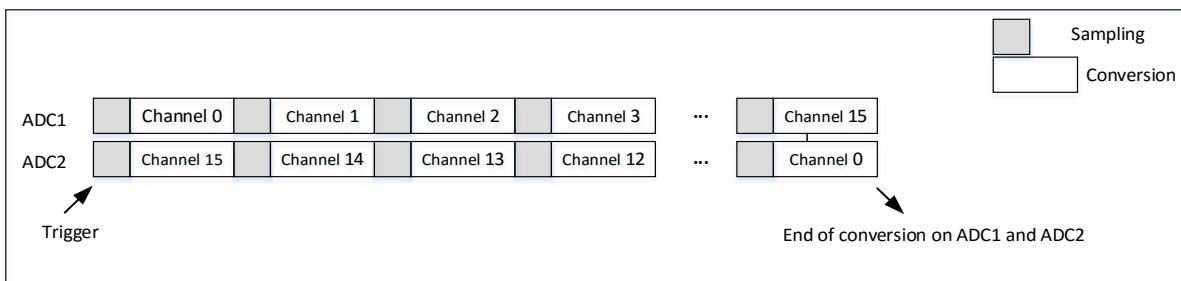
Note: *Do not convert the same channel on two ADCs (no overlapping sampling times for two ADCs on the same channel).*

At the end of ADC1 or ADC2 conversion:

- A 32-bit DMA transfer request is generated (if the DMAEN bit is set). The content of the 32-bit ADC1_RDOR register will be transferred to SRAM, which contains the ADC2 converted data in the upper half word, and the ADC1 converted data in the lower half word.
- An EC interrupt is generated (if enabled on any of the ADC interfaces) when ADC1/ADC2 regular channels are all converted.

Note: *In regular simultaneous mode, users must convert sequences with exactly the same sampling time, or ensure that the trigger interval is longer than the longer sequence within the two sequences. Otherwise, ADC conversion with the shorter sequence might be restarted while the conversion of the longer sequence is not yet completed.*

Figure 13-10 Regular Simultaneous Mode on 16 Channels



13.3.16.3 Fast Interleaved Mode

This mode only applies to regular channel group (usually one channel). The external trigger comes from the regular channel mux of ADC1. After an external trigger occurs:

- ADC2 starts immediately and,
- ADC1 starts after a delay of 7 ADC clock cycles.

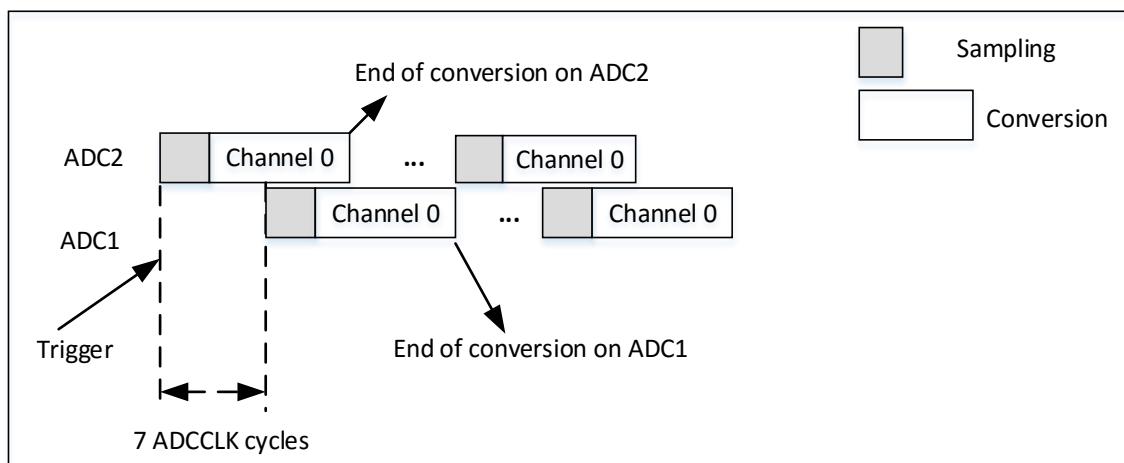
If the CONT bit is set on both ADC1 and ADC2, the selected regular channels of both ADCs are continuously converted.

After an EC interrupt is generated by ADC1 (enabled through the ECIEN bit), a 32-bit DMA transfer request is generated (if the DMAEN bit is set). The 32-bit data of the ADC1_RDOR register is transferred to SRAM. ADC1_RDOR contains the ADC2 converted data in the upper half word and the ADC1

converted data in the lower half word.

Note: *The maximum sampling time allowed is < 7 ADCCLK cycles. This avoids the overlap between ADC1 and ADC2 sampling phases when they convert the same channel.*

Figure 13-11 Fast Interleaved Mode on 1 Channel in Continuous Conversion Mode



13.3.16.4 Slow Interleaved Mode

This mode only applies to regular channel group (only one channel). The external trigger comes from regular channel mux of ADC1. After external trigger occurs:

- ADC2 starts immediately and,
- ADC1 starts after a delay of 14 ADC clock cycles.
- ADC2 restarts after a second delay of 14 ADC cycles, and so on.

Note: *The maximum sampling time allowed is < 14 ADCCLK cycles. This avoids the overlap with the next conversion.*

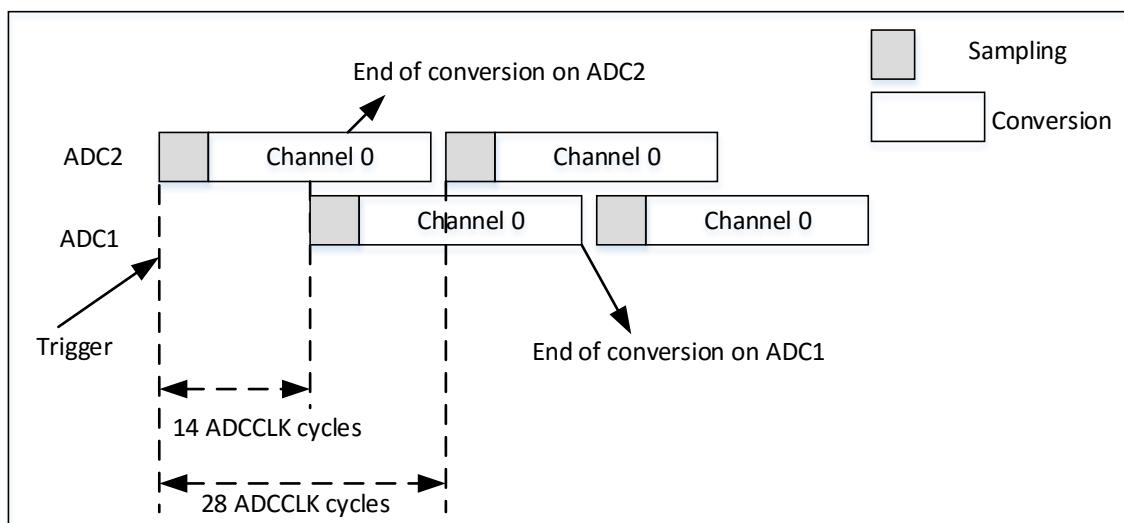
After an EC interrupt is generated by ADC1 (enabled through the ECIEN bit), a 32-bit DMA transfer request is generated (if the DMAEN bit is set). The 32-bit data of the ADC1_RDOR register is transferred to SRAM. ADC1_RDOR contains the ADC2 converted data in the upper half word and the ADC1 converted data in the lower half word.

A new ADC2 conversion is automatically started after 28 ADC clock cycles.

The CON bit cannot be set in this mode since it will continuously convert the selected regular channel.

Note: *The application must ensure that no external trigger for injected channel occurs when interleaved mode is enabled.*

Figure 13-12 Slow Interleaved Mode on 1 Channel



13.3.16.5 Alternate Trigger Mode

This mode only applies to injected channel group. The external trigger comes from the injected group mux of ADC1.

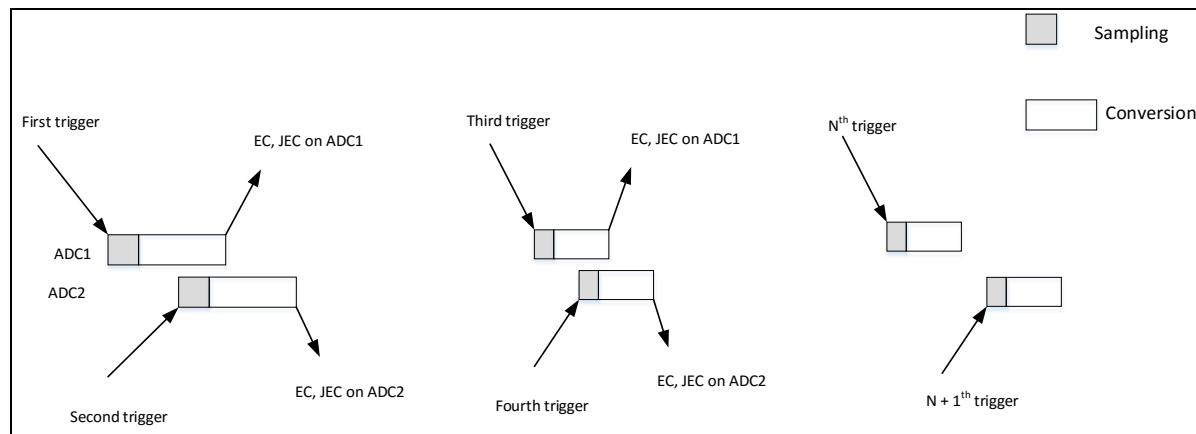
- When the first trigger occurs, all injected group channels of ADC1 are converted.
- When the second trigger arrives, all injected group channels of ADC2 are converted.
- And so on...

A JEC interrupt, if enabled, is generated after all injected group channels of ADC1 are converted.

A JEC interrupt, if enabled, is generated after all injected group channels of ADC2 are converted.

If another external trigger occurs after all injected group channels are converted, the alternate trigger process will restart from converting ADC1 injected group channels.

Figure 13-13 Alternate Trigger: Injected Channel Group of Each ADC



If injected discontinuous mode is enabled on both ADC1 and ADC2:

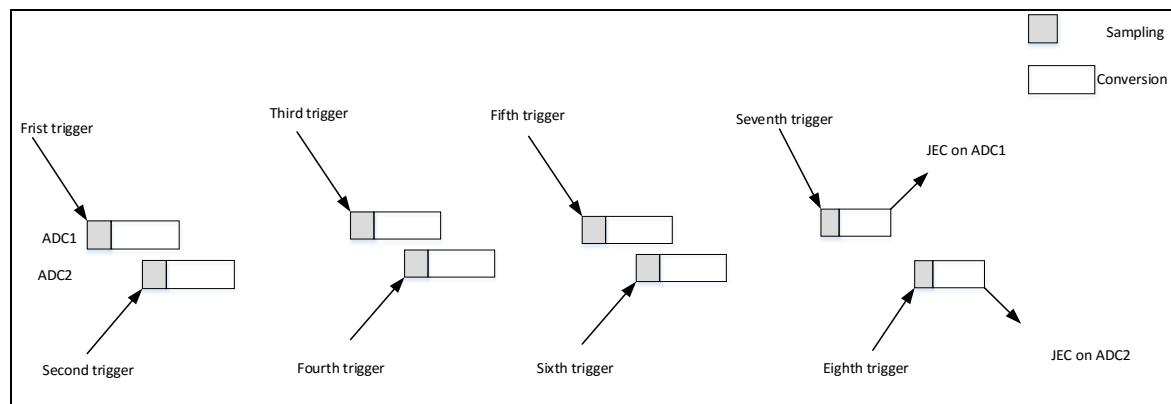
- When the first trigger occurs, the first injected channel of ADC1 is converted.
- When the second trigger arrives, the first injected channel of ADC2 is converted.
- And so on...

A JEC interrupt, if enabled, is generated after all injected group channels of ADC1 are converted.

A JEC interrupt, if enabled, is generated after all injected group channels of ADC2 are converted.

If another external trigger occurs after all injected group channels are converted, the alternate trigger process will restart

Figure 13-14 Alternate Trigger: 4 Injected Channels of Each ADC in Discontinuous Mode



13.3.16.6 Independent mode

In this mode, the dual ADC synchronization is bypassed, and each ADC interface works independently.

13.3.16.7 Combined Regular/Injected Simultaneous Mode

Simultaneous conversion of regular group can be interrupted to start simultaneous conversion of injected group.

Note: In combined regular/injected simultaneous mode, users must convert sequences with exactly the same sampling time, or ensure that the trigger interval is longer than the longer sequence within the two sequences. Otherwise, ADC conversion with the shorter sequence might be restarted while the conversion of the longer sequence is not yet completed.

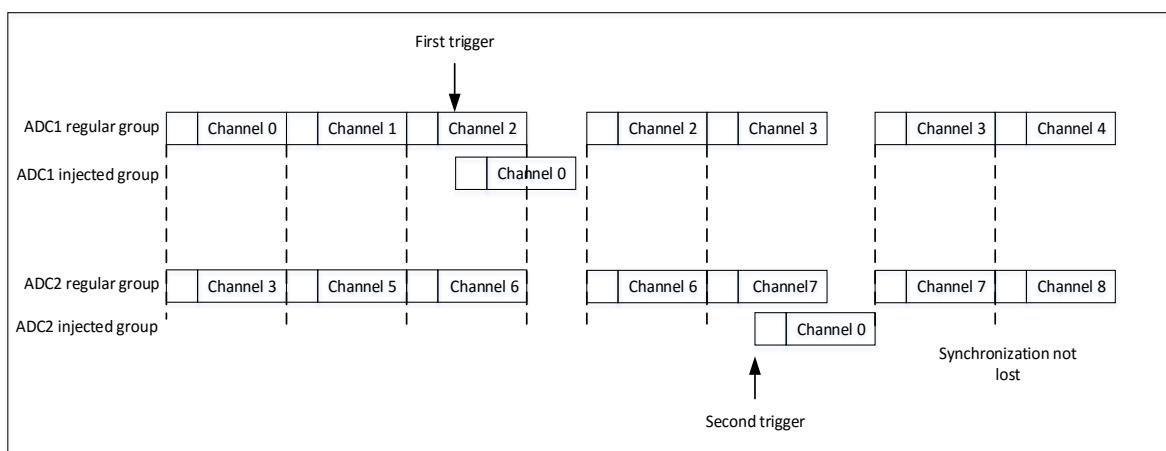
13.3.16.8 Combined Regular Simultaneous + Alternate Trigger Mode

Simultaneous conversion of regular group can be interrupted to start alternate trigger conversion of injected group. [Figure 13-15](#) shows an alternate trigger interrupting a regular simultaneous conversion.

The injected alternate conversion is started immediately after the injected event arrives. If regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all ADCs (master/slave) is stopped and will resume synchronously at the end of the injected conversion.

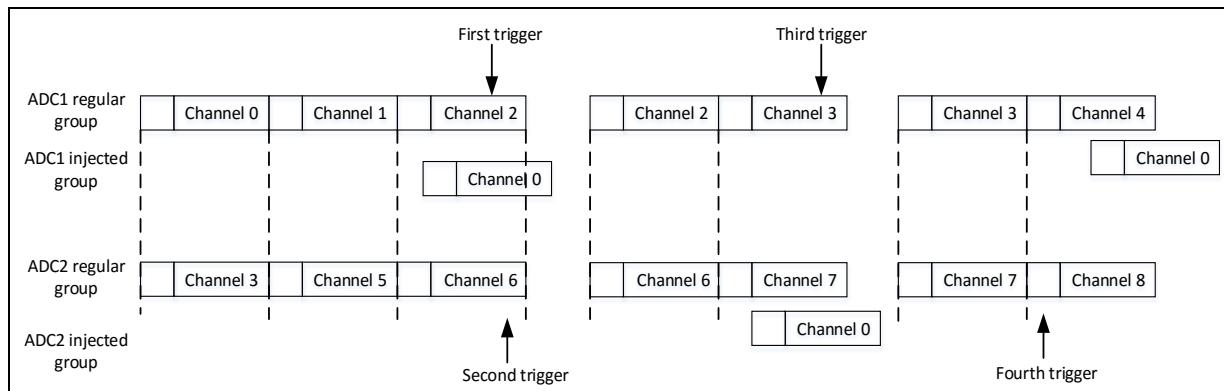
Note: In combined regular simultaneous + alternate trigger mode, users must convert sequences with exactly the same sampling time, or ensure that the trigger interval is longer than the longer sequence within the two sequences. Otherwise, ADC conversion with the shorter sequence might be restarted while the conversion of the longer sequence is not yet completed.

Figure 13-15 Alternate + Regular Simultaneous



If a trigger occurs during an injected conversion that interrupts a regular conversion, the trigger event will be ignored. Figure 13-16 shows the behavior in this case (the second trigger is ignored).

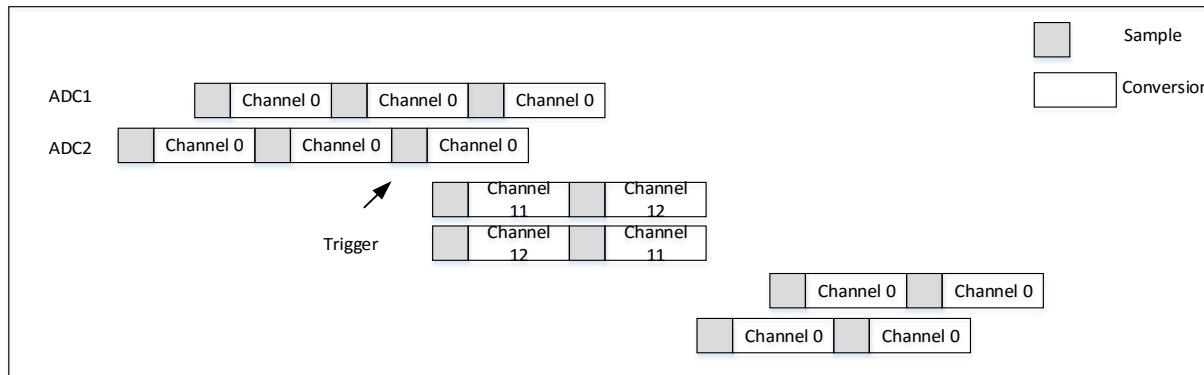
Figure 13-16 Trigger Event Occurring During Injected Conversion



13.3.16.9 Combined Injected Simultaneous + Interleaved mode

An injected event can interrupt an interleaved conversion. In this case, the interleaved conversion is interrupted, and the injected conversion is started. The interleaved conversion is resumed at the end of the injected sequence. Figure 13-17 shows an example of this case.

Figure 13-17 Interleaved Single Channel Conversion Interrupted by Injected Sequence, CH11 and CH12



13.3.17 Temperature Sensor

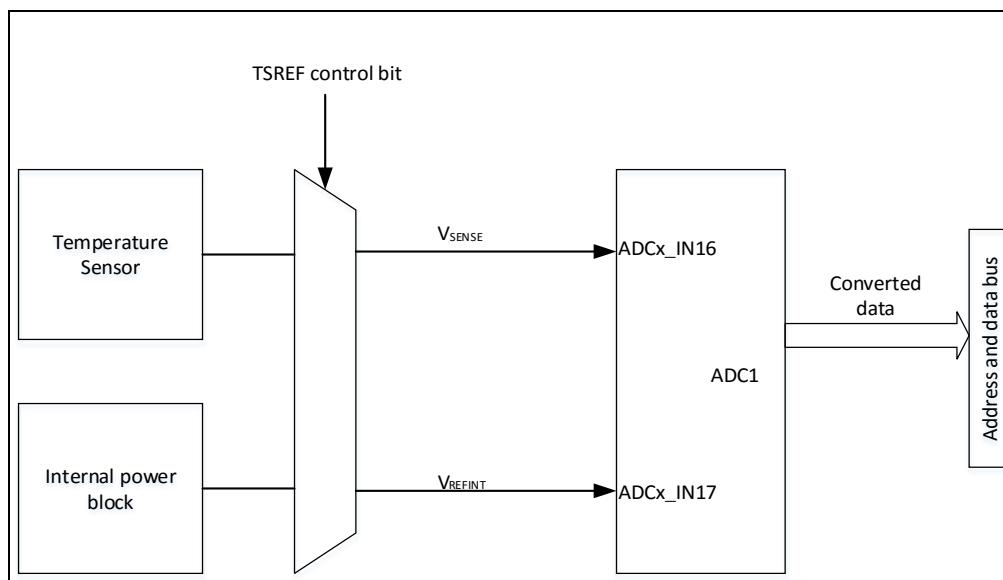
The temperature sensor can measure the ambient temperature (T_A) of the device. It is internally connected to the ADC1_IN16 input channel, which converts the sensor output voltage into a digital value. The recommended sampling time for the temperature sensor analog input is 8.6 μ s.

The block diagram of the temperature sensor is shown in [Figure 13-18](#). When not in use, this sensor can be put in power-down mode.

Note: *The TSREF bit must be set to enable internal channels: ADC1_IN16 (temperature sensor) and ADC1_IN17 (V_{REFINT}) conversion.*

The output voltage of temperature sensor changes linearly with temperature. The offset of this line varies from chip to chip (up to 45 °C) due to process variation. The internal temperature sensor is more appropriate for detecting temperature variations instead of absolute temperatures. If accurate temperature readings are needed, an external temperature sensor should be used.

Figure 13-18 Channel Block Diagram of Temperature Sensor and V_{REFINT}



Reading the temperature

To use the sensor:

1. Select the ADC1_IN16 input channel.
2. Select a sample time of 8.6 μ s.
3. Set the TSREF bit in the ADC control register 2 (ADC_CTRL2), to wake up the temperature

sensor from power-down mode.

4. Start the ADC conversion by setting the ADON bit (or by external trigger).
5. Read the V_{SENSE} data result in the ADC data register.
6. Obtain the temperature with the following formula:

$$\text{Temperature } (\text{°C}) = \{(V_{25} - V_{\text{SENSE}}) / \text{Avg_Slope}\} + 25$$

Where:

V_{25} = V_{SENSE} value for 25 °C

Avg_Slope = Average slope for curve between temperature and V_{SENSE} (given in mV/°C or µV/°C)

Please refer to the section of electrical characteristics in the data sheet for the actual values of V_{25} and Avg_Slope.

Note: After waking from power-down mode, the sensor has a startup time before it can output V_{SENSE} at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADON and TSREF bits should be set at the same time.

13.3.18 ADC Interrupts

For regular and injected groups, an interrupt can be generated at the end of conversion and when the analog watchdog status bit is set. Separate interrupt enable bits are available.

Note: ADC1 and ADC2 interrupts are mapped onto the same interrupt vector, while ADC3 interrupts are mapped onto a separate one.

Two other flags are present in the ADC_register, but they do not have related interrupts:

- JSTR (Start of injected group channels conversion)
- RSTR (Start of regular group channels conversion)

Table 13-7 ADC Interrupts

Interrupt Event	Flag	Enable Control Bit
End of regular group conversion	EC	ECIEN
End of injected group conversion	JEC	JECIEN
The analog watchdog status bit is set	AWD	AWDIEN

13.4 ADC Registers

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

Table 13-8 ADC Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
000h	ADC_STS	Reserved																									RSTR	JSTR	JEC	EC	AWD				
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
004h	ADC_CTRL1	Reserved				AWDEN	JAWDEN	Reserved	DUALM[3:0]				DISN[2:0]				JDSEN	RDISEN	JAUT	AWDSGE	SCN	JECIEN	AWDIEN	ECIEN	AWDCS[4:0]				RSTCAL	CAL	CON	ADON	0		
		0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
008h	ADC_CTRL2	Reserved				EXSEL[3]	JEXSEL[3]	TSREF	EXSEL[2:0]				Reserved				JEXTTRIG	JEXSEL[2:0]				DALIGN	Reserved	DMAEN	Reserved				RSTCAL	CAL	CON	ADON	0		
		0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										

Offset	register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
00Ch	ADC_SMPT1	Reserved					SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			MP11[2:0]			SMP10[2:0]						
	Reset Value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_SMPT2	Reserved	SMP9[2:0]		SMP8[2:0]		SMP7[2:0]		SMP6[2:0]		SMP5[2:0]		SMP4[2:0]		SMP3[2:0]		SMP2[2:0]		SMP1[2:0]		SMP0[2:0]													
010h	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_JOFS1	Reserved														JOFST1[11:0]																		
014h	Reset Value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_JOFS2	Reserved														JOFST2[11:0]																		
018h	Reset Value																																	
	ADC_JOFS3	Reserved														JOFST3[11:0]																		
01Ch	Reset Value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_JOFS4	Reserved														JOFST4[11:0]																		
020h	Reset Value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_WHTR	Reserved														AWHT[11:0]																		
024h	Reset Value															1	1	1	1	1	1	1	1	1	1	1	1	1						
	ADC_WLTR	Reserved														AWLT[11:0]																		
028h	Reset Value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	ADC_RSQ1	Reserved					LEN[3:0]		SQ16[4:0]		SQ15[4:0]		SQ14[4:0]		SQ13[4:0]																			
02Ch	Reset Value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ADC_RSQ2	Reserved	SQ12[4:0]		SQ11[4:0]		SQ10[4:0]		SQ9[4:0]		SQ8[4:0]		SQ7[4:0]																					
030h	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ADC_RSQ3	Reserved	SQ6[4:0]		SQ5[4:0]		SQ4[4:0]		SQ3[4:0]		SQ2[4:0]		SQ1[4:0]																					
034h	Reset Value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ADC_JSQ	Reserved					JLEN[3:0]		JSQ4[4:0]		JSQ3[0]		JSQ2[4:0]		JSQ1[4:0]																			
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
03Ch	ADC_JDOR1	Reserved														JD[15:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
040h	ADC_JDOR2	Reserved														JD[15:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
044h	ADC_JDOR3	Reserved														JD[15:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
048h	ADC_JDOR4	Reserved														JD[15:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
04Ch	ADC_RDOR	AD2D[15:0]														D[15:0]																		
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

13.4.1 ADC Status Register (ADC_STS)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RSTR		JSTR		JEC		EC	
res								rc w0		rc w0		rc w0		rc w0	

Bit 31:15	Reserved. Must be kept as 0.
Bit 4	RSTR: Regular channel Start flag This bit is set by hardware at the start of regular channel conversion, and it is cleared by software. 0: Regular channel conversion is not started yet. 1: Regular channel conversion is started.
Bit 3	JSTR: Injected channel Start flag This bit is set by hardware at the start of injected channel group conversion, and it is cleared by software. 0: Injected channel group conversion is not started yet. 1: Injected channel group conversion is started.
Bit 2	JEC: Injected channel end of conversion This bit is set by hardware at the end of all injected channel group conversion, and it is cleared by software. 0: Conversion is not completed yet. 1: Conversion is completed.
Bit 1	EC: End of conversion This bit is set by hardware at the end of channel group conversion (regular or injected) channel, and it is cleared by software or by reading ADC_RDOR. 0: Conversion is not completed yet. 1: Conversion is completed.
Bit 0	AWD: Analog watchdog flag This bit is set by hardware when the converted voltage exceeds the values programmed in the ADC_WLTR and ADC_WHTR registers. It is cleared by software. 0: No analog watchdog event occurs. 1: An analog watchdog event occurs.

13.4.2 ADC Control Register 1 (ADC_CTRL1)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								AWDEN		JAWDEN		Reserved		DUALM[3:0]	
res								rw		rw		res		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DISN[2:0]		JDISEN	RDISEN	JAUT	AWDSGE	SCN	JECIEN	AWDIEN	ECIEN	AWDCS[4:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:24						Reserved. Must be kept as 0.									

Bit 23	AWDEN: Analog watchdog enable on regular channels This bit is set and cleared by software. 0: Analog watchdog is disabled on regular channel. 1: Analog watchdog is enabled on regular channel.
Bit 22	JAWDEN: Analog watchdog enable on injected channels This bit is set and cleared by software. 0: Analog watchdog is disabled on injected channel. 1: Analog watchdog is enabled on injected channel.
Bit 21:20	Reserved. Must be kept as 0.
Bit 19:16	DUALM[3:0]: Dual mode selection These bits are written by software to select the operation mode. 0000: Independent mode 0001: Combined Regular Simultaneous + Injected Simultaneous Mode 0010: Combined Regular Simultaneous + Alternate Trigger Mode 0011: Combined Injected Simultaneous + Fast Interleaved Mode 0100: Combined Injected Simultaneous + Slow Interleaved Mode 0101: Injected Simultaneous mode 0110: Regular Simultaneous mode 0111: Fast Interleaved Mode 1000: Slow Interleaved Mode 1001: Alternate Trigger Mode Note: These bits are reserved in ADC2 and ADC3. In dual mode, a change of channel configuration generates a restart which will cause loss of synchronization. It is recommended that users disable dual mode before any configuration change.
Bit 15:13	DISN[2:0]: Discontinuous mode channel count These bits are written by software to define the number of regular channels to be converted after receiving an external trigger in discontinuous mode. 000: 1 channel 001: 2 channels 111: 8 channels
Bit 12	JDISEN: Discontinuous mode on injected channels This bit is set and cleared by software to enable or disable discontinuous mode on injected channels. 0: Discontinuous mode is disabled on injected channels. 1: Discontinuous mode is enabled on injected channels.
Bit 11	RDISEN: Discontinuous mode on regular channels This bit is set and cleared by software to enable or disable discontinuous mode on regular channels. 0: Discontinuous mode is disabled on regular channels. 1: Discontinuous mode is enabled on regular channels.
Bit 10	JAUT: Automatic Injected Group conversion This bit is set and cleared by software to enable or disable automatic injected group conversion after regular channel group conversion ends. 0: Automatic injected channel group conversion is disabled. 1: Automatic injected channel group conversion is enabled.
Bit 9	AWDSGE: Enable the watchdog on a single channel in scan mode This bit is set and cleared by software to enable or disable the analog watchdog function defined by the AWDCS[4:0] bits. 0: Analog watchdog is enabled on all channels. 1: Analog watchdog is enabled on a single channel.
Bit 8	SCN: Scan mode This bit is set and cleared by software to enable or disable scan mode. In scan mode, channels selected by the ADC_RSQx or ADC_JSQx register are converted. 0: Scan mode is disabled. 1: Scan mode is enabled. Note: If the ECIEN or JECIEN bits are set respectively, EC or JEC interrupt is generated only after the last channel conversion is completed.

Bit 7	JECIEN: Interrupt enable for injected channels This bit is set and cleared by software to enable or disable interrupt generation after all injected channel conversions are completed. 0: JEC interrupt is disabled. 1: JEC interrupt is enabled. Interrupt is generated when the JEC bit is set by hardware.
Bit 6	AWDIEN: Analog watchdog interrupt enable This bit is set and cleared by software to enable or disable analog watchdog interrupt. In scan mode, if the watchdog detects values exceeding the threshold, the scan will be stopped only if this bit is set. 0: Analog watchdog interrupt is disabled. 1: Analog watchdog interrupt is enabled.
Bit 5	ECIEN: Interrupt enable for EC This bit is set and cleared by software to enable or disable interrupt generation after conversion. 0: EC interrupt is disabled. 1: EC interrupt is enabled. An interrupt is generated when the EC bit is set by hardware.
Bit 4:0	AWDCS[4:0]: Analog watchdog channel select bits These bits are set and cleared by software to select the input channel to be protected by the analog watchdog. 00000: ADC analog input channel 0 00001: ADC analog input channel 1 01111: ADC analog input channel 15 10000: ADC analog input channel 16 10001: ADC analog input channel 17 All the other values are reserved. Note: ADC1 analog channel16 and channel17 are internally connected to the temperature sensor and to V_{REFINT} , respectively. ADC2 analog inputs, channel16 and channel 17, are internally connected to V_{SS} . ADC3 analog inputs, channel 9, channel 14, channel 15, channel 16, and channel 17, are connected to V_{SS} .

13.4.3 ADC Control Register 2 (ADC_CTRL2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved						EXSE L[3]	JEXSE L[3]	TSR EF	SWS TR	JSWS TR	EXTT RIG	EXSE L[2:0]		Reserved	
			res			rw	rw	rw	rw	rw	rw	rw	rw	rw	res
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JEXT TRIG	JEXSEL[2:0]		DALI GN	Reserved	DM AEN	Reserved			RST CAL	CAL	CON	ADON			
rw	rw	rw	rw	rw	res	rw	res			rw	rw	rw	rw	rw	

Bit 31:24	Reserved. Must be kept as 0.
Bit 25	EXSEL[3] Please refer to EXSEL[3:0] definition.
Bit 24	JEXSEL[3] Please refer to JEXSEL[3:0] definition.
Bit 23	TSREF: Temperature sensor and V_{REFINT} enable This bit is set and cleared by software, to enable or disable the temperature sensor and V_{REFINT} channel. In devices with more than one ADC, this bit is present only in ADC1. 0: Temperature sensor and V_{REFIN} are disabled. 1: Temperature sensor and V_{REFIN} are enabled.

Bit 22	SWSTR: Start conversion of regular channels This bit is set by software to start conversion, and it is cleared by hardware as soon as the conversion starts. This bit can start the conversion of a regular channel group if SWSTR is selected as trigger event in the EXSEL[3:0] bits. 0: Reset state 1: Start regular channel conversion
Bit 21	JSWSTR: Start conversion of injected channels This bit is set by software to start conversion, and it can be cleared by software, or by hardware as soon as the conversion starts. This bit can start the conversion of an injected channel group if JSWSTR is selected as trigger event in the JEXSEL[3:0] bits. 0: Reset state 1: Start injected channel conversion
Bit 20	EXTTRIG: External trigger conversion mode for regular channels This bit is set and cleared by software to enable or disable external trigger event which starts regular channel group conversion. 0: Conversion on external event is disabled. 1: Conversion on external event is enabled.
Bit 25 Bit 19:17	EXSEL[3:0]: External event select for regular group These bits select the external event that starts regular channel group conversion. Trigger configurations for ADC1 and ADC2 are as follows: 0000: Timer 1 CC1 event 0001: Timer 1 CC2 event 0010: Timer 1 CC3 event 0011: Timer 2 CC2 event 0100: Timer 3 TRGO event 0101: Timer 4 CC4 event 0110: EXTI line 11/ TMR8_TRGO event 0111: SWSTR 1000: Timer 15 CC1 event 1001: Timer 15 CC2 event 1010: Timer 15 CC3 event 1011: Timer 15 CC4 event 1100: Timer 15 TRGO event 1101: Timer 1 TRGO event 1110: Timer 8 CC1 event 1111: Timer 8 CC2 event Trigger configurations for ADC3 are as follows: 0000: Timer 3 CC1 event 0001: Timer 2 CC3 event 0010: Timer 1 CC3 event 0011: Timer 8 CC1 event 0100: Timer 8 TRGO event 0101: Timer 5 CC1 event 0110: Timer 5 CC3 event 0111: SWSTR 1000: Timer 15 CC1 event 1001: Timer 15 CC2 event 1010: Timer 15 CC3 event 1011: Timer 15 CC4 event 1100: Timer 15 TRGO event 1101: Timer 1 TRGO event 1110: Timer 1 CC1 event 1111: Timer 8 CC3 event
Bit 16	Reserved. Must be kept as 0.
Bit 15	JEXTTRIG: External trigger conversion mode for injected channels This bit is set and cleared by software to enable or disable external trigger event which starts injected channel group conversion. 0: Conversion on external event is disabled. 1: Conversion on external event is enabled.

Bit 24 Bit 14:12	<p>JEXSEL[3:0]: External event select for injected group These bits select the external event that starts injected channel group conversion.</p> <p>Trigger configurations for ADC1 and ADC2 are as follows:</p> <ul style="list-style-type: none"> 0000: Timer 1 TRGO event 0001: Timer 1 CC4 event 0010: Timer 2 TRGO event 0011: Timer 2 CC1 event 0100: Timer 3 CC4 event 0101: Timer 4 TRGO event 0110: EXTI line 15/TMR8_CC4 event 0111: JSWSTR 1000: Timer 15 CC1 event 1001: Timer 15 CC2 event 1010: Timer 15 CC3 event 1011: Timer 15 CC4 event 1100: Timer 15 TRGO event 1101: Timer 1 CC1 event 1110: Timer 8 CC1 event 1111: Timer 8 TRGO event <p>Trigger configurations for ADC3 are as follows:</p> <ul style="list-style-type: none"> 0000: Timer 1 TRGO event 0001: Timer 1 CC4 event 0010: Timer 4 CC3 event 0011: Timer 8 CC2 event 0100: Timer 8 CC4 event 0101: Timer 5 TRGO event 0110: Timer 5 CC4 event 0111: JSWSTR 1000: Timer 15 CC1 event 1001: Timer 15 CC2 event 1010: Timer 15 CC3 event 1011: Timer 15 CC4 event 1100: Timer 15 TRGO event 1101: Timer 1 CC1 event 1110: Timer 1 CC2 event 1111: Timer 8 TRGO event
Bit 11	<p>DALIGN: Data alignment This bit is set and cleared by software. Please refer to Figure 13-6 and Figure 13-7. 0: Right alignment 1: Left alignment</p>
Bit 10:9	Reserved. Must be kept as 0.
Bit 8	<p>DMAEN: Direct memory access mode This bit is set and cleared by software. Please refer to the section of DMA Controller. 0: DMA mode is disabled. 1: DMA mode is enabled. Note: Only ADC1 and ADC3 generate DMA requests.</p>
Bit 7:4	Reserved. Must be kept as 0.
Bit 3	<p>RSTCAL: Reset calibration This bit is set by software and cleared by hardware. It is cleared after the calibration register is initialized. 0: Calibration register is initialized. 1: Initialize calibration register Note: If RSTCAL is set when conversion is ongoing, additional cycles are required to clear the calibration registers.</p>
Bit 2	<p>CAL: A/D Calibration This bit is set by software to start calibration and cleared by hardware when the calibration is completed. 0: Calibration is completed. 1: Start calibration</p>

Bit 1	CON: Continuous conversion This bit is set and cleared by software. If this bit is set, conversion will be continuously performed until this bit is cleared. 0: Single conversion mode 1: Continuous conversion mode
Bit 0	ADON: A/D converter ON/OFF This bit is set and cleared by software. When this bit is '0', writing '1' will wake up the ADC from power-down mode. When this bit is '1', writing '1' will enable conversion. The application should allow a delay of t _{STAB} between power-up and start of conversion. Please refer to Figure 13-2 . 0: Disable ADC conversion/calibration and enter power-down mode 1: Enable ADC to start conversion Note: If any other bit in this register is changed with the ADON bit at the same time, then conversion is not triggered. This is to prevent triggering a wrong conversion.

13.4.4 ADC Sample Time Register 1 (ADC_SMPT1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								SMP17[2:0]	SMP16[2:0]	SMP15[2:1]					
				res				rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15 [0]	SMP14[2:0]		SMP13[2:0]		SMP12[2:0]		SMP11[2:0]		SMP10[2:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:24	Reserved. Must be kept as 0.
Bit 23:0	SMPx[2:0]: Channel x Sample time selection These bits select the sample time for each channel individually. During sample cycles, channel selection bits must remain unchanged. 000: 1.5 cycles 100: 41.5 cycles 001: 7.5 cycles 101: 55.5 cycles 010: 13.5 cycles 110: 71.5 cycles 011: 28.5 cycles 111: 239.5 cycles Note: ADC1 analog input channel16 and channel 17 are internally connected to the temperature sensor and to V _{REFINT} , respectively. ADC2 analog input, channel 16 and channel 17, are internally connected to V _{ss} . ADC3 analog inputs, channel 14, channel 15, channel 16, and channel 17, are connected to V _{ss} .

13.4.5 ADC Sample Time Register 2 (ADC_SMPT2)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SMP9[2:0]			SMP8[2:0]			SMP7[2:0]		SMP6[2:0]		SMP5[2:1]			
		res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]		SMP3[2:0]		SMP2[2:0]		SMP1[2:0]		SMP0[2:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:30		Reserved. Must be kept as 0.													

Bit 29:0	SMPx[2:0]: Channel x Sample time selection These bits select the sample time for each channel individually. During sample cycles, channel selection bits must remain unchanged.	
	000: 1.5 cycles	100: 41.5 cycles
	001: 7.5 cycles	101: 55.5 cycles
	010: 13.5 cycles	110: 71.5 cycles
	011: 28.5 cycles	111: 239.5 cycles
Note: ADC3 analog input channel 9 is connected to Vss.		

13.4.6 ADC Injected Channel Data Offset Register x (ADC_JOFSx) (x = 1...4)

Address offset: 0x14 ~ 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		JOFSTx[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved. Must be kept as 0.													
Bit 11:0		JOFSTx[11:0]: Data offset for injected channel x These bits define the offset to be subtracted from the raw converted data when converting injected channels. The conversion result can be read from in the ADC_JDORx registers.													

13.4.7 ADC Watchdog High Threshold Register (ADC_WHTR)

Address offset: 0x24

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		AWHT[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved. Must be kept as 0.													
Bit 11:0		AWHT[11:0]: Analog watchdog high threshold These bits define the high threshold for analog watchdog.													

13.4.8 ADC Watchdog Low Threshold Register (ADC_WLTR)

Address offset: 0x28

Reset value: 0x0000 0000

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

Reserved	AWLT[11:0]											
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12	Reserved. Must be kept as 0.											
Bit 11:0	AWLT[11:0]: Analog watchdog low threshold These bits define the low threshold for analog watchdog.											

13.4.9 ADC Regular Sequence Register 1 (ADC_RSQ1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							LEN[3:0]				SQ16[4:1]				
				res			rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ16[0]	SQ15[4:0]							SQ14[4:0]				SQ13[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:24	Reserved. Must be kept as 0.											
Bit 23:20	LEN[3:0]: Regular channel sequence length These bits are set by software to define the total number of conversions in the regular channel conversion sequence. 0000: 1 conversion 0001: 2 conversions 1111: 16 conversions											
Bit 19:15	SQ16[4:0]: 16th conversion in regular sequence These bits are set by software to define the channel number (0 ~ 17) of the 16th conversion in the sequence.											
Bit 14:10	SQ15[4:0]: 15th conversion in regular sequence											
Bit 9:5	SQ14[4:0]: 14th conversion in regular sequence											
Bit 4:0	SQ13[4:0]: 13th conversion in regular sequence											

13.4.10 ADC Regular Sequence Register 2 (ADC_RSQ2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	SQ12[4:0]							SQ11[4:0]				SQ10[4:1]			
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ10[0]	SQ9[4:0]							SQ8[4:0]				SQ7[4:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:30	Reserved. Must be kept as 0.											
Bit 29:25	SQ12[4:0]: 12th conversion in regular sequence These bits are set by software to define the channel number (0 ~ 17) of the 12th conversion in the sequence.											
Bit 24:20	SQ11[4:0]: 11th conversion in regular sequence											

Bit 19:15	SQ10[4:0]: 10th conversion in regular sequence													
Bit 14:10	SQ9[4:0]: 9th conversion in regular sequence													
Bit 9:5	SQ8[4:0]: 8th conversion in regular sequence													
Bit 4:0	SQ7[4:0]: 7th conversion in regular sequence													

13.4.11 ADC Regular Sequence Register 3 (ADC_RSQ3)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]				SQ5[4:0]				SQ4[4:1]					
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ4[0]	SQ3[4:0]				SQ2[4:0]				SQ1[4:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:30	Reserved. Must be kept as 0.													
Bit 29:25	SQ6[4:0]: 6th conversion in regular sequence These bits are set by software to define the channel number (0 ~ 17) of the 6th conversion in the sequence.													
Bit 24:20	SQ5[4:0]: 5th conversion in regular sequence													
Bit 19:15	SQ4[4:0]: 4th conversion in regular sequence													
Bit 14:10	SQ3[4:0]: 3rd conversion in regular sequence													
Bit 9:5	SQ2[4:0]: 2nd conversion in regular sequence													
Bit 4:0	SQ1[4:0]: 1st conversion in regular sequence													

13.4.12 ADC Injected Sequence Register (ADC_JSQ)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										JLEN[3:0]	JSQ4[4:0]				
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ4[0]	JSQ3[4:0]				JSQ2[4:0]				JSQ1[4:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:22	Reserved. Must be kept as 0.													
Bit 21:20	JLEN[1:0]: Injected sequence length These bits are set by software to define the total number of conversions in the injected channel conversion sequence. 00: 1 conversion 01: 2 conversions 10: 3 conversions 11: 4 conversions													

Bit 19:15	JSQ4[4:0] : 4th conversion in injected sequence These bits are set by software to define the channel number (0 ~ 17) of the 4th conversion in the sequence. Note: Unlike a regular conversion sequence, if the length of JLEN[1:0] is less than four, the channels are converted in a sequence starting from (4-JLEN). For example, ADC_JSQ[21:0] = 10 00011 00011 00111 00010 means that the scan conversion will convert in the following channel sequence: 7, 3, 3, instead of 2, 7, 3.
Bit 14:10	JSQ3[4:0] : 3rd conversion in injected sequence
Bit 9:5	JSQ2[4:0] : 2nd conversion in injected sequence
Bit 4:0	JSQ1[4:0] : 1st conversion in injected sequence

13.4.13 ADC Injected Data Register x (ADC_JDORx) (x = 1...4)

Address offset: 0x3C - 0x48

Reset value: 0x0000 0000

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

Reserved

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

res

JD[15:0]

r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Bit 31:16	Reserved. Must be kept as 0.
Bit 21:20	JD[15:0] : Injected data These bits are read only. They contain the conversion result of the injected channels. The data will be left-aligned or right-aligned as shown in Figure 13-6 and Figure 13-7 .

13.4.14 ADC Regular Data Register (ADC_RDOR)

Address offset: 0x4C

Reset value: 0x0000 0000

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

ADC2D[15:0]

r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

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

D[15:0]

r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Bit 31:16	ADC2D[15:0] : ADC2 data - In ADC1: In dual mode, these bits contain the channel data of ADC2. Please refer to Section 13.3.16 . - In ADC2 and ADC3: these bits are not used.
Bit 15:0	D[15:0] : Regular data These bits are read only. They contain the conversion result of the regular channels. The data will be left-aligned or right-aligned as shown in Figure 13-6 and Figure 13-7 .

14 Digital-to-Analog Converter (DAC)

14.1 DAC Introduction

The DAC module is a 12-bit digital input and voltage output digital-to-analog converter. The DAC can be configured in 8-bit or 12-bit mode and can also be used with the DMA controller. In 12-bit mode, data could be left-aligned or right-aligned. The DAC has two output channels, and each with its own converter. In dual DAC mode, conversions could be done independently or simultaneously on both channels for synchronous update operation. An input reference pin V_{REF+} is available for more accurate resolution.

14.2 DAC Main Features

- Two DAC converters: one output channel to each converter
- 8-bit or 12-bit monotonic output
- Left or right data alignment in 12-bit mode
- Synchronized update capability
- Noise wave generation
- Triangular wave generation
- Dual DAC channel independent or simultaneous conversions
- DMA capability on each channel
- External triggers for conversion
- Input reference voltage V_{REF+}

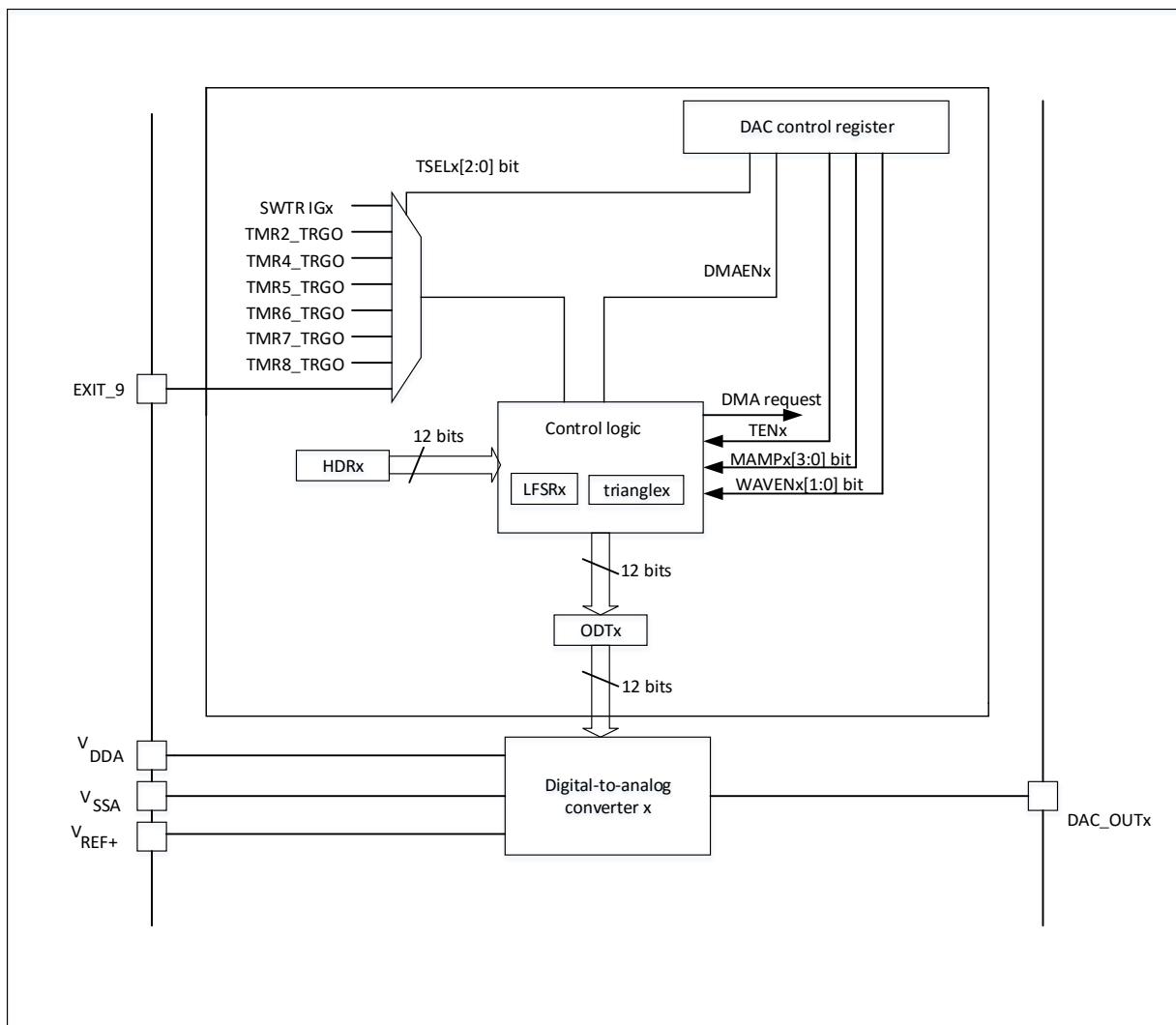
The block diagram of a DAC channel is shown in Figure 14-1, and the pin description is listed in [Table 14-1](#).

Table 14-1 DAC Pins

Name	Signal Type	Description
V_{REF+}	Input, analog reference positive	The higher/positive reference voltage for the DAC, $2.4 \text{ V} \leq V_{REF+} \leq V_{DDA}$ (3.3 V)
V_{DDA}	Input, analog supply	Analog power supply
V_{SSA}	Input, analog supply ground	Ground for analog power supply
DAC_OUTx	Analog output signal	DAC channel x analog output

Note: Once $DACx$ channel is enabled, the corresponding GPIO pin (PA4 or PA5) is automatically connected to the DAC analog output (DAC_OUTx). To avoid parasitic interference and extra consumption, the PA4 or PA5 pin should first be configured to analog input (AIN).

Figure 14-1 Block Diagram of DAC Channels



14.3 DAC Function Overview

14.3.1 DAC Channel Enable

DAC channel x can be powered on by setting the CHENx bit in the DAC_CTRL register. DAC channel x is then enabled after a startup time t_{WAKEUP} .

Note: The CHENx bit only enables the analog part of DAC channel x. The digital interface of DAC channel x still works even if the CHENx bit is '0'.

14.3.2 DAC Output Buffer Enable

The DAC integrates two output buffers that can be used to reduce output impedance, and to drive external loads directly without an external operational amplifier. Each DAC channel output buffer can be enabled and disabled through the BFFx bit in the DAC_CTRL register.

14.3.3 DAC Data Format

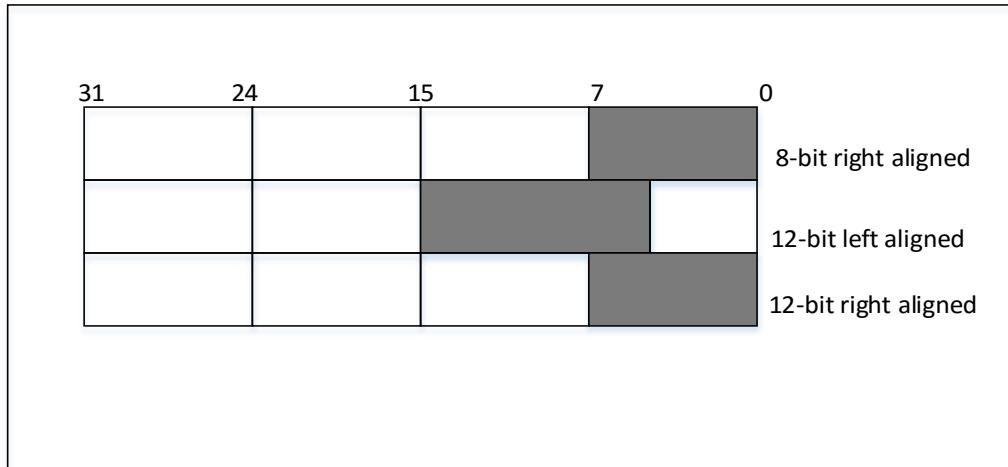
Based on the selected configuration mode, data is written to the specified registers as follows:

- Single DAC channel x, three possibilities:
 - 8-bit right alignment: Users have to write data into the DAC_HDR8Rx[7:0] bits (actually stored into the HDRx[11:4] bits)
 - 12-bit left alignment: Users have to write data into the DAC_HDR12Lx[15:4] bits (actually stored into the HDRx[11:0] bits)
 - 12-bit right alignment: Users have to write data into the DAC_HDR12Rx[11:0]

bits (actually stored into the $\text{HDR}_x[11:0]$ bits)

Depending on the loaded DAC_HDR_{yyx} register, and after the corresponding bits are shifted, the data written will be stored into the HDR_x register (HDR_x is the internal data holding register x). Then, the contents of the HDR_x register will be loaded into the ODT_x register automatically, or by software trigger, or by an external event trigger.

Figure 14-2 Data Register in Single DAC Channel Mode

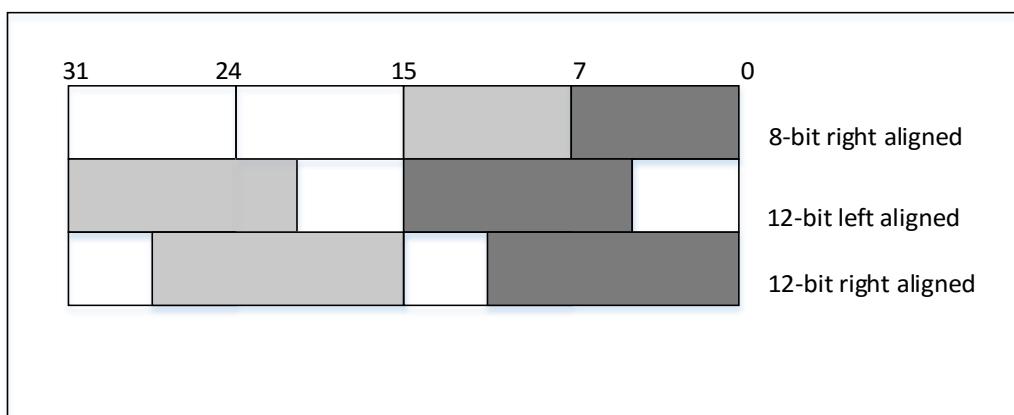


- Dual DAC channel x , three possibilities:

- 8-bit right alignment: Users have to write DAC channel 1 data into the $\text{DAC_HDR8Rx}[7:0]$ bits (actually stored into the $\text{HDR1}[11:4]$ bits), and write DAC channel 2 data into the $\text{DAC_HDR8RD}[15:8]$ bits (actually stored into the $\text{HDR2}[11:4]$ bits).
- 12-bit left alignment: Users have to write DAC channel 1 data into the $\text{DAC_HDR12LD}[15:4]$ bits (actually stored into the $\text{HDR1}[11:0]$ bits), and write DAC channel 2 data into the $\text{DAC_HDR12LD}[31:20]$ bits (actually stored into the $\text{HDR2}[11:0]$ bits).
- 12-bit right alignment: Users have to write DAC channel 1 data into the $\text{DAC_HDR12RD}[11:0]$ bits (actually stored into the $\text{HDR1}[11:0]$ bits), and write DAC channel 2 data into the $\text{DAC_HDR12RD}[27:16]$ bits (actually stored into the $\text{HDR2}[11:0]$ bits).

Depending on the loaded DAC_HDR_{yyD} register, and after the corresponding bits are shifted, the data written will be stored into the HDR1 and HDR2 registers (HDR1 and HDR2 are the internal data holding registers x). Then, the contents of the HDR1 and HDR2 registers will be loaded into the ODT_x register automatically, or by software trigger, or by an external event trigger.

Figure 14-3 Data Register in Dual DAC Channel Mode



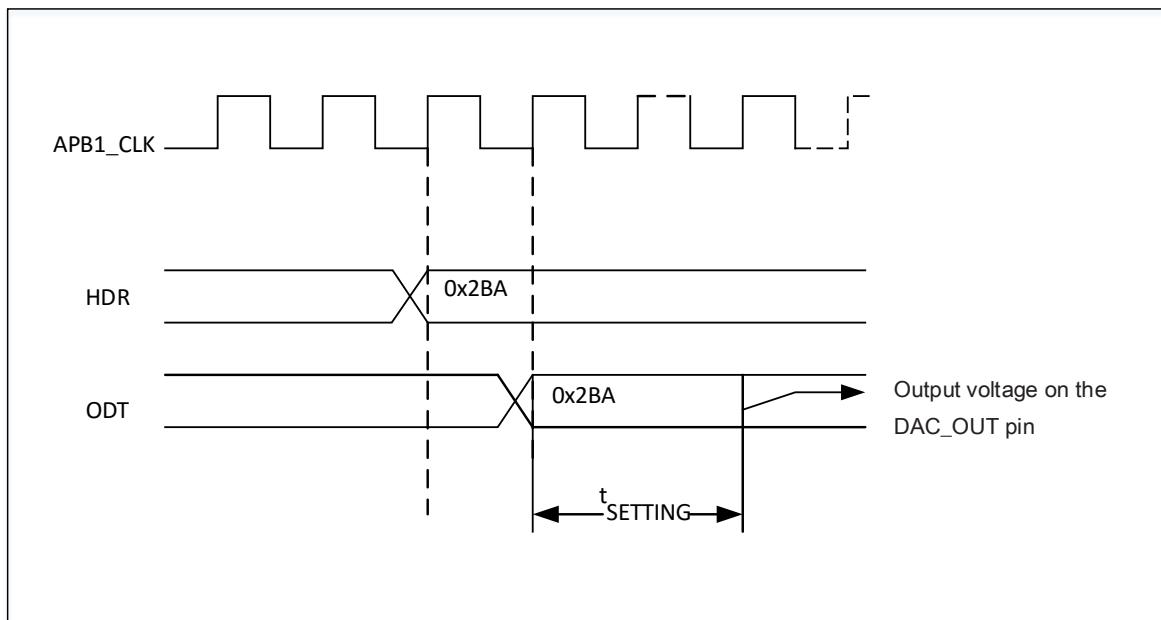
14.3.4 DAC Conversion

The DAC_ODT_x register cannot be written with data directly, and any data outputted to the DAC channel x must be written to the DAC_HDR_x register (actually written into the DAC_HDR8Rx , DAC_HDR12Lx , DAC_HDR12Rx , DAC_HDR8RD , DAC_HDR12LD , or DAC_HDR12RD registers).

Data stored into the DAC_HDRx register will be automatically transferred to the DAC_ODTx register after one APB1 clock cycle if no hardware trigger is selected (the TENx bit in DAC_CTRL register is '0'). However, when a hardware trigger is selected (the TENx bit in DAC_CTRL register is set), and a trigger occurs, the transfer of data is performed after three APB1 clock cycles.

When DAC_HDRx is loaded with the DAC_ODTx contents, the output becomes available after a time of $t_{SETTLING}$, of which length varies with different power supply voltages and the analog output loads.

Figure 14-4 Timing Diagram for Conversion with Trigger Disabled, TEN = 0



14.3.5 DAC Output Voltage

Digital inputs are converted to analog output voltage linearly, and its range is between 0 and V_{REF+} . The analog output voltages on any DAC channel pin are determined by the following equation:

$$\text{DAC output} = V_{REF} \times (ODT/4095)$$

14.3.6 DAC Trigger Selection

If the TENx bit is set, DAC conversion can then be triggered by an external event (timer counter, external interrupt line). The TGSELx[2:0] control bits can select 1 event out of 8 possible events to trigger conversion.

Table 14-2 External Triggers

Source	Type	TGSELx[2:0]
Timer 6 TRGO event	Internal signal from on-chip timer	000
Timer 8 TRGO event		001
Timer 7 TRGO event		010
Timer 5 TRGO event		011
Timer 2 TRGO event		100
Timer 4 TRGO event		101
EXTI line 9	External pins	110
SWTRG (software trigger)	Software control bit	111

Each time a DAC interface detects a rising edge on the selected timer TRGO output, or on the external interrupt line 9, the last data stored into the DAC_HDRx register is transferred into the DAC_ODTx register. The DAC_ODTx register is updated after three APB1 cycles.

If the software trigger is selected, the conversion starts once the SWTRG bit is set. SWTRG is cleared automatically by hardware after the data is transferred from the DAC_HDRx register to the DAC_ODTx register.

- Note:**
1. *TGSELx[2:0] bit cannot be changed when the CHENx bit is set.*
 2. *When software trigger is selected, it takes only one APB1 clock cycle for data to be transferred from the DAC_HDRx register to the DAC_ODTx register.*

14.3.7 DMA Request

Each DAC channel has DMA capability. Two DMA channels are used for DMA requests of the two DAC channels. A DMA request is generated when an external trigger occurs while the DMAENx bit is set. The data of the DAC_HDRx register is then transferred to the DAC_ODTx register.

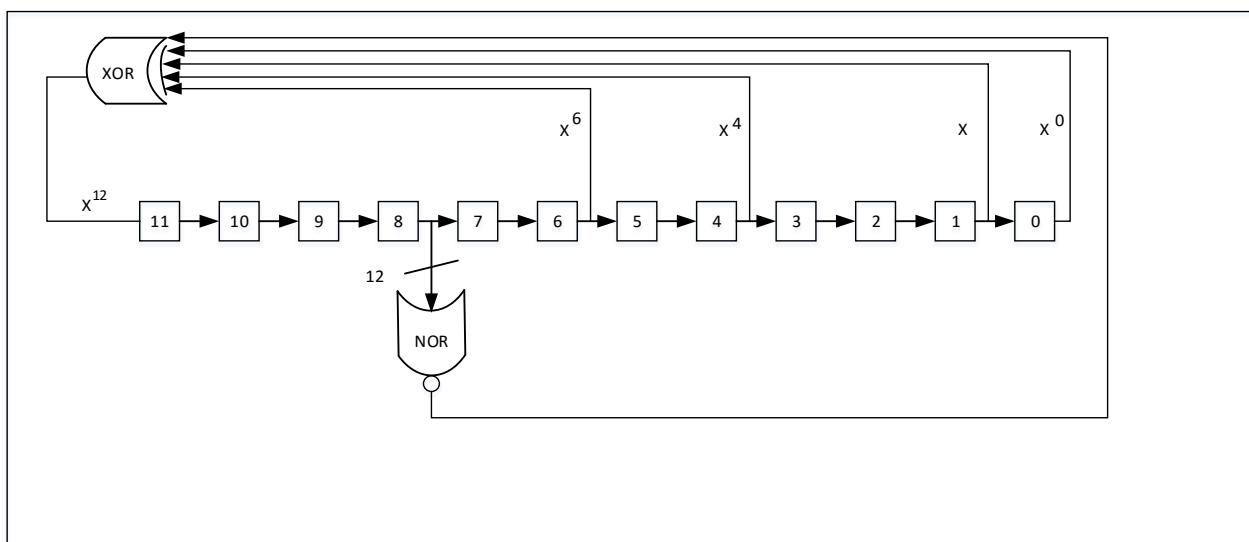
In dual DAC mode, if the DMAENx bits are set in both channels, two DMA requests are generated. If only one DMA request is needed, users should set only one of the corresponding DMAENx bit. In this way, the application can manage both DAC channels in dual mode by using one DMA request and one DMA channel.

DMA requests of DAC are not queued so that if a second external trigger arrives before the acknowledgement of the first request, the second request will not be taken, and no error will be reported.

14.3.8 Noise Generation

The linear feedback shift register (LFSR) can generate a variable-amplitude pseudo noise. The DAC noise generation is selected by setting WAVE[1:0] to "01". The preloaded value in LFSR is 0xAAA. Following a specific calculation algorithm, the register value is updated three APB1 clock cycles later after each trigger event.

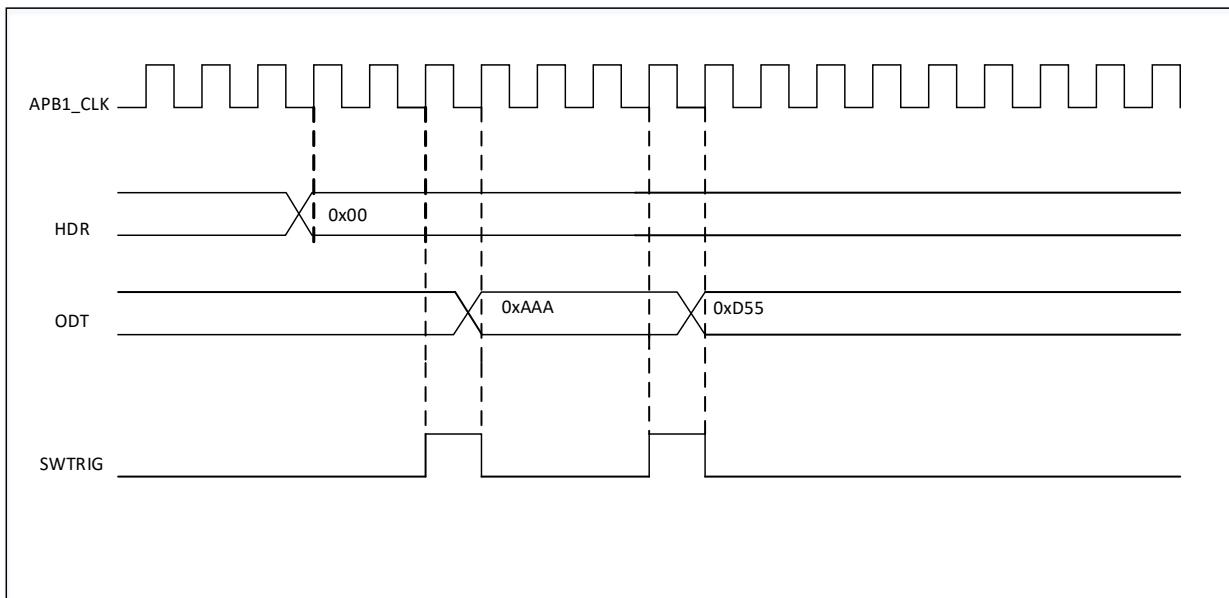
Figure 14-5 DAC LFSR Register Calculation Algorithm



The LFSR value, masked partially or completely through the MAMSx [3:0] bits in the DAC_CTRL register, is added to DAC_HDRx value, subtracted the overflow bit, and then written into the DAC_ODTx register.

If LFSR is 0x0000, a '1' is injected into it (anti-lock mechanism). It is possible to reset LFSR wave generation algorithm by resetting the WAVE[1:0] bits.

Figure 14-6 DAC Conversion (Software Trigger Enabled) with LFSR Wave Generation



Note: For noise generation, DAC trigger must be enabled by setting the *TENx* bit in the *DAC_CTRL* register.

14.3.9 Triangle Wave Generation

It is possible to add a small-amplitude triangular waveform on a DC or a slowly varying signal. DAC triangle wave generation is selected by setting *WAVE_x[1:0]* to '10'. The amplitude is configured through the *MAMS_x[3:0]* bits in the *DAC_CTRL* register. After each trigger event, the internal triangle counter is incremented with 1 after three APB1 clock cycles. The value of this counter is then added to the *DAC_HDR_x* register, subtracted overflow bit, and the written into the *DAC_ODT_x* register. The triangle counter is gradually incremented when the value transferred to the *DAC_ODT_x* registers is less than the maximum amplitude defined by the *MAMS_x[3:0]* bits. Once the configured amplitude is reached, the counter is decremented down to 0, then incremented again, and so on.

Triangle wave generation is reset by resetting the *WAVE_x[1:0]* bits.

Figure 14-7 DAC Triangle Wave Generation

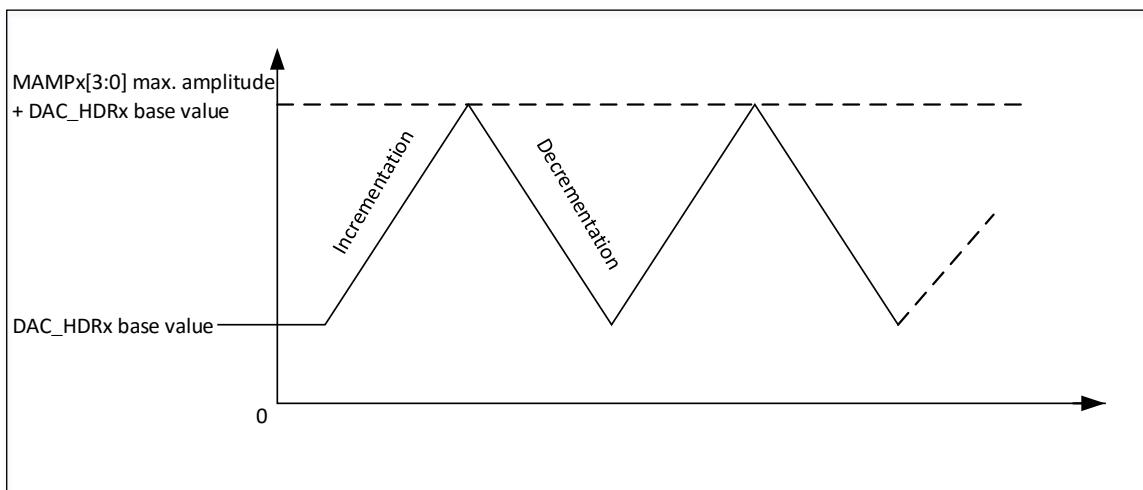
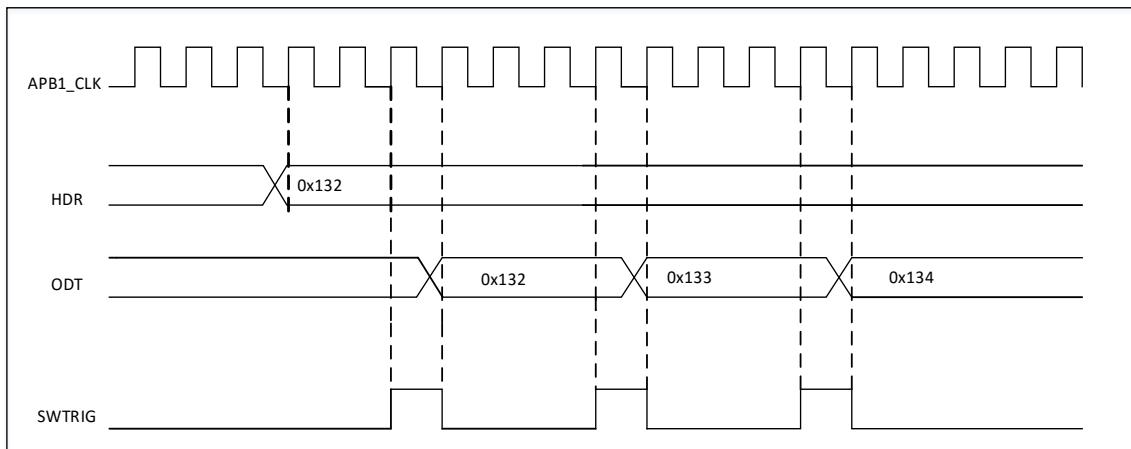


Figure 14-8 DAC Conversion (Software Trigger Enabled) with Triangle Wave Generation



Note:

1. For triangular wave generation, DAC trigger must be enabled by setting the TENx bit in the DAC_CTRL register.
2. MAMS[3:0] bits must be configured before enabling the DAC; otherwise, their values cannot be changed.

14.4 Dual DAC Channel Conversion

To efficiently use the bus bandwidth in applications that require two DACs to work at the same time, the DAC implements three registers for dual mode: HDR8RD, HDR12RD, and HDR12LD. Only one access to the register is required to drive both DAC channels at the same time. Eleven possible conversion modes are available for the two DAC channel conversion and these dedicated registers. When only one DAC channel is used, all these conversion modes can still be operated through the independent HDRx register. All the modes are described in the sections below.

14.4.1 Independent Trigger without Wave Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure different trigger sources for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as different values.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a DAC channel 1 trigger arrives, the value of HDR1 register is transferred into DAC_ODT1 (after three APB1 clock cycles).

When a DAC channel 2 trigger arrives, the value of HDR2 register is transferred into DAC_ODT2 (after three APB1 clock cycles).

14.4.2 Independent Trigger with Same LFSR Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure different trigger sources for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as different values.
- Configure the two DAC channel WAVEx[1:0] bits as '01' and set MAMSx[3:0] as the same LFSR mask value.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter value with the same mask is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR2 counter value with the same mask is added to the HDR2 register value, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then,

the LFSR2 counter is updated.

14.4.3 Independent Trigger with Different LFSR Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure different trigger sources for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as different values.
- Configure the two DAC channel WAVEx[1:0] bits as '01' and set MAMSx[3:0] as different LFSR mask values.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a DAC channel 1 trigger arrives, the LFSR1 counter value, with the mask configured by MAMS1 [3:0], is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then the LFSR1 counter is updated.

When a DAC channel 2 trigger arrives, the LFSR 2 counter value, with the mask configured by MAMS2 [3:0], is added to the HDR2 register value, and the sum is transferred into DAC_ODT2(after three APB1 clock cycles). Then, the LFSR2 counter is updated.

14.4.4 Independent Trigger with Same Triangle Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure different trigger sources for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as different values.
- Configure the two DAC channel WAVEx[1:0] bits as '1x' and set MAMSx[3:0] as the same triangle wave amplitude value.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a DAC channel 1 trigger arrives, the same triangle amplitude value is added to the HDR1 register, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the DAC channel1 triangle counter is updated.

When a DAC channel 2 trigger arrives, the same triangle amplitude value is added to the HDR2 register, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the DAC channel2 triangle counter is updated.

14.4.5 Independent Trigger with Different Triangle Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure different trigger sources for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as different values.
- Configure the two DAC channel WAVEx[1:0] bits as '1x' and set MAMSx[3:0] as different triangle wave amplitude values.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a DAC channel 1 trigger arrives, the triangle amplitude value set by MAMS1[3:0] is added to the HDR1 register, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the DAC channel1 triangle counter is updated.

When a DAC channel 2 trigger arrives, the triangle amplitude value set by MAMS2[3:0] is added to the HDR2 register, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the DAC channel2 triangle counter is updated.

14.4.6 Simultaneous Software Start

Configure the DAC in this mode with the following sequence:

- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

In this configuration, after one APB1 clock cycle, HDR1 and HDR2 register values are transferred into the DAC_ODT1 and DAC_ODT2 registers, respectively.

14.4.7 Simultaneous Trigger without Wave Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure the same trigger source for both DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as the same value.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a trigger arrives, (after three APB1 clock cycles), HDR1 and HDR2 register values are transferred into the DAC_ODT1 and DAC_ODT2 registers, respectively.

14.4.8 Simultaneous Trigger with Same LFSR Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure the same trigger source for both DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as the same value.
- Configure the two DAC channel WAVEx[1:0] bits as '01' and set MAMSx[3:0] as the same LFSR mask value.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a trigger arrives, the LFSR1 counter value with mask set by MAMS1[3:0] is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the LFSR1 counter is updated.

Likewise, the LFSR2 counter value with mask set by MAMS2[3:0] is added to the HDR2 register value, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the LFSR2 counter is updated.

14.4.9 Simultaneous Trigger with Different LFSR Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure the same trigger source for both DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits as the same value.
- Configure the two DAC channel WAVEx[1:0] bits as '01' and set MAMSx[3:0] as different LFSR mask values.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a trigger arrives, the LFSR1 counter value with the same mask is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the LFSR1 counter is updated. At the same time, the LFSR2 counter value with the same mask is added to the HDR2 register value, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the LFSR2 counter is updated.

14.4.10 Simultaneous Trigger with Same Triangle Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure the same trigger source for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits with the same value.
- Configure the two DAC channel WAVEx[1:0] bits as '1x' and set MAMSx[3:0] as

the same triangle wave amplitude value.

- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a trigger arrives, the same triangle amplitude value is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the LFSR1 counter is updated.

At the same time, the same triangle amplitude value is added to the HDR2 register value, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the LFSR2 counter is updated.

14.4.11 Simultaneous Trigger with Different Triangle Generation

Configure the DAC in this mode with the following sequence:

- Set the two DAC channel trigger enable bits, TEN1 and TEN2.
- Configure the same trigger source for the two DAC channels by setting the TGSL1[2:0] and TGSL2[2:0] bits with the same value.
- Configure the two DAC channel WAVEx[1:0] bits as ‘1x’ and set MAMSx[3:0] as different triangle wave amplitude values.
- Load the dual DAC channel data into the desired HDR register (HDR12RD, HDR12LD, or HDR8RD).

When a trigger arrives, the triangle amplitude value set by MAMS1[3:0] is added to the HDR1 register value, and the sum is transferred into DAC_ODT1 (after three APB1 clock cycles). Then, the LFSR1 counter is updated.

At the same time, the triangle amplitude value set by MAMS2[3:0] is added to the HDR2 register value, and the sum is transferred into DAC_ODT2 (after three APB1 clock cycles). Then, the LFSR2 counter is updated.

14.5 DAC Registers

These peripheral registers must be accessed by words (32-bit). Table 14-3 lists all the DAC registers.

Table 14-3 DAC Register Map

Offset	register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
000h	DAC_CTRL	Reserve d	DMAE	MAMS2 [3:0]			WAVE2 [1:0]	TGSEL2 [2:0]		TEN2	BFF2	EN2	Reserve d	DMAE	MAMS1 [3:0]			WAVE1 [1:0]	TGSEL1 [2:0]		TEN1	BFF1	EN1	0	0	0	0	0	0	0	0	0	0			
	Reset Value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
004h	DAC_SWTR G	Reserved																											SWTRG2	SWTRG1	0	0				
	Reset Value																																			
008h	DAC_HDR1 2R1	Reserved												D1HDR[11:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
00Ch	DAC_HDR1 2L1	Reserved												D1HDR[11:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
010h	DAC_HDR8 R1	Reserved												D1HDR[7:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
014h	DAC_HDR1 2R2	Reserved												D2HDR[11:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
018h	DAC_HDR1 2L2	Reserved												D2HDR[11:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
01Ch	DAC_HDR8 R2	Reserved												D2HDR[7:0]																						
	Reset Value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

020h	DAC_HDR1 2RD	Reserved	D2HDR[11:0]								Reserved	D1HDR[11:0]								
	Reset Value		0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0									0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								
024h	DAC_HDR1 2LD	D2HDR[11:0]								Reserved	D1HDR[11:0]								Reserved	
	Reset Value	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0									0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0									
028h	HDR8RD	Reserved								D2HDR[7:0]								D1HDR[7:0]		
	Reset Value									0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
02Ch	DAC_ODT1	Reserved								D1ODT[11:0]								0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
	Reset Value									0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
030h	DAC_ODT2	Reserved								D2ODT[11:0]								0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
	Reset Value									0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0								0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		

14.5.1 DAC Control Register (DAC_CTRL)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
Reserved	DMA EN2	MAMS2[3:0]				WAVE2[2:0]				TGSL2[2:0]				TEN2	BFF2	EN2					
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		0					
Reserved	DMA EN1	MAMS1[3:0]				WAVE1[2:0]				TGSL1[2:0]				TEN1	BFF1	EN1					
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		
Bit 31:29					Reserved.																
Bit 28					DMAEN2: DAC channel2 DMA enable This bit is set and cleared by software. 0: DAC channel 2 DMA mode is disabled. 1: DAC channel 2 DMA mode is enabled.																
Bit 27:24					MAMS2[3:0]: DAC channel2 mask/amplitude selector These bits are written by software to select mask in noise generation mode/amplitude in triangle wave generation mode. 0000: LSFR bit 0 is not masked/Triangle wave amplitude is 1. 0001: LSFR bit [1:0] is not masked/Triangle wave amplitude is 3. 0010: LSFR bit [2:0] is not masked/Triangle wave amplitude is 7. 0011: LSFR bit [3:0] is not masked/Triangle wave amplitude is 15. 0100: LSFR bit [4:0] is not masked/Triangle wave amplitude is 31. 0101: LSFR bit [5:0] is not masked/Triangle wave amplitude is 63. 0110: LSFR bit [6:0] is not masked/Triangle wave amplitude is 127. 0111: LSFR bit [7:0] is not masked/Triangle wave amplitude is 255. 1000: LSFR bit [8:0] is not masked/Triangle wave amplitude is 511. 1001: LSFR bit [9:0] is not masked/Triangle wave amplitude is 1023. 1010: LSFR bit [10:0] is not masked/Triangle wave amplitude is 2047. ≥1011: LSFR bit [11:0] is not masked/Triangle wave amplitude is 4095.																
Bit 23:22					WAVE2[1:0]: DAC channel2 noise/triangle wave generation enable The 2 bits are set and cleared by software. 00: Wave generation is disabled. 10: Noise wave generation is enabled. 1x: Triangle wave generation is enabled.																
Bit 21:19					TGSL2[2:0]: DAC channel2 trigger selection These bits select the external trigger event for DAC channel 2. 000: TMR6 TRGO event 001: TMR8 TRGO event 010: TMR7 TRGO event 011: TMR5 TRGO event 100: TMR2 TRGO event 101: TMR4 TRGO event 110: External interrupt line 9 111: Software trigger Note: These bits can be set only when TEN2 = 1 (DAC channel 2 trigger is enabled).																

Bit 18	TEN2: DAC channel2 trigger enable This bit is set and cleared by software to enable/disable DAC channel 2 trigger. 0: DAC channel 2 trigger is disabled. Data written into the DAC_HDRx register is transferred to the DAC_ODT2 register after one APB1 clock cycle. 1: DAC channel 2 trigger is enabled. Data written into the DAC_HDRx register is transferred to the DAC_ODT2 register after three APB1 clock cycles. Note: When software trigger is selected, it takes only one APB1 clock cycle for data written to DAC_HDRx to be transferred to DAC_ODT2 .
Bit 17	BFF2: DAC channel2 output buffer disable This bit is set and cleared by software to enable/disable DAC channel 2 output buffer. 0: DAC channel 2 output buffer is enabled. 1: DAC channel 2 output buffer is disabled.
Bit 16	EN2: DAC channel2 enable This bit is set and cleared by software to enable/disable DAC channel 2. 0: DAC channel 2 is disabled. 1: DAC channel 2 is enabled.
Bit 15:13	Reserved.
Bit 12	DMAEN1: DAC channel1 DMA enable This bit is set and cleared by software. 0: DAC channel 1 DMA mode is disabled. 1: DAC channel 1 DMA mode is enabled.
Bit 11:8	MAMS1[3:0]: DAC channel1 mask/amplitude selector These bits are written by software to select mask in noise generation mode/amplitude in triangle wave generation mode. 0000: LSFR bit 0 is not masked/Triangle wave amplitude is 1. 0001: LSFR bit [1:0] is not masked/Triangle wave amplitude is 3. 0010: LSFR bit [2:0] is not masked/Triangle wave amplitude is 7. 0011: LSFR bit [3:0] is not masked/Triangle wave amplitude is 15. 0100: LSFR bit [4:0] is not masked/Triangle wave amplitude is 31. 0101: LSFR bit [5:0] is not masked/Triangle wave amplitude is 63. 0110: LSFR bit [6:0] is not masked/Triangle wave amplitude is 127. 0111: LSFR bit [7:0] is not masked/Triangle wave amplitude is 255. 1000: LSFR bit [8:0] is not masked/Triangle wave amplitude is 511. 1001: LSFR bit [9:0] is not masked/Triangle wave amplitude is 1023. 1010: LSFR bit [10:0] is not masked/Triangle wave amplitude is 2047. ≥1011: LSFR bit [11:0] is not masked/Triangle wave amplitude is 4095.
Bit 7:6	WAVE1[1:0]: DAC channel1 noise/triangle wave generation enable The 2 bits are set and cleared by software. 00: Wave generation is disabled. 01: Noise wave generation is enabled. 1x: Triangle wave generation is enabled.
Bit 5:3	TGSL1[2:0]: DAC channel1 trigger selection These bits select the external trigger event for DAC channel 1. 000: TMR6 TRGO event 001: TMR8 TRGO event 010: TMR7 TRGO event 011: TMR5 TRGO event 100: TMR2 TRGO event 101: TMR4 TRGO event 110: External interrupt line 9 111: Software trigger Note: These bits can be set only when TEN1 = 1 (DAC channel 1 trigger is enabled).
Bit 2	TEN1: DAC channel1 trigger enable This bit is set and cleared by software to enable/disable DAC channel 1 trigger. 0: DAC channel 1 trigger is disabled. Data written into the DAC_HDRx register is transferred to the DAC_ODT1 register after one APB1 clock cycle. 1: DAC channel 1 trigger is enabled. Data written into the DAC_HDRx register is transferred to the DAC_ODT1 register after three APB1 clock cycles. Note: When software trigger is selected, it takes only one APB1 clock cycle for data written to DAC_HDRx to be transferred to DAC_ODT1 .

Bit 1	BFF1: DAC channel1 output buffer disable This bit is set and cleared by software to enable/disable DAC channel 1 output buffer. 0: DAC channel 1 output buffer is enabled. 1: DAC channel 1 output buffer is disabled.
Bit 0	EN1 DAC channel1 enable This bit is set and cleared by software to enable/disable DAC channel 1. 0: DAC channel 1 is disabled. 1: DAC channel 1 is enabled.

14.5.2 DAC Software Trigger Register (DAC_SWTRG)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved																	
res																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved												SWT RG2	SWT RG1				
res																	
Bit 31:2		Reserved.															
Bit 1		SWTRG2: DAC channel2 software trigger This bit is set and cleared by software to enable/disable software trigger. 0: DAC channel 2 software trigger is disabled. 1: DAC channel 2 software trigger is enabled. Note: This bit is reset by hardware once the DAC_HDR2 register data is transferred to the DAC_ODT2 register (after one APB1 clock cycle).															
Bit 0		SWTRG1: DAC channel1 software trigger This bit is set and cleared by software to enable/disable software trigger. 0: DAC channel 1 software trigger is disabled. 1: DAC channel 1 software trigger is enabled. Note: This bit is reset by hardware once the DAC_HDR1 register data is transferred to the DAC_ODT1 register (after one APB1 clock cycle).															

14.5.3 DAC Channel 1 12-bit Right-aligned Data Holding Register (DAC_HDR12R1)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		D1HDR[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved													
Bit 11:0		D1HDR[11:0]: DAC channel1 12-bit right-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.													

14.5.4 DAC Channel 1 12-bit Left-aligned Data Holding Register (DAC_HDR12L 1)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1HDR[11:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res
Bit 31:16		Reserved													
Bit 15:4		D1HDR[11:0]: DAC channel1 12-bit left-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.													
Bit 3:0		Reserved.													

14.5.5 DAC Channel 1 8-bit Right-aligned Data Holding Register (DAC_HDR8R1)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D1HDR[7:0]							
res								rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:18		Reserved													
Bit 7:0		D1HDR[7:0]: DAC channel1 8-bit right-aligned data This bit is written by software to indicate DAC channel 1 8-bit data.													

14.5.6 DAC Channel 2 12-bit Right-aligned Data Holding Register (DAC_HDR12 R2)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								D2HDR[11:0]							
res								rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved													

Bit 11:0

D2HDR[11:0]: DAC channel2 12-bit right-aligned data
This bit is written by software to indicate DAC channel 2 12-bit data.

14.5.7 DAC Channel 2 12-bit Left-aligned Data Holding Register (DAC_HDR12L_2)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D2HDR[11:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res
Bit 31:16															
Reserved															
Bit 15:4															
DAC2HDR[11:0]: DAC channel2 12-bit left-aligned data This bit is written by software to indicate DAC channel 2 12-bit data.															
Bit 3:0															
Reserved															

14.5.8 DAC Channel 2 8-bit Right-aligned Data Holding Register (DAC_HDR8R2)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
D2HDR[7:0]															
res															
rw															
Bit 31:18															
Reserved															
Bit 7:0															
D2HDR[7:0]: DAC channel2 8-bit right-aligned data This bit is written by software to indicate DAC channel 2 8-bit data.															

14.5.9 Dual DAC 12-bit Right-aligned Data Holding Register (DAC_HDR12RD)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		D2HDR[11:0]													
res		RW													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		D1HDR[11:0]													

res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:28	Reserved.													
Bit 27:16	D2HDR[11:0]: DAC channel2 12-bit right-aligned data This bit is written by software to indicate DAC channel 2 12-bit data.													
Bit 15:12	Reserved.													
Bit 11:0	D1HDR[11:0]: DAC channel1 12-bit right-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.													

14.5.10 Dual DAC 12-bit Left-aligned Data Holding Register (DAC_HDR12LD)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																																																																		
D2HDR[11:0]														Reserved																																																																			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res																																																																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																																		
D1HDR[11:0]														Reserved																																																																			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res																																																																			
<table border="1"> <tr> <td>Bit 31:20</td> <td colspan="15">D2HDR[11:0]: DAC channel2 12-bit left-aligned data This bit is written by software to indicate DAC channel 2 12-bit data.</td></tr> <tr> <td>Bit 19:16</td> <td colspan="15">Reserved.</td></tr> <tr> <td>Bit 15:4</td> <td colspan="15">D1HDR[11:0]: DAC channel1 12-bit left-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.</td></tr> <tr> <td>Bit 3:0</td> <td colspan="15">Reserved.</td></tr> </table>																Bit 31:20	D2HDR[11:0]: DAC channel2 12-bit left-aligned data This bit is written by software to indicate DAC channel 2 12-bit data.															Bit 19:16	Reserved.															Bit 15:4	D1HDR[11:0]: DAC channel1 12-bit left-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.															Bit 3:0	Reserved.																
Bit 31:20	D2HDR[11:0]: DAC channel2 12-bit left-aligned data This bit is written by software to indicate DAC channel 2 12-bit data.																																																																																
Bit 19:16	Reserved.																																																																																
Bit 15:4	D1HDR[11:0]: DAC channel1 12-bit left-aligned data This bit is written by software to indicate DAC channel 1 12-bit data.																																																																																
Bit 3:0	Reserved.																																																																																

14.5.11 Dual DAC 8-bit Right-aligned Data Holding Register (DAC_HDR8RD)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16																																																
Reserved																																																															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																																
<table border="1"> <tr> <td colspan="8">D2HDR[7:0]</td><td colspan="8">D1HDR[7:0]</td></tr> <tr> <td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td><td>rw</td></tr> </table>														D2HDR[7:0]								D1HDR[7:0]								rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw																		
D2HDR[7:0]								D1HDR[7:0]																																																							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw																																																
<table border="1"> <tr> <td>Bit 31:16</td> <td colspan="15">Reserved.</td></tr> <tr> <td>Bit 15:8</td> <td colspan="15">D2HDR[7:0]: DAC channel2 8-bit right-aligned data This bit is written by software to indicate DAC channel 2 8-bit data.</td></tr> <tr> <td>Bit 7:0</td> <td colspan="15">D1HDR[7:0]: DAC channel1 8-bit right-aligned data This bit is written by software to indicate DAC channel 1 8-bit data.</td></tr> </table>																Bit 31:16	Reserved.															Bit 15:8	D2HDR[7:0]: DAC channel2 8-bit right-aligned data This bit is written by software to indicate DAC channel 2 8-bit data.															Bit 7:0	D1HDR[7:0]: DAC channel1 8-bit right-aligned data This bit is written by software to indicate DAC channel 1 8-bit data.														
Bit 31:16	Reserved.																																																														
Bit 15:8	D2HDR[7:0]: DAC channel2 8-bit right-aligned data This bit is written by software to indicate DAC channel 2 8-bit data.																																																														
Bit 7:0	D1HDR[7:0]: DAC channel1 8-bit right-aligned data This bit is written by software to indicate DAC channel 1 8-bit data.																																																														

14.5.12 DAC Channel 1 Data Output Register (DAC_ODT1)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		D1ODT[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved.													
Bit 11:0		D1ODT[11:0]: DAC channel1 data output This bit is written by software to indicate DAC channel 1 output data.													

14.5.13 DAC Channel 2 Data Output Register (DAC_ODT2)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		D2ODT[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:12		Reserved.													
Bit 11:0		D2ODT[11:0]: DAC channel2 data output This bit is written by software to indicate DAC channel 2 output data.													

15 I²C Interface

15.1 I²C Introduction

I²C (inter-integrated circuit) bus interface serves as an interface between the microcontroller and the serial I²C bus. It provides multimaster capability and controls all I²C bus-specific sequencing, protocol, arbitration, and timing. It supports standard mode and fast mode, and is compatible with SMBus 2.0. I²C may be used for a variety of purposes, including CRC generation and verification, SMBus (system management bus), and PMBus (power management bus). According to the needs of specific devices, DMA can be utilized to reduce CPU overload.

15.2 I²C Main Features

- Parallel-bus/I²C protocol converter
- Multimaster capability: The same interface can serve as master or slave.
- I²C Master features
 - Clock generation
 - Start and Stop signals generation
- I²C Slave features
 - Programmable I²C address detection
 - Dual addressing capability to acknowledge 2 slave addresses
 - Stop bit detection
- Generation and detection of 7-bit/10-bit addressing and general call
- Supports different communication speeds
 - Standard speed (Up to 100 kHz)
 - Fast speed (Up to 400 kHz)
- Status flags
 - Transmitter/Receiver mode flag
 - End-of-byte-transmission flag
 - I²C busy flag
- Error flags
 - Arbitration lost in master mode
 - Acknowledgment (ACK) failure after address/data transmission
 - Detection of misplaced Start or Stop conditions
 - Overrun/Underrun (If clock stretching is disabled)
- 2 interrupts vectors
 - 1 interrupt for successful address/data communication
 - 1 interrupt for error
- Optional clock stretching disable
- DMA with 1-byte buffer
- Configurable PEC (packet error checking) generation or verification
 - PEC value can be transmitted as the last byte in Tx mode
 - PEC error checking for the last received byte
- SMBus 2.0 compatibility
 - 25 ms clock low timeout delay
 - 10 ms master cumulative clock low extend time
 - 25 ms slave cumulative clock low extend time
 - Hardware PEC generation/verification with ACK control
 - Support Address Resolution Protocol (ARP)
- PMBus compatibility

15.3 I²C Function Overview

I²C module receives and transmits data, and converts data from serial to parallel format, and vice versa. Interrupts can be enabled or disabled. The interface is connected to the I²C bus through a data pin (SDA) and a clock pin (SCL). It is allowed to connect to a standard (up to 100 kHz) or a fast (up to 400 kHz) I²C bus.

15.3.1 Mode Selection

The interface can operate in one of the following four modes:

- Slave transmitter mode
- Slave receiver mode
- Master transmitter mode
- Master receiver mode

By default, the interface works in slave mode. After generating a Start condition, it automatically switches from slave mode to master mode. If an arbitration is lost or a Stop is generated, it switches from master mode to slave mode. It also has multimaster capability.

Communication flow

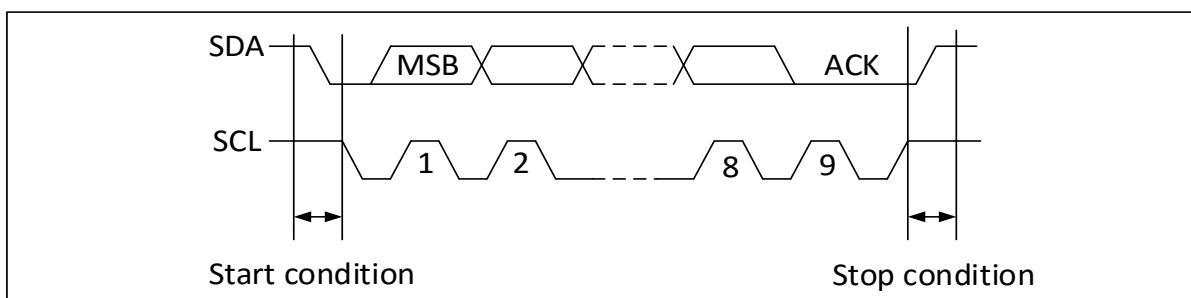
In master mode, the I²C interface initiates data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. Both Start and Stop conditions are generated in master mode by software.

In slave mode, the I²C interface can recognize its own addresses (7-bit or 10-bit) and the general call address. General call address recognition can be enabled or disabled by software.

Data and address are transferred as 8-bit (one byte), MSB first. The first or second byte following the Start condition is the address (one byte in 7-bit mode, two bytes in 10-bit mode). Addresses are always transmitted in master mode.

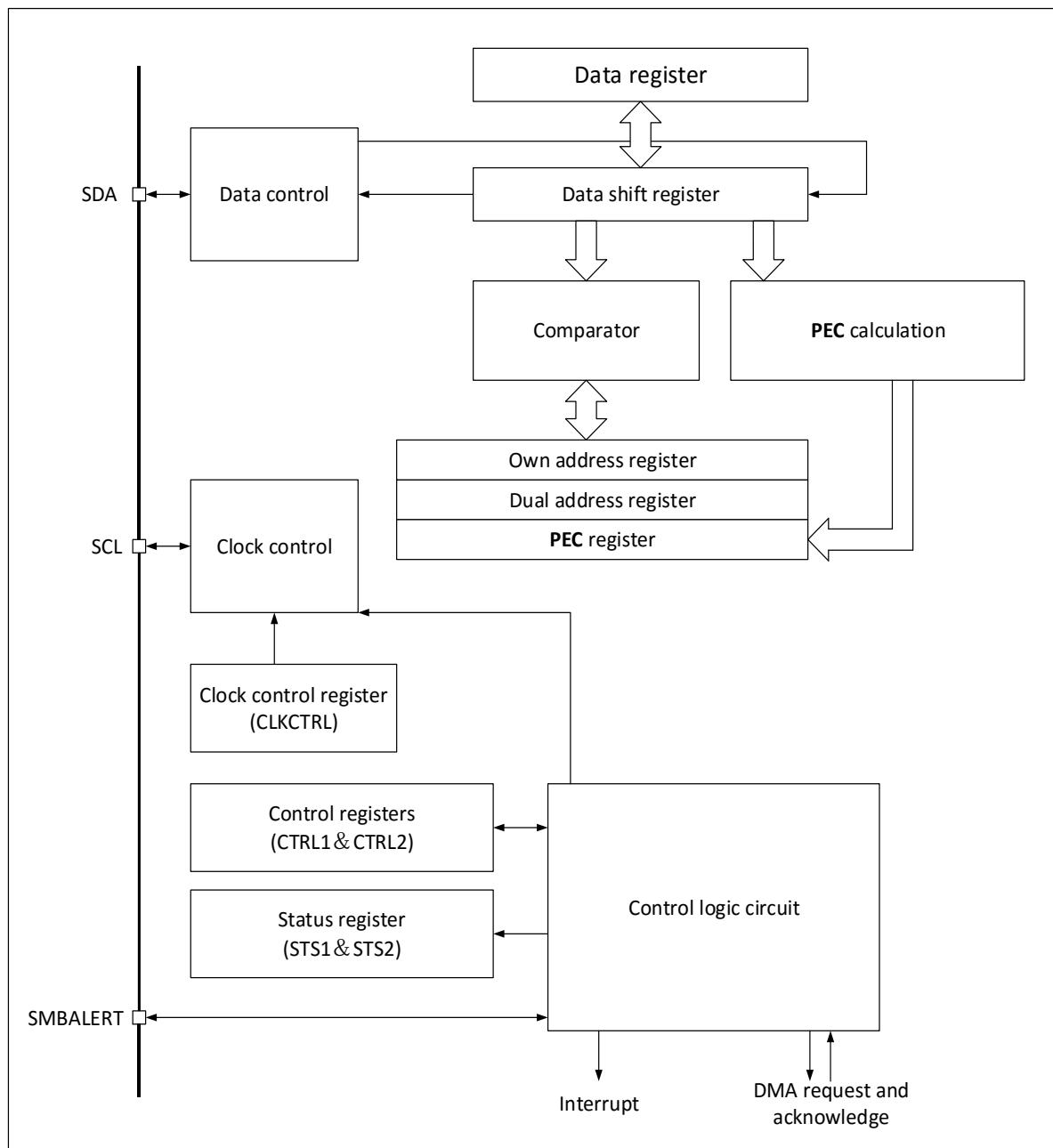
At the 9th clock period following the 8 clock cycles of a byte transfer, the receiver must send an acknowledge bit (ACK) to the transmitter. Please refer to Figure 15-1.

Figure 15-1 I²C Bus Protocol



Acknowledge (ACKEN) can be enabled or disabled by software, which can also set I²C interface addresses (7-bit, 10-bit, or general call address).

The block diagram of the I²C interface is shown in Figure 15-2.

Figure 15-2 Block Diagram of I²C Function

Note: *SMBALERT* is an optional signal in SMBus mode. This signal cannot be used if SMBus is disabled.

15.3.2 I²C Slave Mode

By default, the I²C interface operates in slave mode. To switch from slave mode to master mode, a Start condition generation is needed. In order to generate correct timings, the input clock of the module must be programmed in the I2C_CTRL2 register. The input clock frequency must be at least:

- In standard mode: 2 MHz
- In fast mode: 4 MHz

As soon as a Start condition is detected, the address received from the SDA line is sent to the shift register. Then, it is compared with the address of the MCU, OADDR1 and OADDR2 (if DUALEN = 1), or the general call address (if GCEN = 1).

Note: *In 10-bit addressing mode, the comparison includes the header sequence (11110xx0), where xx denotes the two most significant bits of the address.*

Header or address not matched: The I²C interface ignores it and waits for another Start condition.

Header matched (10-bit mode only): If the ACKEN bit is set, the I²C interface generates an acknowledge pulse and waits for the 8-bit slave address.

Address matched: The I²C interface generates the following sequence:

- An acknowledge pulse is generated if the ACKEN bit is set.
- The ADDRF bit is set by hardware; an interrupt is generated if the EVTITEN bit is set.
- If DUALEN = 1, the software must read the DUALF bit to see which slave address is acknowledged.

In 10-bit mode, after receiving the address sequence, the slave is always in receiver mode. It will enter transmitter mode when receiving a repeated Start condition, followed by the header sequence matching the address, and the least significant bit set (11110xx1).

In slave mode, the TRF bit indicates whether the slave is in receiver or transmitter mode.

Slave transmitter

After the address is received and the ADDRF bit is cleared, the slave transmitter sends bytes from the DT register to the SDA line via the internal shift register.

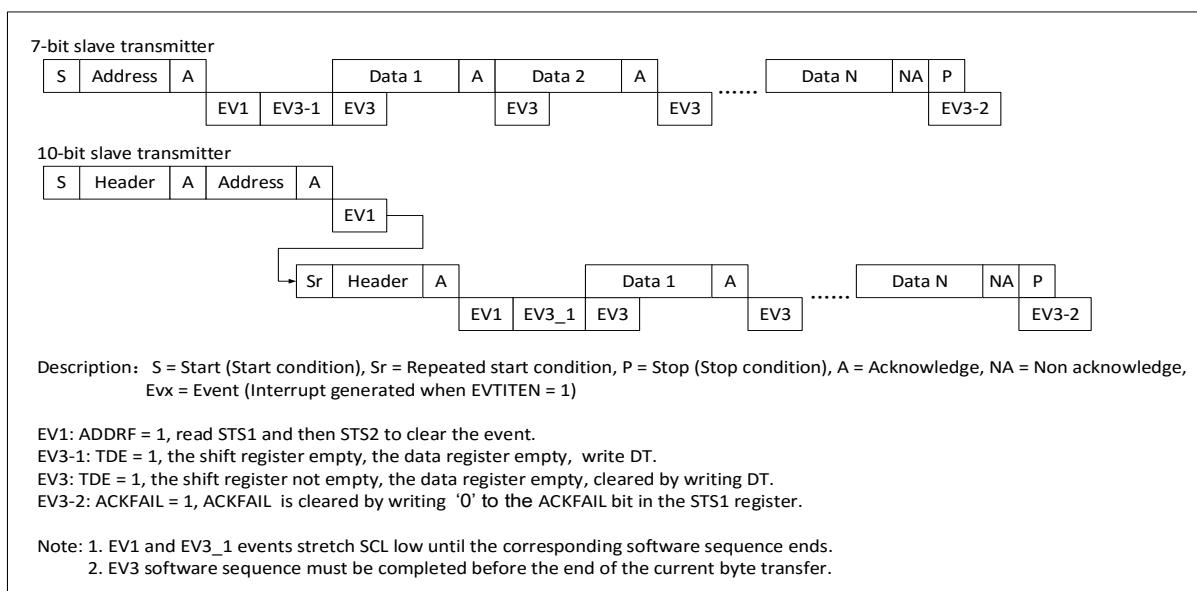
The slave stretches SCL low until the ADDRF bit is cleared, and the data to be sent is written into the DT register. (Please refer to EV1 and EV3 in [Figure 15-3](#).)

When the acknowledge pulse is received:

- The TDE bit is set by hardware. An interrupt is generated if the EVTITEN and BUFITEN bits are set.

If the TDE bit is set, but there is no new data written into the I2C_DT register before the end of the next data transmission, then the BTFF bit is set. The I²C interface stretches SCL low until the BTFF bit is cleared. A read to I2C_STS1 followed by a write to the I2C_DT register will clear the BTFF bit.

Figure 15-3 Transfer Sequence Diagram of Slave Transmitter



Slave receiver

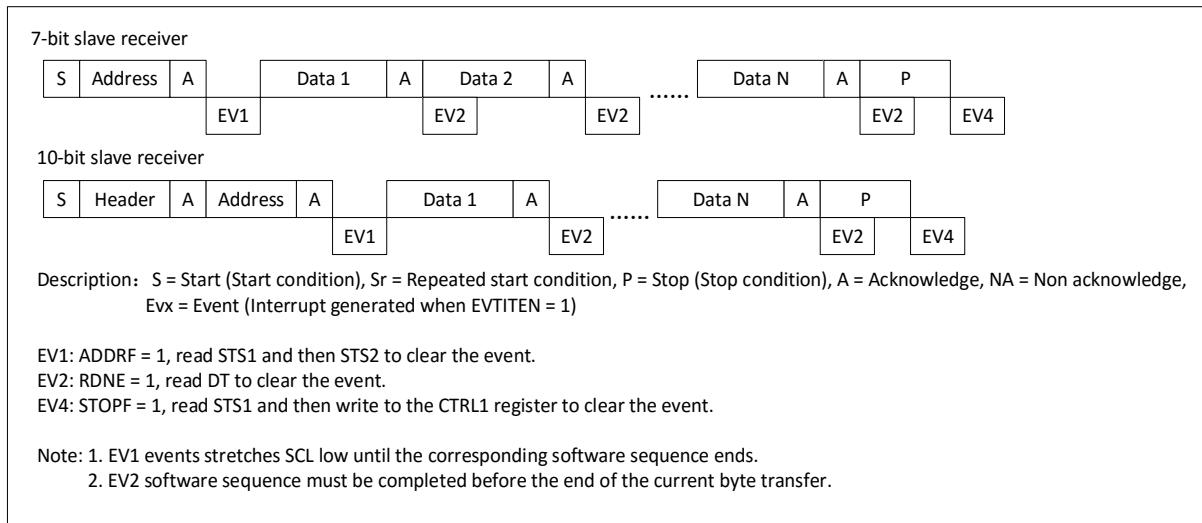
After the address is received and the ADDRF bit is cleared, the slave receiver stores bytes received from the SDA line via the internal shift register into the DT register.

The I²C interface operates as follows after each byte reception:

- An acknowledge pulse is generated if the ACKEN bit is set.
- Set RDNE = 1 by hardware. An interrupt is generated if the EVTITEN and BUFITEN bits are set.

If RDNE is set, and the data in the DT register is not read before the end of the next data reception, then the BTFF bit is set. The I²C interface stretches SCL low until BTFF is cleared. A read to I2C_STS1 followed by a read to the I2C_DT register will clear the BTFF bit.

Figure 15-4 Transfer Sequence Diagram of Slave Receiver



Closing slave communication

After the last data byte is transferred, a Stop condition is generated by the master. When the I²C interface detects this condition:

- Set the STOPF bit = 1 and generate an interrupt if the EVTITEN bit is set.

Then, the I²C interface waits for reading the STS1 register, followed by writing to CTRL1 register. After this operation is completed, the STOPF bit is cleared (Please refer to EV4 in [Figure 15-4](#)).

15.3.3 I²C Master Mode

In master mode, the I²C interface initiates data transfer and generates the clock signal. A serial data transfer always begins with a Start condition and ends with a Stop condition. When a Start condition is generated on the bus through the STARTGEN bit, the device enters master mode.

The sequence required in master mode is as follows:

- Program the input clock in the I2C_CTRL2 register to generate correct timings.
- Configure the clock control registers
- Configure the rise time register
- Program the I2C_CTRL1 register to enable the peripheral
- Set the STARTGEN bit in the I2C_CTRL1 register to generate a Start condition

The frequency of I²C input clock must be at least:

- In standard mode: 2 MHz
- In fast mode: 4 MHz

Master clock generation

CLKCTRL is used to generate the high and low level of the I²C clock, starting from the rising and falling edges of the clock. Since a slave may stretch the SCL line, the peripheral will check the level of the clock line after the maximum rising edge generation time programmed in the TMRISE register.

- If the clock line is low, it means that a slave is stretching the bus, and the high level counter stops until the clock line is detected high. This guarantees the HIGH period of the SCL clock.
- If the clock line is high, the high level counter keeps counting.

Start condition

Set the STARTGEN = 1 when BUSYF = 0, and the I²C interface will generate a Start condition and switch to master mode (the MSF bit is set).

Note: In master mode, setting the STARTGEN bit will generate a re-Start condition by hardware at the end of the current byte transfer.

Once the Start condition is generated:

- The STARTF bit is set by hardware. An interrupt is generated if the EVTITEN bit is set.

Then, the master waits for a read to the STS1 register, followed by writing the slave address into the DT

register (Please refer to EV5 in [Figure 15-5](#) and [Figure 15-6](#)).

Slave address transmission

The slave address is sent to the SDA line via the internal shift register.

- In 10-bit addressing mode, sending the header sequence generates the following events:

- The ADDR10F bit is set by hardware, and an interrupt is generated if the EVTITEN bit is set.

Then, the master waits for a read to the STS1 register, and then writes the second address byte into the DT register (Please refer to [Figure 15-5](#) and [Figure 15-6](#)).

- The ADDRF bit is set by hardware, and an interrupt is generated if the EVTITEN bit is set.

Then, the master waits for a read to the STS1 register, and then a read to the STS2 register (Please refer to [Figure 15-5](#) and [Figure 15-6](#)).

- In 7-bit addressing mode, only one address byte is sent.

As soon as the address byte is sent,

- The ADDRF bit is set by hardware, and an interrupt is generated if the EVTITEN bit is set.

Then, the master waits for a read to the STS1 register, and then a read to the STS2 register (Please refer to [Figure 15-5](#) and [Figure 15-6](#)).

The master can decide to enter transmitter or receiver mode according to the LSB of the slave address sent.

- In 7-bit addressing mode
 - To enter transmitter mode, a master sends the slave address with LSB reset.
 - To enter receiver mode, a master sends the slave address with LSB set.
- In 10-bit addressing mode
 - To enter transmitter mode, a master first sends the header (11110xx0), and then the slave address (where xx denotes the two most significant bits of the 10-bit address).
 - To enter receiver mode, a master first sends the header (11110xx0), and then the slave address. Then, it resends a repeated Start condition followed by the header (11110xx1), (where xx denotes the two most significant bits of the 10-bit address).

The TRF bit indicates whether the master is in receiver or transmitter mode.

Master transmitter

After address is transferred and the ADDRF bit is cleared, the master sends bytes from the DT register to the SDA line via the internal shift register.

The master waits until the data is written into the DT register and TDE is cleared (Please refer to EV8 in [Figure 15-5](#)).

When the acknowledge pulse is received:

- The TDE bit is set by hardware, and an interrupt is generated if the EVTITEN and BUFITEN bits are set.

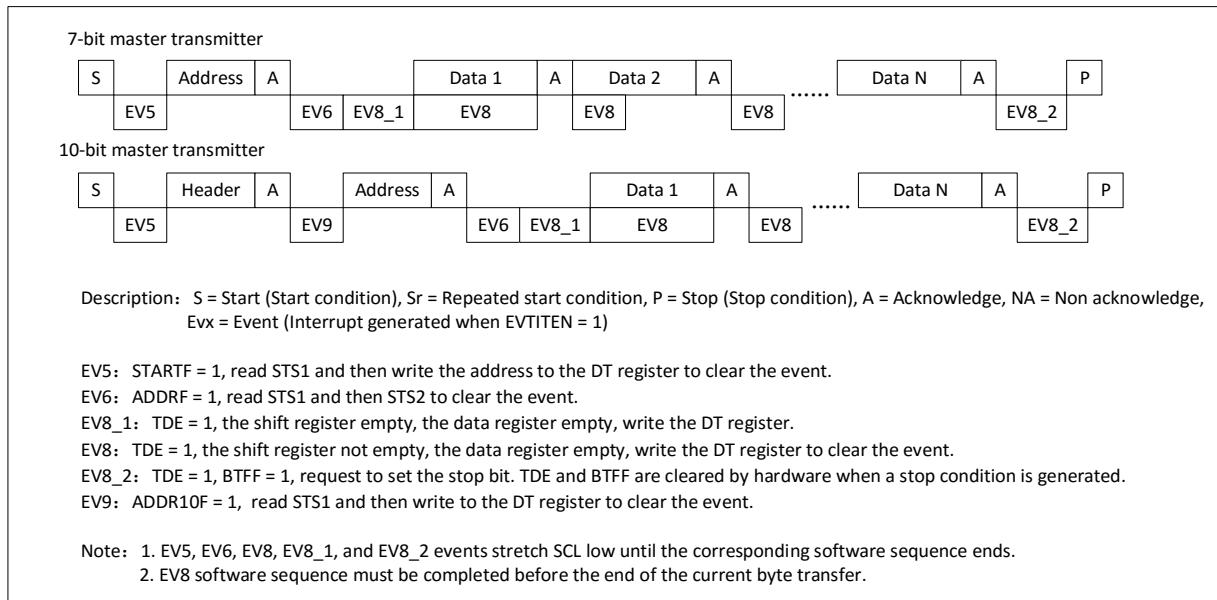
If TDE is set and there is no new data byte written into the DT register before the end of the last data transmission, then BTFF is set by hardware. The interface stretches SCL low until BTFF is cleared. BTFF is cleared by a read to I2C_STS1 followed by a write to I2C_DT.

Closing the communication

After the last byte is written to the DT register, the STOPGEN bit is set to generate a Stop condition (please refer to EV8_2 in [Figure 15-5](#)). Then, the I²C interface automatically returns to slave mode (the MSF bit is cleared).

Note: Stop condition should be programmed during EV8_2 event when either the TDE bit or the BTFF bit is set.

Figure 15-5 Transfer Sequence Diagram of Master Transmitter



Master receiver

After the address is transmitted, and ADDRFF is cleared, the I²C interface enters master receiver mode. In this mode, the I²C interface receives data bytes from the SDA line and sends them into the DT register via the internal shift register. After each byte, the I²C interface generates the following in sequence:

- An acknowledge pulse is sent if the ACKEN bit is set.
- Set RDNE = 1 by hardware. An interrupt is generated if the EVTITEN and BUFITEN bits are set (Please refer to EV7 in [Figure 15-6](#)).

If the RDNE bit is set, and the data in the DT register is not read before the end of the last data reception, the BTFF bit will be set by hardware. The I²C interface stretches SCL low until BTFF is cleared. BTFF is cleared by a read to I2C_STS1 followed by a read to I2C_DT.

Closing the communication

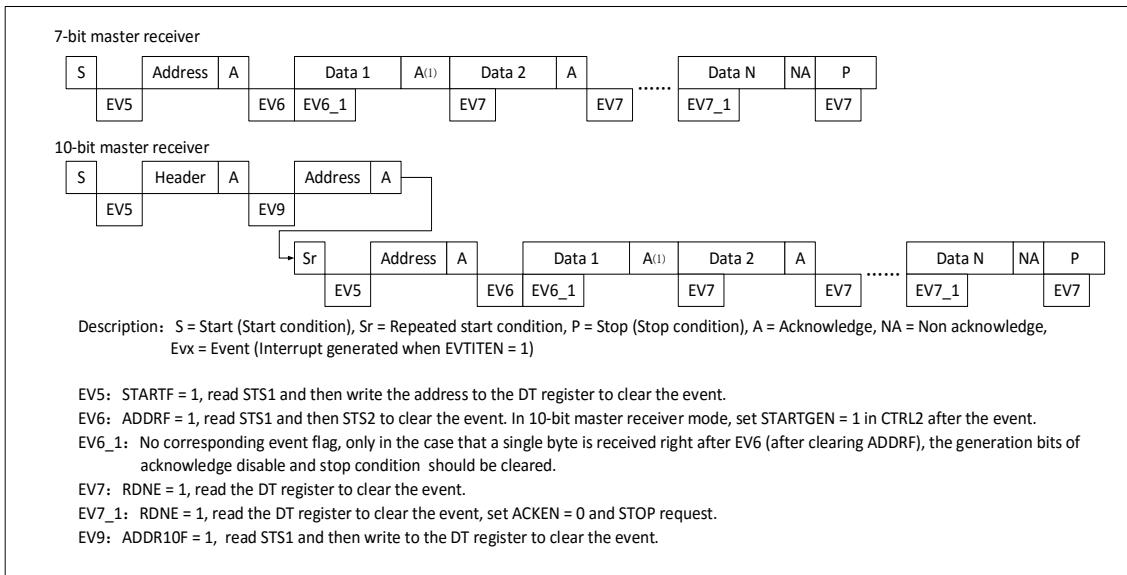
Case 1: I²C interrupt is set as the highest priority.

The master sends a NACK after the last byte is received from the slave. After receiving the NACK, the slave releases the control on the SCL and SDA lines. Then, the master can send a Stop/re-Start condition.

- To generate a NACK pulse after the last data byte is received, the ACKEN bit must be cleared after reading the second last data byte (after the second last RDNE event).
- To generate a Stop/re-Start condition, software must set the STOPGEN/STARTGEN bit after reading the second last data byte (after the second last RDNE event).
- When only a single byte is received, the generation bits of acknowledge and Stop condition should be disabled right after EV6 (in EV6_1, after ADDRFF is cleared).

After the Stop condition is generated, the I²C interface returns to slave mode automatically (The MSF bit is cleared).

Figure 15-6 Transfer Sequence Diagram of Master Receiver

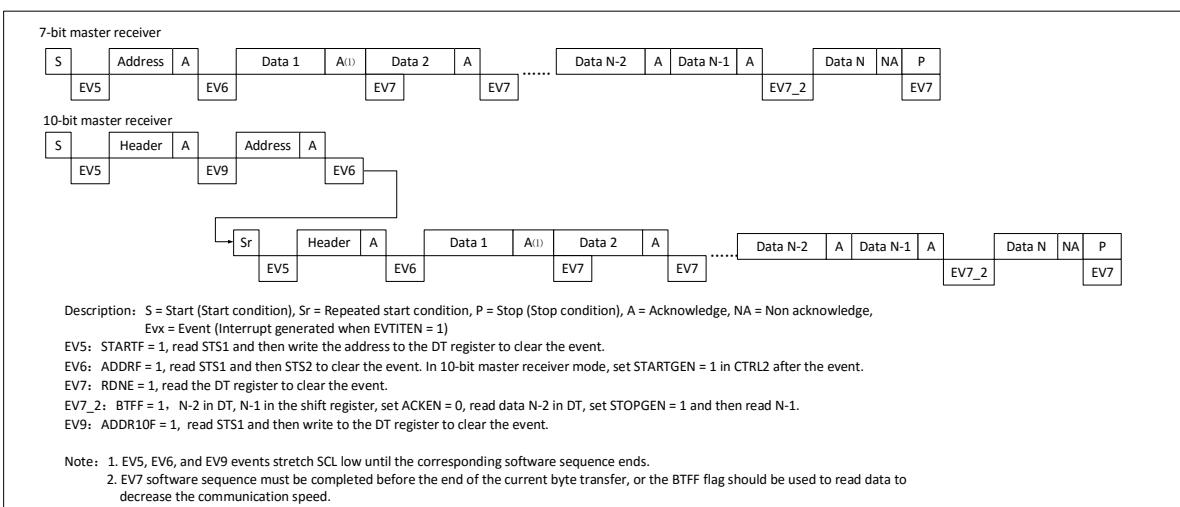
**Note:**

1. If a single byte is received, it is NA.
2. The EV5, EV6, and EV9 events stretch SCL low until the end of the corresponding software sequence.
3. The EV7 software sequence must be completed before the end of the current byte transfer.
4. The EV6_1 or EV7_1 software sequence must be completed before the ACK pulse of the current byte transfer.

Case 2: I²C interrupt is not set as the highest priority, and the total number of bytes received is more than 2, N > 2.

In this case, data N-2 received is not read; after data N-1 is received, the clock is stretched, and communication is stopped. At this time, clear the ACKEN bit and then read data N-2 one by one. After that, set STOPGEN/STARTGEN and read data N-1. Wait until RDNE is set, read data N. This is illustrated in Figure 15-7:

Figure 15-7 Transfer Sequence Diagram for Master Receiver when N > 2



When 3 bytes are left to be read, software procedure is as follows:

- RDNE = 1 => No action
- Data N-1 received
- BTFF = 1, N-2 in DT, N-1 in the shift register, clock stretched low
- Clear the ACKEN bit
- Read data N-2, and the bus starts receiving data N

- Data N received, and respond to NACK
- Set the STARTGEN/STOPGEN bit
- RDNE = 1
- Read data N

Case 3: I²C interrupt is not set as the highest priority, and the total number of bytes received is 2 or 1, N = 2 or N = 1

- Receiving a single byte
 - Clear the ACKEN bit at ADDRIF event
 - Clear the ADDRIF bit
 - Set the STARTGEN/STOPGEN bit
 - Read data when the RDNE bit is set
- Receiving two bytes
 - Set the POSEN and ACKEN bits
 - Wait for the ADDRIF bit to be set
 - Clear the ADDRIF bit
 - Clear the ACKEN bit
 - Wait for BTFF to be set
 - Set the STOPGEN bit
 - Read the DT register twice

Figure 15-8 Transfer Sequence Diagram for Master Receiver when N = 2

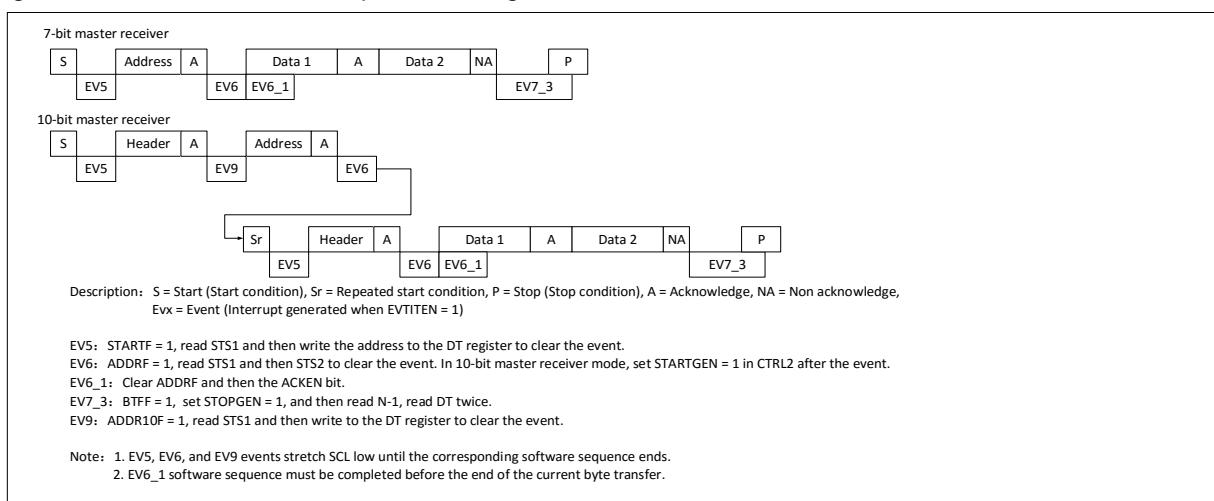
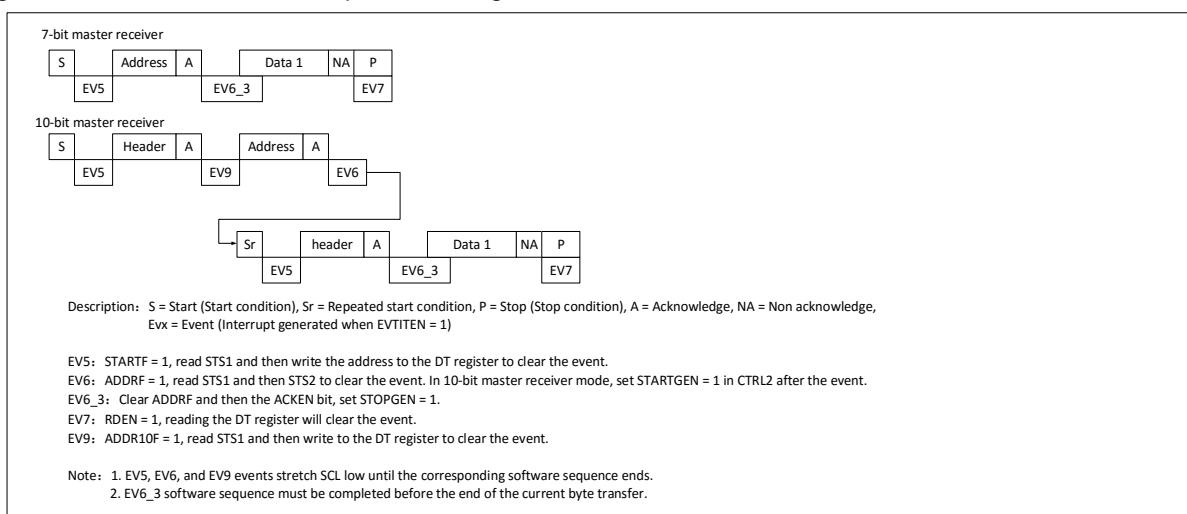


Figure 15-9 Transfer Sequence Diagram for Master Receiver when N = 1



15.3.4 Error Condition

The following error conditions may cause communication fail.

Bus error (BUSERR)

Bus error occurs when the I²C interface detects an external Stop or Start condition during an address or a data transfer. In this case:

- The BUSERR bit is set; an interrupt is generated if the ERRITEN bit is set.
- In slave mode, data are discarded, and the bus is released by hardware.
 - If it is a misplaced Start, the slave should clear the BUSERR bit and wait for the next Start.
 - If it is a misplaced Stop, the slave works normally for a Stop condition, and the bus is released by hardware at the same time.
- In master mode, the bus is not released by hardware, and the state of the current transmission is not affected. Whether to stop the current transmission is determined by the software.

Acknowledge failure (ACKFAIL)

Acknowledge failure occurs when the interface detects a non-acknowledge bit. In this case:

- The ACKFAIL bit is set; an interrupt is generated if the ERRITEN bit is set.
- When the transmitter receives a NACK, the communication must be reset.
 - If it is in slave mode, the bus is released by hardware.
 - If it is in master mode, a Stop or repeated Start condition must be generated by software.

Arbitration lost (ARLOST)

Arbitration lost occurs when the I²C interface detects an arbitration lost condition. In this case:

- The ARLOST bit is set by hardware; an interrupt is generated if the ERRITEN bit is set.
- The I²C interface automatically returns to slave mode (the MSF bit is cleared). When the I²C interface loses the arbitration, it cannot acknowledge its slave address in the same transfer. However, it can acknowledge after the master, which wins the bus, transmits a repeated Start.
- Bus is released by hardware.

Overrun/underrun error (OVRUN)

In slave mode, an overrun error occurs if clock stretching is disabled, the I²C interface receives a byte (RDNE = 1) when receiving data, but the last byte data is not yet read in the DT register. In this case:

- The last received byte is lost.
- In overrun error, software should clear the RDN bit, and the transmitter should re-transmit the last received byte.

In slave mode, an underrun error occurs if clock stretching is disabled, the I²C interface is transmitting data, but before the clock of the next byte arrives, new data is not yet written into the DT register (TDE = 1). In this case:

- The last byte in the DT register will be sent again.
- Users should ensure that when an underrun error occurs, repeated data received on the receiver side should be discarded. The transmitter side should update the DT register within the time specified by the I²C bus standard.

When the first byte is to be transmitted, the DT register must be written after ADDRF is cleared and before the first SCL rising edge. If this is not possible, the receiver must discard the first data.

15.3.5 SDA/SCL Line Control

- If clock stretching is enabled:
 - Transmitter mode: If TDE = 1 and BTFF = 1: The I²C interface holds the clock line low before transmission, to wait for the software to read STS1 and then write the data into the data register (both the buffer and the shift register are empty).
 - Receiver mode: If RDNE = 1 and BTFF = 1: The I²C interface holds the clock line low after data reception, to wait for the software to read STS1 and then read data

- in the data register, DT (both the buffer and the shift register are full).
- If clock stretching is disabled in slave mode:
 - If RDNE = 1, an overrun error occurs when DT is not read yet before the next byte is received. The last received byte is lost.
 - If TDE = 1, an underrun error occurs when no new data is written to DT before the next byte should be sent. The same byte will be sent again.
 - Write collision is not controlled.

15.3.6 SMBus

Introduction

The System Management Bus (SMBus) is a 2-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on I²C. SMBus provides a control bus for system and power management related tasks. A system may use SMBus to send messages to and from devices instead of individual control lines.

The System Management Bus (SMBus) Specification refers to three types of devices. A slave is a device that receives or responds to a command. A master is a device that issues commands, generates the clocks, and terminates the transfer. A host is a specialized master that provides the main interface to the CPU of the system. A host must have master-slave capability and must support the SMBus host notify protocol. Only one host is allowed in a system.

Similarities between SMBus and I²C

- 2-wire bus protocol (1 clock, 1 data) + optional SMBus alert line
- Master-slave communication, master provides clocks
- Multi-master capability
- SMBus data format similar to I²C 7-bit addressing format (Please refer to [Figure 15-1](#))

Differences between SMBus and I²C

Table 15-1 lists the differences between SMBus and I²C.

Table 15-1 Comparison between SMBus and I²C

SMBus	I ² C
Max. speed of 100 kHz	Max. speed of 400 kHz
Min. speed of 10 kHz	No minimum speed
35 ms clock low timeout	No clock timeout
Fixed logic levels	V _{DD} -dependent logic levels
Different address types (reserved, dynamic, etc.)	7-bit, 10-bit, and general call slave address types
Different bus protocols (quick command, process call, etc.)	No bus protocols

SMBus applications

With System Management Bus, devices can provide manufacturer information, tell the system its model/part number, save its state for a suspend event, report different types of errors, accept control parameters, and return its status. SMBus provides a control bus for system and power management related tasks.

Device identification

Any device on the System Management Bus as a slave has a unique address called the “slave address.” For the list of reserved slave addresses, please refer to the SMBus specification version 2.0 (<http://smbus.org/specs/>).

Bus protocols

The SMBus specification supports up to 9 bus protocols. For more details on these protocols and SMBus address types, please refer to SMBus specification version 2.0. (<http://smbus.org/specs/>). These protocols should be implemented by the user software.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. The Address Resolution Protocol (ARP) has the following features:

- Address assignment with the standard SMBus physical layer arbitration mechanism
- Assigned addresses remain constant while device power is applied; address retention after device power loss is also allowed.
- No additional SMBus packet overhead is incurred after address assignment. (i.e. subsequent accesses to assigned slave addresses have the same overhead as accesses to fixed address devices.)
- Any SMBus master can enumerate the bus.

Unique device identifier (UDID)

In order to assign addresses, a mechanism to isolate each device is needed. Each device must implement a unique device identifier.

For more details on the 128-bit UDID on ARP, please refer to SMBus specification version 2.0. (<http://smbus.org/specs/>). (UDID)

SMBus alert mode

SMBus Alert is an optional signal with an interrupt line, and it is used for devices that trade a pin for extending their control ability. SMBALERT, as well as SCL and SDA signals, is a wired-AND signal. SMBALERT is usually used with the SMBus General Call Address. Messages invoked with the SMBus are two bytes long.

A slave-only device can use SMBALERT to signal the host through the SMBALERT bit of I2C_CTRL1 register, showing its intention to communicate. The host then processes the interrupt and accesses all SMBALERT devices through the Alert Response Address, ARA (Alert Response Address, its address is 0001100x). Only the device(s) which pull SMBALERT low can acknowledge ARA. This status is identified by the SMBALERTF Status flag in the I2C_STS1 register. The host performs a modified Receive Byte operation. The 7-bit device address provided by the slave transmission device is placed in the 7 most significant bits of the byte. The eighth bit can be a '0' or '1'.

If more than one device pulls SMBALERT low, the device with the highest priority (lowest address) will win the communication via standard arbitration during the address transfer. After acknowledging the slave address, the device must disengage its SMBALERT pull-down. If the host still sees SMBALERT low when the message transfer is completed, it knows that ARA should be read again.

A host which does not implement the SMBALERT signal can periodically access ARA.

For more details on SMBus Alert mode, please refer to SMBus specification version 2.0. standard (<http://smbus.org/specs/>).

Timeout error

In the timing specifications, there are several differences between I²C and SMBus.

SMBus defines a clock low timeout, TIMEOUT of 35 ms. SMBus specifies TLOW: SEXT as the cumulative clock low extend time for a slave device. SMBus specifies TLOW: MEXT as the cumulative clock low extend time for a master device. For more details on these timeouts, please refer to SMBus specification version 2.0. standard (<http://smbus.org/specs/>).

The status flag Timeout or Tlow Error in I2C_STS1 shows the status of this feature.

How to use the interface in SMBus mode

To switch from I²C mode to SMBus mode, the following sequence should be performed:

- Set the SMBMODE bit in the I2C_CTRL1 register
- Configure the SMBTYPE and ARPEN bits in the I2C_CTRL1 register as the application required.

If the device is to be configured as a master, follow the Start condition generation procedure in [Section 15.3.3](#). Otherwise, follow [Section 15.3.2](#).

The software application should handle various SMBus protocols.

- SMB Device Default Address acknowledged if ARPEN = 1 and SMBTYPE = 0
- SMB Host Header acknowledged if ARPEN = 1 and SMBTYPE = 1
- SMB Alert Response Address acknowledged if SMBALERTF = 1

15.3.7 DMA Request

DMA requests (when enabled) are only for data transfer. DMA requests are generated when the data register becomes empty in transmission or becomes full in reception. The DMA requests must be acknowledged before the end of current byte transfer. When the number of data transfers, which has been programmed for the corresponding DMA channel, is reached, the DMA controller sends an End of Transfer signal, EOT, to the I²C interface, and generates a Transfer Complete interrupt if enabled:

- Master transmitter: In the EOT interrupt routine, DMA requests should be disabled, and then wait for a BTFF event before programming the Stop condition.
- Master receiver: When the number of bytes to be received is equal to or greater than two, the DMA controller sends a hardware signal, EOT_1, corresponding to DMA transfer (number of bytes – 1). If the DMALAST bit is set in the I2C_CTRL2 register, the hardware automatically sends a NACK after the next byte following EOT_1. Users can generate a Stop condition in the DMA Transfer Complete interrupt routine if enabled.

Transmission using DMA

DMA mode can be enabled by setting the DMAEN bit in the I2C_CTRL2 register. Once the TDE bit is set, data will be loaded from the programmed Memory to the I2C_DT register through DMA. To assign a DMA channel for I²C, perform the following sequence (where x is the channel number):

1. Set the I2C_DT register address in the DMA_CPBAX register. The data will be sent to this address from the memory after each TDE event.
2. Set the memory address in the DMA_CMBAX register. The data will be sent to I2C_DT from this memory after each TDE event.
3. Configure the total number of bytes to be transferred in the DMA_TCNTx register. After each TDE event, this value will be decremented.
4. Configure the channel priority by the CHPL[0:1] bits in the DMA_CHCTRLx register.
5. Set the DIR bit in the DMA_CHCTRLx register and configure interrupt requests to be sent after half transfer or full transfer according to the application requirements.
6. Activate the channel by setting the CHEN bit in the DMA_CHCTRLx register.

When the number of data transfers, programmed in the DMA controller, is reached, the DMA controller sends an End of Transfer signal, EOT/EOT_1, to the I²C interface. A DMA interrupt will be generated if enabled.

Note: Do not set the BUFITEN bit in the I2C_CTRL2 register if DMA is used for transmission.

Reception using DMA

DMA mode can be enabled for reception by setting the DMAEN bit in the I2C_CTRL2 register. Whenever a data byte is received, data will be sent from the I2C_DT register to the programmed Memory area through DMA (Please refer to DMA description). To set DMA channel for I²C reception, perform the following sequence (where x is the channel number):

1. Set the I2C_DT register address in the DMA_CPBAX register. The data will be sent from this address to the memory after each RDNE event.
2. Set the memory address in the DMA_CMBAX register. The data will be sent from the I2C_DT register to this memory after each RDNE event.
3. Configure the total number of bytes to be transferred in the DMA_TCNTx register. After each RDNE event, this value will be decremented.
4. Configure the channel priority by the CHPL[0:1] bits in the DMA_CHCTRLx register.
5. Set the DIR bit in the DMA_CHCTRLx register and configure interrupt requests to be sent after half transfer or full transfer according to the application requirements.
6. Activate the channel by setting the CHEN bit in the DMA_CHCTRLx register.

When the number of data transfers, programmed in the DMA controller, is reached, the DMA controller sends an End of Transfer signal, EOT/EOT_1, to the I²C interface. A DMA interrupt will be generated if enabled.

Note: Do not set the BUFITEN bit in the I2C_CTRL2 register if DMA is used for reception.

15.3.8 Packet Error Checking (PEC)

A packet error checking (PEC) calculator can improve the reliability of communication. The calculator calculates by using the following polynomial on each bit serially.

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

- PEC calculation is enabled by setting the PECEN bit in the I2C_CTRL1 register. PEC calculates with CRC-8 computation on all message bytes, including addresses and R/W bits.
 - In transmission: Set the PECTRA transfer bit in the I2C_CTRL1 register after the last TDE event. PEC will be transferred after the last transmitted byte.
 - In reception: Set the PECTRA bit in the I2C_CTRL1 register after the last RDNE event. If the next received byte is not equal to the internally calculated PECVAL, the receiver sends a NACK. No matter what the checking results is, a NACK is always sent after PEC. The PECTRA bit must be set before the ACK pulse of the current byte is received.
- A PECERR error flag/interrupt is available in the I2C_STS1 register.
- If DMA and PEC calculator are both enabled:
 - In transmission: When the I²C interface receives an EOT signal from the DMA controller, it automatically sends PECVAL after the last byte.
 - In reception: When the I²C interface receives an EOT_1 signal from the DMA controller, it will automatically consider the next byte as PECVAL and check it. A DMA request is generated after PECVAL is received.
- To allow intermediate PEC transfers, a control bit (the DMALAST bit) is available in the I2C_CTRL2 register to determine if it is really the last DMA transfer. If it is the last DMA request for a master receiver, a NACK is automatically sent after the last received byte.
- PEC calculation is corrupted when arbitration is lost.

15.3.9 I²C Interrupt Request

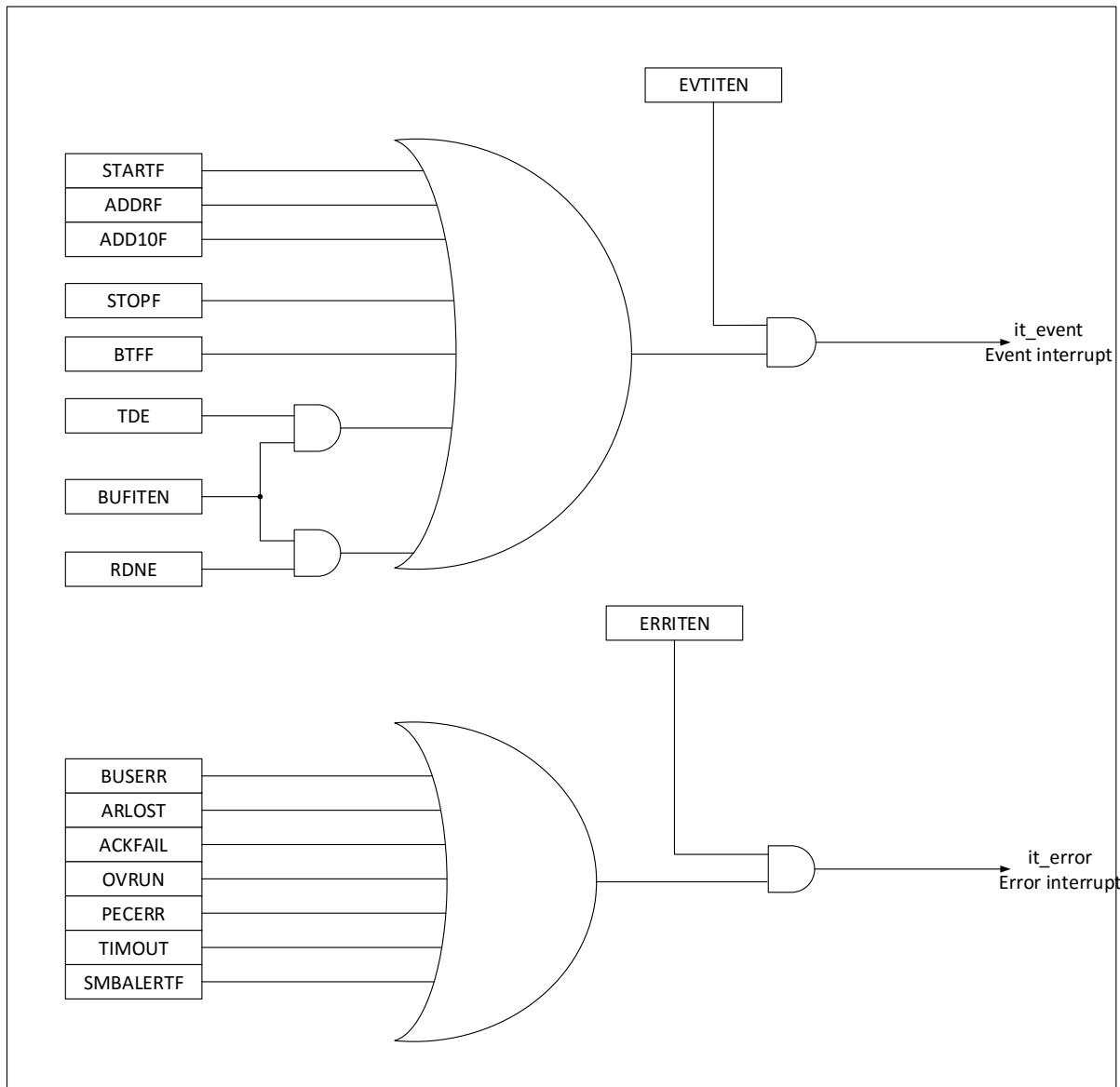
Table 15-2 lists all the I²C interrupt request.

Table 15-2 I²C Interrupt Request

Interrupt Event	Event Flag	Enable Control Bit
Start bit sent (Master)	STARTF	EVTITEN
Address sent (Master) or Address matched (Slave)	ADDRF	
10-bit header sent (Master)	ADDR10F	
Stop received (Slave)	STOPF	
Data byte transfer finished	BTFF	EVTITEN and BUFITEN
Receive buffer not empty	RDNE	
Transmit buffer empty	TDE	
Bus error	BUSERR	ERRITEN
Arbitration loss (Master)	ARLOST	
Acknowledge failure	ACKFAIL	
Overrun/Underrun	OVRUN	
PEC error	PECERR	
Timeout/Tlow error	TIMOUT	
SMBus Alert	SMBALERTF	

- Note: 1. STARTF, ADDR1F, ADDR10F, STOPF, BTFF, RDNE, and TDE are logically OR-ed on the same interrupt channel.
2. BUSERR, ARLOST, ACKFAIL, OVRUN, PECERR, TIMOUT, and SMBALERTF are logically OR-ed on the same interrupt channel.

Figure 15-7 I²C Interrupt Mapping Diagram



15.3.10 I²C Debug Mode

When the microcontroller enters debug mode (Cortex®-M4F core halted), the SMBUS timeout either continues to work normally or stops, depending on the DBG_I2Cx_SMBUS_TIMEOUT configuration bits in the DBG module. For more details, please refer to [Section 22.3.1](#).

15.4 I²C Registers

These peripheral registers can be accessed by half-word (16-bit) or word (32-bit).

Table 15-3 I²C Register Map and Reset Value

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
000h	I ² C_CTRL1	Reserved																SWRESET	Reserved	SMBALERT	PECTRA	POSEN	ACKEN	STOPGEN	STARTGEN	NOCLKSTRECH	GCEN	PECEN	ARPEN	SMB_TYPE	Reserved	SMBMODE	PEN		
	Reset Value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
004h	I ² C_CTRL2	Reserved																Reserve d	DMALAST	DMAEN	BUFTEN	EVTITEN	ERRITEN	0	0	0	0	0	0	0	0	0	0		
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
008h	I ² C_OADDR1	Reserved																O	ADDRMOD	Reserved		ADDR[9:8]		ADDR[7:1]											
	Reset Value																																		
00Ch	I ² C_OADDR2	Reserved																	Reserved		Reserved		ADDR2[7:1]												
	Reset Value																																		
010h	I ² C_DT	Reserved																	DT[7:0]																
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
014h	I ² C_STS1	Reserved																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Reset Value																		TDE	RDNE	Reserved	PECERR	OVRUN	ACKFAIL	ARLOST	BUSERR	DUALF	SMB_HOST	GCADDRF	STOPF	ADDR10F	BTFF	TRF	Reserved	0
018h	I ² C_STS2	Reserved																	PECVAL[7:0]								DUALF	SMBDEFTADDRF	GCADDRF	STOPF	ADDR10F	BTFF	TRF	Reserved	0
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
01Ch	I ² C_CLKCTRL	Reserved																	CLKCTRL[11:0]																
	Reset Value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
020h	I ² C_TMRISE	Reserved																	TMRISE[5:0]								0	0	0	0	0	1	0	0	
	Reset Value																																		

15.4.1 Control Register 1 (I²C_CTRL1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW RESE T	Reserve d	SMB ALER T	PE C TR A	PO S EN	AC K EN	STO P GEN	STAR T GEN	NOC L KST R ETC H	GCE N	PE C EN	AR P EN	SMB TYP E	Reserve d	SMB MOD E	PE N

Bit 15	SWRESET: Software reset When set, I ² C is under reset state. Before resetting this bit, ensure that I ² C pins are released, and the bus is free. 0: I ² C peripheral is not at reset state. 1: I ² C peripheral is at reset state. Note: This bit can be used when the BUSYF bit is '1', and no Stop condition is detected on the bus.
Bit 14	Reserved. Forced to be '0' by hardware.
Bit 13	SMBALERT: SMBus alert This bit is set or cleared by software. It is cleared by hardware when PEN = 0. 0: Release SMBAlert pin high. Alert Response Address Header follows the NACK signal. 1: Drive SMBAlert pin low. Alert Response Address Header follows the ACK signal.
Bit 12	PECTRA: Packet error checking This bit is set or cleared by software. It is cleared by hardware after PECVAL is sent, or under Stop/Start condition. 0: No PEC transfer 1: PEC transfer (in transmitter or receiver mode) Note: PEC calculation is corrupted when arbitration is lost.
Bit 11	POSEN: Acknowledge/PEC Position (for data reception) This bit is set or cleared by software. 0: The ACKEN bit controls (N)ACK of the current byte being received in the shift register. The PECTRA bit indicates that the current byte in shift register is PECVAL. 1: The ACKEN bit controls (N)ACK of the next byte to be received in the shift register. The PECTRA bit indicates that the next byte received in the shift register is PECVAL. Note: The POSEN bit is used only for reception of 2 bytes. It must be configured before data reception starts. To NACK the 2nd byte, the ACKEN bit must be cleared after ADDR is cleared. To check the PECVAL bit of the 2nd byte, the PECTRA bit should be set after the POSEN bit is set, during ADDR and event stretch.
Bit 10	ACKEN: Acknowledge enable This bit is set or cleared by software. 0: No acknowledge returns. 1: Acknowledge returns after a byte is received (matched address or data).
Bit 9	STOPGEN: Stop generation This bit is set or cleared by software. It is cleared by hardware when a Stop condition is detected. It is set by hardware when a timeout error is detected. In master mode: 0: No Stop condition is generated. 1: Stop condition is generated after the data register and the shift register complete transfer or after the current Start condition is sent. In slave mode: 0: No Stop condition is generated. 1: Release the SCL and SDA lines after the current byte transfer. Note: If the STOPGEN, STARTGEN, or PECTRA bit is set, do not perform any write access to I2C_CTRL1 before this bit is cleared by hardware. Otherwise, the STOPGEN, STARTGEN, or PECTRA bit may be set for the second time.

Bit 8	STARTGEN: Start generation This bit is set or cleared by software. It is cleared by hardware when a Start condition is sent. In master mode: 0: No Start condition is generated. 1: Start condition is generated repeatedly. In slave mode: 0: No Start condition is generated. 1: Start condition is generated when the bus is free.
Bit 7	NOCLKSTRETCH: Clock stretching disable (Slave mode) This bit is used to disable clock stretching in slave mode when ADDRF or BTFF flag is set, until it is reset by software. 0: Clock stretching is enabled. 1: Clock stretching is disabled.
Bit 6	GCEN: General call enable 0: General call disabled. Address 00h is NACKed. 1: General call enabled. Address 00h is ACKed.
Bit 5	PECEN: PEC enable 0: PEC calculation is disabled. 1: PEC calculation is enabled
Bit 4	ARPEN: ARP enable 0: ARP is disabled. 1: ARP is enabled. If SMBTYPE = 0, SMBus device default address is recognized. If SMBTYPE = 1, SMBus host address is recognized.
Bit 3	SMBTYPE: SMBus type 0: SMBus device 1: SMBus host
Bit 2	Reserved. Forced to be '0' by hardware.
Bit 1	SMBMODE: SMBus mode 0: I ² C mode 1: SMBus mode
Bit 0	PEN: Peripheral enable 0: I ² C peripheral is disabled. 1: I ² C peripheral is enabled. According to the configuration of the SMBus bit, the corresponding I/O port should be configured as Alternate Function. Note: 1. If this bit is reset while a communication is ongoing, the I ² C peripheral is disabled at the end of the current communication, and return to IDLE state. 2. All bits are reset as PE = 0 at the end of the communication. 3. In master mode, this bit must not be reset before the end of the communication.

Note: When the STARTGEN, STOPGEN, or PECTRA bit is set, the software must write to I²C_CTRL1 after the corresponding bit is cleared by hardware. Otherwise, there is a risk of generating a second STARTGEN, STOPGEN, or PECTRA request.

15.4.2 Control Register 2 (I²C_CTRL2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DMA LAST	DM AEN	BUF ITEN	EVT ITEN	ERR ITEN	CLKFREQ[7:0]									
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:13		Reserved. Forced to be '0' by hardware.													

Bit 12	DMALAST : DMA last transfer 0: The next DMA EOT is not the last transfer. 1: The next DMA EOT is the last transfer. Note: This bit is used in master receiver mode to permit the generation of a NACK at the last data reception.
Bit 11	DMAEN : DMA requests enable 0: DMA request is disabled. 1: DMA request is enabled when TDE = 1 or RDNE = 1.
Bit 10	BUFITEN : Buffer interrupt enable 0: No interrupt is generated when TDE = 1 or RDNE = 1. 1: An interrupt is generated when TDE = 1 or RDNE = 1. (Regardless of the state of DMAEN).
Bit 9	EVTITEN : Event interrupt enable 0: Event interrupt is disable. 1: Event interrupt is enable. Interrupts are generated in the following conditions: – STARTF = 1 (Master mode) – ADDR10F = 1 (Master/slave mode) – STOPF = 1 (Slave mode) – BTFF = 1, but no TDE or RDNE event – If BUFITEN = 1, TDE event is 1. – If BUFITEN = 1, RDNE event is 1.
Bit 8	ERRITEN : Error interrupt enable 0: Error interrupt is disable. 1: Error interrupt is enable. Interrupts are generated in the following conditions: – BUSERR = 1 – ARLOST = 1 – ACKFAIL = 1 – OVRUN = 1 – PECERR = 1 – TIMOUT = 1 – SMBALERTF = 1
Bit 7:0	CLKFREQ[7:0] : Peripheral clock frequency Correct input clock frequency must be set to generate correct timings. The range allowed is between 2 and 100 MHz: 00000000: Not allowed 00000001: Not allowed 00000010: 2 MHz ... 01100100: 100 MHz

15.4.3 Own Address Register 1 (I²C_OADDR1)

Address offset: 0x08

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRMODE	Reserved	Reserved		ADDR[9:8]		ADDR[7:1]		ADDR0							
rw	r	res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15	ADDRMODE : Addressing mode (Slave mode) 0: 7-bit slave address (10-bit address not acknowledged) 1: 10-bit slave address (7-bit address not acknowledged)
Bit 14:10	Reserved. Forced to be '0' by hardware.

Bit 9:8	ADDR[9:8] : Interface address Not attended in 7-bit addressing mode. The 9 ~ 8 bits of the address in 10-bit addressing mode.
Bit 7:1	ADDR[7:1] : Interface address The 7 ~ 1 bits of the address.
Bit 0	ADDR0 : Interface address Not attended in 7-bit addressing mode. The 0 bit of the address in 10-bit addressing mode.

15.4.4 Own Address Register 2 (I²C_OADDR2)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						ADDR2[7:1]						DUALEN			
res						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:8	Reserved. Forced to be '0' by hardware.
Bit 7:1	ADDR2[7:1] : Interface address The 7 ~ 1 bits in the dual address mode.
Bit 0	DUALEN : Dual addressing mode enable 0: Only OADDR1 is recognized in 7-bit addressing mode. 1: Both OADDR1 and OADDR2 are recognized in 7-bit addressing mode.

15.4.5 Data Register (I²C_DT)

Address offset: 0x10

Reset value: 0x0000

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

Bit 15:8	Reserved. Forced to be '0' by hardware.
Bit 7:0	DT[7:0] : 8-bit data register Used to store data received or to be transmitted to the bus. Transmitter mode: Data transmission starts automatically when a byte is written in the DT register. Once the transmission starts (TDE = 1), I ² C will maintain a continuous transmit stream if the next data to be transmitted is written to the DT register in time. Receiver mode: Received byte is copied into the DT register (RDNE = 1). A continuous transmit stream can be maintained if the data register is read before the next data byte is received (RDNE = 1). Note 1: In slave mode, the address is not copied into the data register, DT. Note 2: Write collision is not managed by hardware (DT can still be written if TDE = 0). Note 3: If an ARLOST event occurs on ACK pulse, the received byte is not copied into the data register, so it cannot be read.

15.4.6 Status Register 1 (I²C_STS1)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMB ALER TF	TIM OUT	Res erve d	PEC ERR	OVR UN	ACK FAIL	AR LOS T	BUS ERR	TDE	RDN E	Rese rved	STO PF	ADD R10F	BTF F	ADD RF	STA RTF
rcw0	rcw0	res	rcw0	rcw0	w0	w0	w0	r	r	res	r	r	r	r	r

Bit 15	SMBALERTF: SMBus alert In SMBus host mode: 0: No SMBus alert 1: SMBALERTF event occurs on the pin. In SMBus slave mode: 0: No SMBAlert response address header. 1: SMBAlert response address header to SMBAlert LOW received. – This bit is cleared by software, or by hardware when PEN = 0.
Bit 14	TIMOUT: Timeout or Tlow error 0: No timeout error 1: SCL remains LOW for 25 ms (Timeout), or Master cumulative clock low extend time is more than 10 ms (Tlow: mext), or Slave cumulative clock low extend time is more than 25 ms (Tlow: sext). – When set in slave mode: Slave resets the communication, and bus is released by hardware – When set in master mode: Stop condition is sent by hardware. – Cleared by software, or by hardware when PEN = 0 Note: This function is available only in SMBus mode.
Bit 13	Reserved. Forced to be '0' by hardware.
Bit 12	PECERR: PEC Error in reception 0: No PEC error: Receiver returns ACK after PEC reception (if ACKEN = 1). 1: PEC error occurs: Receiver returns NACK after PEC reception (whatever the ACKEN value is). – This bit is cleared by software.
Bit 11	OVRUN: Overrun/Underrun 0: No overrun/underrun 1: Overrun or underrun occurs. – Set by hardware in slave mode when NOCLKSTRETCH = 1, and: – In reception mode, if a new byte is received (including ACK pulse), and the data register is not read yet, then the new received byte will be lost. – In transmission mode, if a new byte is to be sent, and the data register is not written yet, then the same byte will be sent twice. – Cleared by software, or by hardware when PEN = 0. Note: If the data register write access occurs very close to SCL rising edge, the sent data is unspecified, and a hold timing error occurs.
Bit 10	ACKFAIL: Acknowledge failure 0: No acknowledge failure 1: Acknowledge failure occurs. – Set by hardware when no acknowledge is returned. – Cleared by software, or by hardware when PEN = 0.
Bit 9	ARLOST: Arbitration lost (master mode) 0: No arbitration lost is detected. 1: Arbitration lost is detected. Set by hardware when the interface loses the arbitration of the bus to another master – Cleared by software, or by hardware when PEN = 0. After an ARLOST event, the I ² C interface switches back to slave mode automatically (M/SL = 0). Note: In SMBUS mode, the arbitration on the data in slave mode occurs only at the data phase, or the acknowledge transmission (not on the address acknowledge).

Bit 8	<p>BUSERR: Bus error</p> <p>0: No misplaced Start or Stop condition</p> <p>1: Misplaced Start or Stop condition occurs.</p> <ul style="list-style-type: none"> – Set by hardware when the interface detects a misplaced Start or Stop condition. – Cleared by software, or by hardware when PEN = 0. <p>When BUSERR error occurs, this flag should be cleared by software in time to ensure that normal communication is resumed.</p>
Bit 7	<p>TDE: Data register empty (Transmitters)</p> <p>0: Data register is not empty.</p> <p>1: Data register is empty.</p> <ul style="list-style-type: none"> – Set when the data register is empty in transmission. This bit is not set at address transmission phase. – Cleared by software writing to the DT register, or automatically by hardware after a Start or a Stop condition occurs. <p>This bit is not set either if a NACK is received, or if the next byte to be transmitted is PECVAL (PECTRA = 1).</p> <p>Note: The TDE bit is not cleared by writing the first data to be transmitted, or by writing data when BTFF is set, since in both cases the data register is still empty.</p>
Bit 6	<p>RDNE: Data register not empty (Receivers)</p> <p>0: Data register is empty.</p> <p>1: Data register is not empty.</p> <ul style="list-style-type: none"> – Set when the data register is not empty in receiver mode. This bit is not set at address reception phase. – Cleared by software reading or writing the data register or by hardware when PEN = 0 <p>RDNE is not set at ARLOST event.</p> <p>Note: RDNE is not cleared by reading data when BTFF is set, since the data register is still full.</p>
Bit 5	Reserved. Forced to be '0' by hardware.
Bit 4	<p>STOPF: Stop detection (Slave mode)</p> <p>0: No Stop condition is detected.</p> <p>1: Stop condition is detected.</p> <ul style="list-style-type: none"> – Set by hardware when a Stop condition is detected on the bus by the slave after an acknowledge (if ACKEN = 1). – Cleared by a write to the CTRL1 register after the software reads the STS1 register, or by hardware when PEN = 0. <p>Note: The STOPF bit is not set after a NACK reception.</p>
Bit 3	<p>ADDR10F: 10-bit header sent (Master mode)</p> <p>0: No ADDR10F event occurs.</p> <p>1: Master sends the first address byte.</p> <ul style="list-style-type: none"> – Set by hardware when the master sends the first byte in 10-bit addressing mode. – Cleared by a write to the CTRL1 register after the software reads the STS1 register, or by hardware when PEN = 0. <p>Note: The ADDR10F bit is not set after a NACK reception.</p>
Bit 2	<p>BTFF: Byte transfer finished</p> <p>0: Data byte transfer is not completed yet.</p> <p>1: Data byte transfer is completed.</p> <p>When NOCLKSTRETCH = 0, set by hardware in the following conditions:</p> <ul style="list-style-type: none"> – In reception mode, when a new byte is received (including ACK pulse), and the data register is not read yet (RDNE = 1). – In transmission mode, when a new byte is to be sent, and the data register is not written yet (TDE = 1). – Cleared by read or write access to the data register after software reads the STS1 register, or by hardware after a Start or Stop condition is sent in transmission. <p>Note: The BTFF bit is not set after a NACK reception.</p> <p>The BTFF bit is not set if next byte to be transmitted is PECVAL (TRF = 1 in the I2C_STS2 register, and at the same time PECTRA = 1 in the I2C_CTRL1 register).</p>

Bit 1	<p>ADDRF: Address sent (Master mode)/matched (Slave mode) This bit is cleared by read access to the STS2 register after software reads the STS1 register.</p> <p>Address matched (Slave mode)</p> <p>0: Address is mismatched or not received. 1: Received address is matched.</p> <ul style="list-style-type: none"> Set by hardware when the received slave address matches with the content of the OADDR register, or a general call, or a SMBus device default address, or SMBus host recognizes SMBus alert (If the corresponding configuration is enabled). <p>Address sent (Master mode)</p> <p>0: Address transmission does not end. 1: Address transmission ends.</p> <ul style="list-style-type: none"> Set after ACK of the 2nd byte is received in 10-bit addressing mode. Set after ACK of the byte is received in 7-bit addressing mode. <p>Note: ADDR is not set after a NACK reception.</p>
Bit 0	<p>STARTF: Start bit (Master mode)</p> <p>0: No Start condition 1: Start condition is generated.</p> <ul style="list-style-type: none"> Set when a Start condition is generated. Cleared by write access to the data register after software reads the STS1 register.

15.4.7 Status Register 2 (I²C_STS2)

Address offset: 0x18

Reset value: 0x0000

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

Bit 15:8	<p>PECVAL[7:0]: Packet error checking register When PECEN = 1, PECVAL[7:0] stores internal PEC values. It is cleared when PECEN is reset.</p>
Bit 7	<p>DUALF: Dual flag (Slave mode)</p> <p>0: Received address matches with the content of OADDR1. 1: Received address matches with the content of OADDR2.</p> <ul style="list-style-type: none"> This bit is cleared by hardware when a Stop condition or a repeated Start condition is generated, or PEN = 0.
Bit 6	<p>SMBHOSTADDRF: SMBus host header (Slave mode)</p> <p>0: SMBus host address is not received. 1: SMBus host address is received when SMBTYPE = 1 and ARPEN = 1.</p> <ul style="list-style-type: none"> This bit is cleared by hardware when a Stop condition or a repeated Start condition is generated, or PEN = 0.
Bit 5	<p>SMBDEFTADDRF: SMBus device default address (Slave mode)</p> <p>0: SMBus device default address is not received. 1: SMBus device default address is received when ARPEN = 1.</p> <ul style="list-style-type: none"> This bit is cleared by hardware when a Stop condition or a repeated Start condition is generated, or PEN = 0.
Bit 4	<p>GCADDRF: General call address (Slave mode)</p> <p>0: General call address is not received. 1: General call address is received when GCEN = 1.</p> <ul style="list-style-type: none"> This bit is cleared by hardware when a Stop condition or a repeated Start condition is generated, or PEN = 0.
Bit 3	Reserved. Forced to be '0' by hardware.
Bit 2	<p>TRF: Transmitter/receiver</p> <p>0: Data is received. 1: Data is transmitted. At the end of the whole address transmission phase, this bit is set depending on the R/W bit of the address byte.</p> <p>It is cleared by hardware after detection of Stop condition (STOPF = 1), repeated Start condition, bus arbitration lost (ARLOST = 1), or when PEN = 0.</p>

Bit 1	BUSYF: Bus busy 0: No data communication on the bus 1: Data communication is ongoing on the bus. – Set by hardware on detection of SDA or SCL low – Cleared by hardware on detection of a Stop condition This bit indicates the ongoing communication on the bus. This information is still updated when the interface is disabled (PEN = 0).
Bit 0	MSF: Master/Slave 0: Slave mode 1: Master mode – Set by hardware when the interface is in master mode (STARTF = 1) – Cleared by hardware when a Stop condition is detected on the bus, during arbitration lost (ARLOST = 1), or when PEN = 0.

15.4.8 Clock Control Register (I²C_CLKCTRL)

Address offset: 0x1C

Reset value: 0x0000

- Note:
1. *FCLK1 must be a multiple of 10 MHz so that the 400 kHz fast mode clock can be correctly generated.*
 2. *The CLKCTRL register can be configured only when I²C is disabled (PEN = 0).*

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
F/S MODE	FM DUTY	Reserved	CLKCTRL[11:0]													
rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bit 15	F/SMODE: I ² C master mode selection 0: Standard mode I ² C 1: Fast mode I ² C
Bit 14	FMDUTY: Fast mode duty cycle 0: In fast mode: $T_{low}/T_{high} = 2$ 1: In fast mode: $T_{low}/T_{high} = 16/9$ (Please refer to CLKCTRL).
Bit 13:12	Reserved. Forced to be '0' by hardware.
Bit 11:0	CLKCTRL[11:0] : Clock control register in Fast/Standard mode (Master mode) The divider ratio sets the SCL clock in master mode. In slave mode, it does not need to be configured, or is the same as the one in master mode according to the equation. <u>In I²C standard mode or SMBus mode:</u> $T_{high} = CLKCTRL \times T_{PCLK1}$ $T_{low} = CLKCTRL \times T_{PCLK1}$ <u>In I²C fast mode:</u> If FMDUTY = 0: $T_{high} = CLKCTRL \times T_{PCLK1}$ $T_{low} = 2 \times CLKCTRL \times T_{PCLK1}$ If FMDUTY = 1: (Up to 400 kHz) $T_{high} = 9 \times CLKCTRL \times T_{PCLK1}$ $T_{low} = 16 \times CLKCTRL \times T_{PCLK1}$ For example: In standard mode, to generate SCL frequency of 100 kHz: If CLKFREQ = 08, $T_{PCLK1} = 125$ ns, then CLKCTRL should be written 0x28 (40×125 ns = 5000 ns). Note: 1. The minimum value allowed is 0x04. In FAST DUTY mode, the minimum value allowed is 0x01. 2. $T_{high} = t_{f(SCL)} + t_{w(SCLH)}$. Please refer to the data sheet for detailed parameter definitions. 3. $T_{low} = t_{f(SCL)} + t_{w(SCLL)}$. Please refer to the data sheet for detailed parameter definitions. 4. These delays are without filter. 5. The CLKCTRL register can be configured only when I ² C is disabled (PEN = 0). 6. fck must be a multiple of 10 MHz so that the 400 kHz fast mode clock can be correctly generated.

15.4.9 TMRISE Register (I²C_TMRISE)

Address offset: 0x20

Reset value: 0x0002

																0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1									
Reserved								TMRISE[5:0]															
res								rw															
Bit 15:6								Reserved. Forced to be '0' by hardware.															
Bit 5:0								TMRISE[5:0]: Maximum rise time in Fast/Standard mode (Master mode) These bits must be programmed as the maximum SCL rise time given in the I ² C bus specification, incremented by 1. For instance: In standard mode, the maximum SCL rise time allowed is 1000 ns. If, in the I2C_CTRL2 register, the value of CLKFREQ[7:0] bits is equal to 0x08, and T _{PCLK1} = 125 ns, then the TRISE[5:0] bits must be programmed with 09h (1000 ns/125 ns = 8+1). The filter value can also be added to TMRISE[5:0]. If the result is not an integer, TMRISE[5:0] must be programmed with the integer part to ensure the t _{HIGH} parameter. Note: TMRISE[5:0] must be configured only when I ² C is disabled (PEN = 0).															

16 Universal Synchronous/Asynchronous Receiver/Transmitter (USART)

16.1 USART Introduction

The universal synchronous asynchronous receiver transmitter (USART) offers a flexible means to realize full-duplex data exchange with external devices which require industry standard NRZ asynchronous serial data format. With a fractional baud rate generator, the USART offers a very wide range of baud rates.

It supports synchronous one-way communication and half-duplex single-wire communication. It also supports the LIN (local interconnection network), Smartcard Protocol, IrDA (infrared data association) SIR ENDEC specifications, and CTS/RTS (Clear To Send/Request To Send) hardware flow operation. It also allows multi-processor communication.

High speed data communication is possible by using the DMA with multi-buffer configuration.

16.2 USART Main Features

- Full duplex, asynchronous communication
- Half duplex, single wire communication
- NRZ standard format (Mark/Space)
- Programmable baud rate generator
 - A common programmable transmit and receive baud rate up to 6.25 Mbit/s
- Programmable data word length (8 bits or 9 bits)
- Configurable stop bits - Support 1 or 2 stop bits
- LIN master with break generation capability and LIN slave with break detection capability
 - When USART is hardware configured as LIN, 13-bit break generation and 10-bit/11-bit break detection
- Transmitter clock output for synchronous transmission
- IrDA SIR Encoder Decoder
 - Support 3-bit/16-bit duration in normal mode
- Asynchronous Smartcard protocol defined in ISO7816-3 Standard
 - Support 0.5 or 1.5 stop bits in Smartcard mode
- Configurable DMA multi-buffer communication
 - Buffering of received/transmitted data with DMA
- Separate enable bits for transmitter and receiver
- Detection flag
 - Receive buffer full
 - Transmit buffer empty
 - End of transition flag
- Parity control
 - Transmit parity bit
 - Parity check of received data
- Four error detection flags
 - Overrun error
 - Noise error

- Framing error
- Parity error
- 10 interrupt sources with flags
 - CTSF changes
 - LIN break detection
 - Transmit data register empty
 - Transmission completed
 - Receive data register full
 - Idle line detected
 - Overrun error
 - Framing error
 - Noise error
 - Parity error
- Multiprocessor communication – If address does not match, enter mute mode.
- Wake up from mute mode (by idle line detection or address detection)
- Two ways to wake up receiver: Address bit (MSB, the 9th bit), idle line

16.3 USART Function Overview

The interface is connected to other devices through three pins (See [Figure 16-1](#)). Any USART bidirectional communication requires at least two pins: receive data input (Rx) and transmit data output (Tx):

Rx: Serial data input uses oversampling techniques for data recovery by discriminating between data and noise.

Tx: Serial data output. When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and does not transmit data, the Tx pin is at high level. In single-wire and Smartcard modes, this I/O port is used to transmit and receive the data at the same time.

- An idle line before transmission or reception
- A start bit
- A data word (8 bits or 9 bits), the least significant bit first
- 0.5, 1, 1.5, 2 stop bits, indicating the end of frame
- Use a fractional baud rate generator —— With 12-bit integer and 4-bit decimal
- A status register (USART_STS)
- Data register (USART_DT)
- A baud rate register (USART_BAUDR) , 12-bit integer and 4-bit decimal
- A guard time register (GVAL) in Smartcard mode

Please refer to [Section 16.6](#) for the detailed descriptions of each bit in the registers mentioned above.

In synchronous mode, the following pins are required:

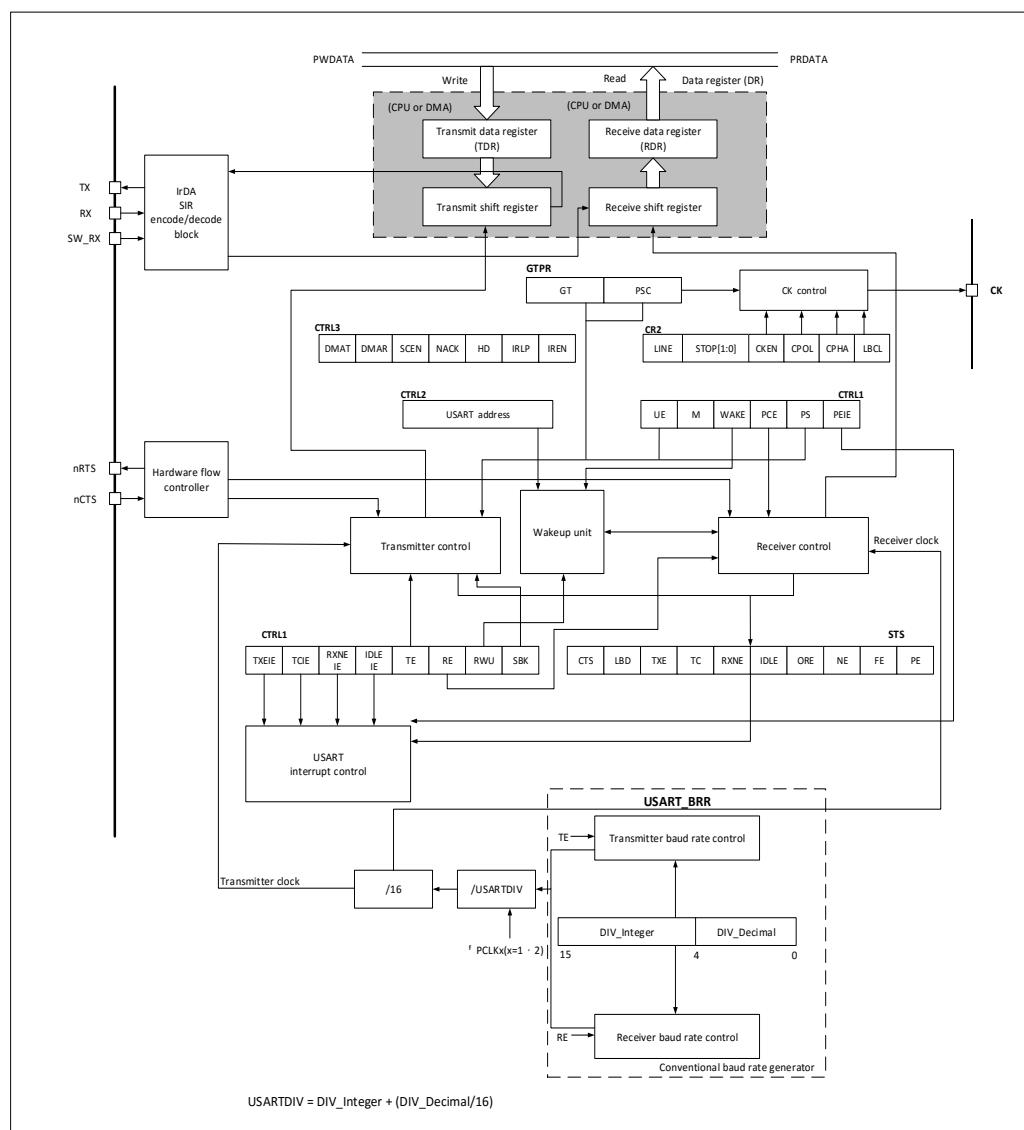
- CK: Transmitter clock output. This pin output is used for synchronous transmission clock (There is no clock pulses on the start bit and stop bit, and software can optionally send a clock pulse on the last data bit). Data can be received synchronously on Rx. This can be used to control peripherals that have shift registers (e.g. LCD drivers). The clock phase and polarity are software programmable. In Smartcard mode, CK can provide the clock for the Smartcard.

The following pins are required in hardware flow control mode:

- CTS: Cleared after reception. If it is low level, the next data transmission will occur after the current transmission ends. If it is high level, the next data transmission will be stopped by the end of current data transmission.

- RTS: Send requests. If it is low level, the USART is ready to receive a data.

Figure 16-1 USART Block Diagram



16.3.1 USART Feature Description

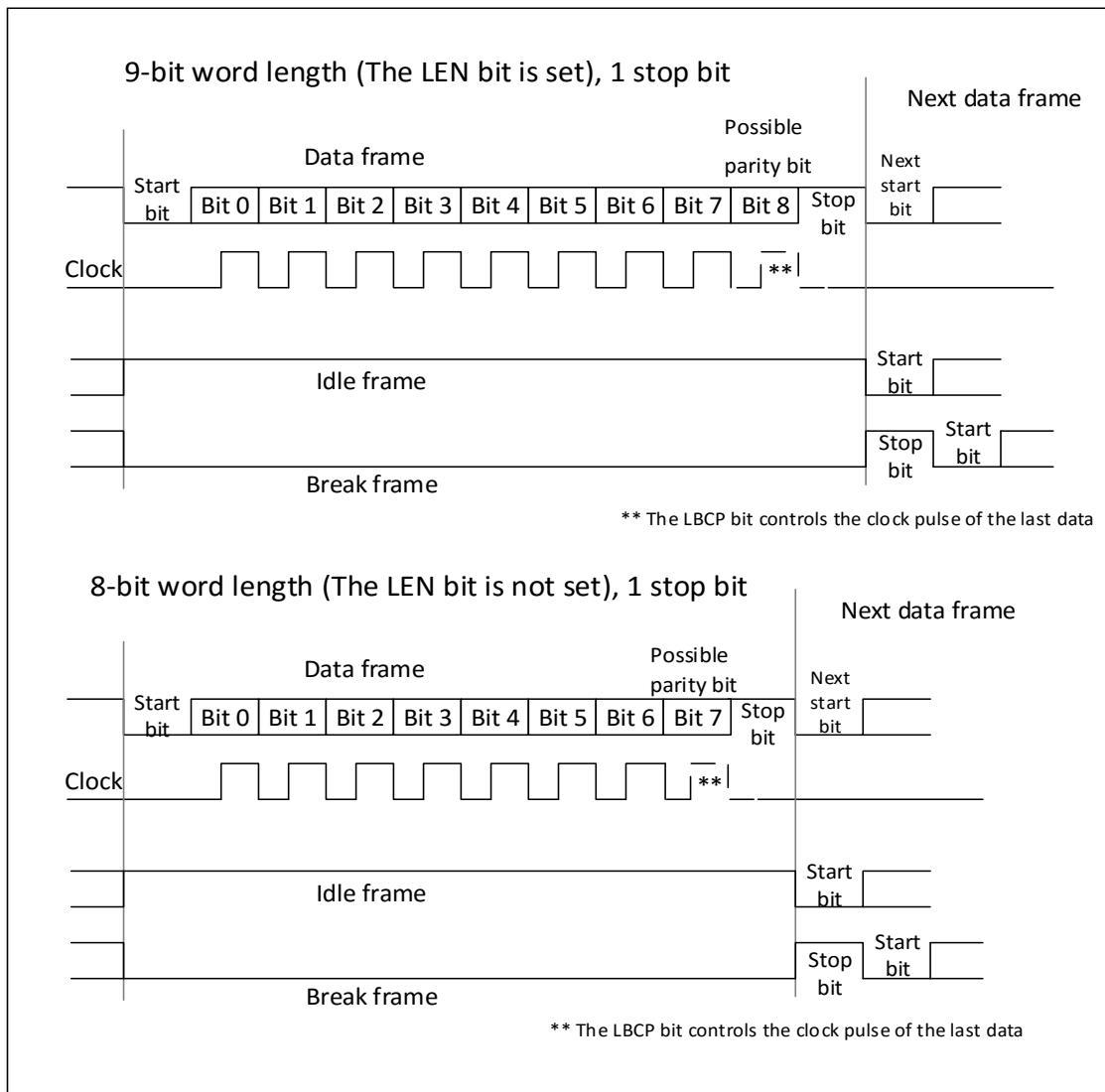
Word length may be selected as 8 bits or 9 bits by programming the LEN bit in the USART_CTRL1 register (See [Figure 16-2](#)). The Tx pin is at low state during the start bit. It is at high state during the stop bit.

Idle frame: A complete data frame of all '1', also including the stop bits of data. For example: If LEN = 0, idle frame consists of ten '1'; if LEN = 1, then idle frame consists of eleven '1'.

Break frame: Receiving all '0' for a frame period (including the stop bit period, which is also '0'). At the end of the break frame, the transmitter inserts either 1 or 2 stop bits ('1') to acknowledge the start bit.

Transmission and reception are driven by a common baud rate generator, the clock for each is generated when the enable bits of the transmitter and receiver are set respectively.

Figure 16-2 Word Length Programming



Note: In this chapter, if it is not specified, “set” means that a register is set as ‘1’, while “reset” and “clear” means that a register is set as ‘0’. Hardware or programs can set or reset a register. Please refer to the detailed descriptions in this chapter.

16.3.2 Transmitter

The transmitter can send either 8-bit or 9-bit data words depending on the status of the LEN bit. When the transmit enable bit (TEN) is set, the data in the transmit shift register is outputted on the Tx pin; the corresponding clock pulses are outputted on the CK pin through configuration.

16.3.2.1 Character Transmission

During a USART transmission, shift out the least significant bit of data first on the Tx pin. In this mode, the USART_DT register includes a buffer between the internal bus and the transmit shift register (See [Figure 16-1](#)).

Every character is preceded by a low level start bit. The number of the stop bits followed is configurable. The USART supports several stop bit configurations: 0.5, 1, 1.5, and 2 stop bits.

- Note:
1. The TEN bit cannot be reset during data transmission, or the data on the Tx pin will be corrupted as the baud rate counters stop counting. The current data being transmitted will be lost.
 2. After the TEN bit is enabled, the USART will automatically send an idle frame.

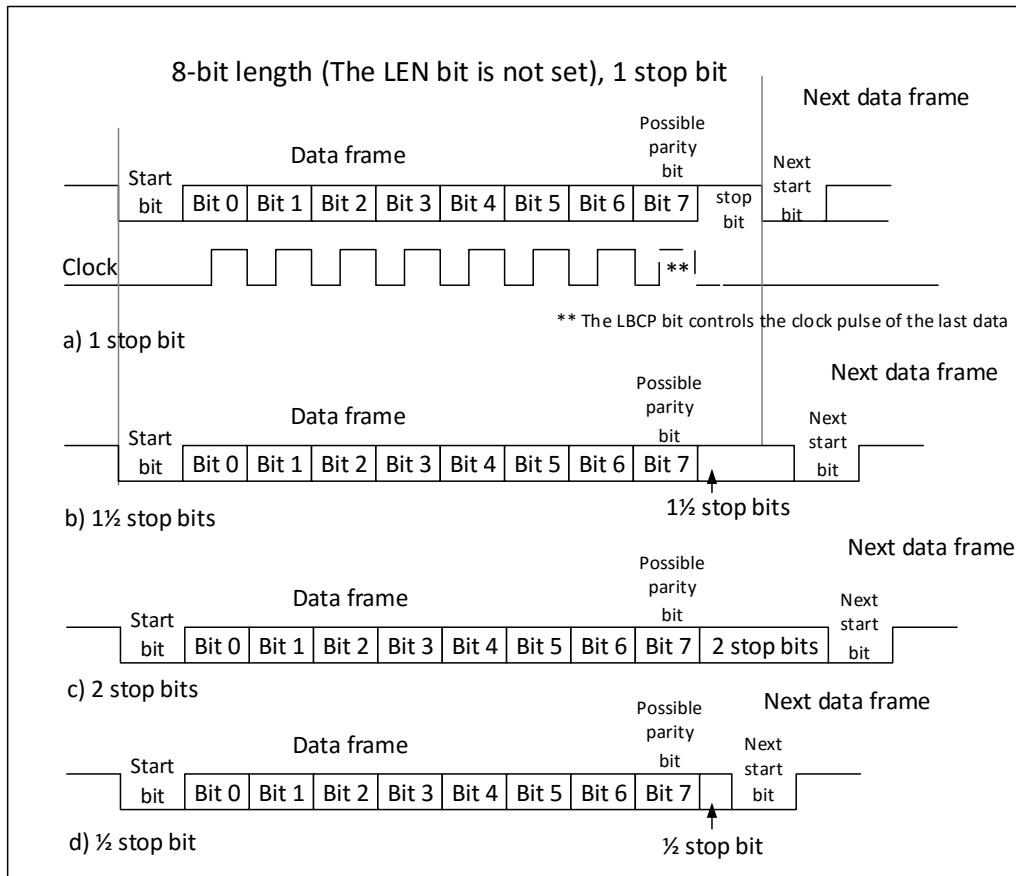
16.3.2.2 Configurable Stop Bit

The number of stop bits transmitted by each character can be programmed by the bits 13 and 12 in the control register 2 (USART_CTRL2).

1. 1 stop bit: Default value
2. 2 stop bits: Used in regular USART mode, single-wire mode, and modem mode
3. 0.5 stop bit: Used when receiving data in Smartcard mode
4. 1.5 stop bits: Used when transmitting and receiving data in Smartcard mode. An idle frame includes the stop bits.

A break frame is 10-bit low level followed by the configured number of stop bits (when LEN = 0), or 11-bit low level followed by the configured number of stop bits (when LEN = 1). It is not possible to transmit longer break frames (length greater than 10 bits or 11 bits).

Figure 16-3 Stop Bits Configuration



Configuration procedure:

1. Enable the USART by writing 1 into the UEN bit in the USART_CTRL1 register.
2. Program the LEN bit in USART_CTRL1 to define the word length.
3. Program the STOPB bit in USART_CTRL2 to define stop bit.
4. If multi-buffer communication is adopted, program the DMA enable bit (DMATEN) in USART_CTRL3. The DMA register is configured according to the description of multi-buffer communication.
5. Program the USART_BAUDR register to define baud rate.
6. Program the TEN bit in USART_CTRL1 (that is, write 1 to the TEN bit), and the USART will automatically send an idle frame as the first data transmission.
7. Write the data to be sent in the USART_DT register (which clears the TDE bit). Repeat this step to each data to be transmitted in single buffer cases.
8. After writing the last data into the USART_DT register, wait until TRAC = 1. This indicates the end of the last frame transmission. When the USART needs to be disabled or before entering Halt mode, the end of transmission should be confirmed to avoid corrupting the last transmission.

16.3.2.3 Single Byte Communication

The TDE bit is always cleared by a write to the data register.

The TDE bit is set by hardware (that is, the USART IP core), and this indicates:

- The data is moved from the TDR register (Transmitter Data Register) to the shift register, and the data transmission starts.
- The TDR register is empty.
- The next data can be written in the USART_DT register without overwriting the previous data.

This flag generates an interrupt if the TDEIEN bit is set. If the USART is transmitting data at this time, a write access to the USART_DT register stores the data in the TDR register, and the data is copied to the shift register at the end of the current transmission.

If the USART is not transmitting, a write access to the USART_DT register places the data directly in the shift register, the data transmission starts, and the TDE bit is immediately set (that is, written '1').

If a frame transmission is completed (after the stop bit), and the TDE bit is set, the TRAC bit goes high. An interrupt is generated if the TRACIEN bit is set in the USART_CTRL1 register.

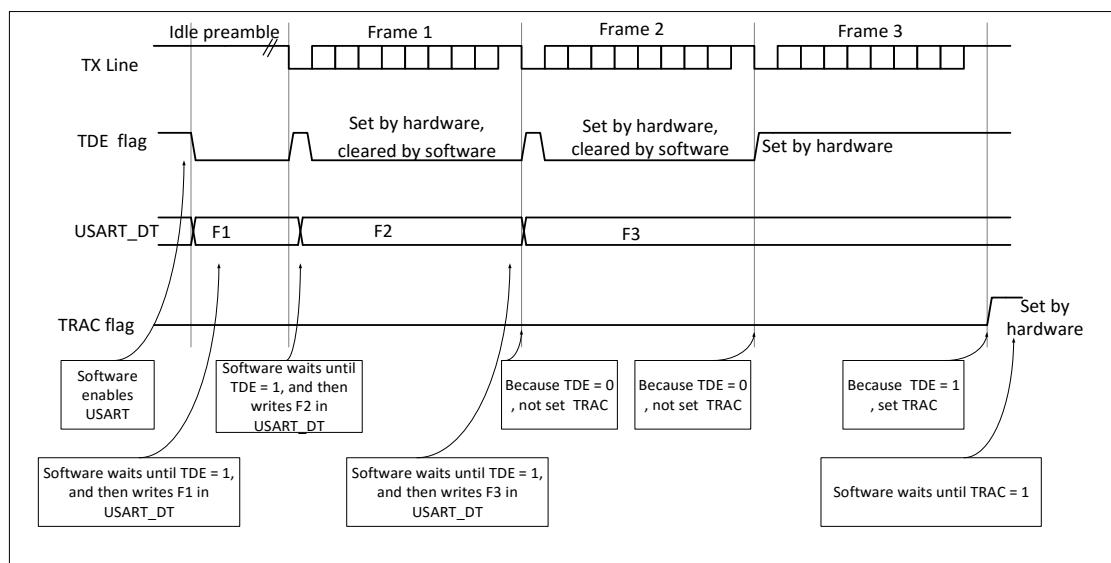
After writing the last data into the USART_DT register, it is mandatory to wait for TRAC = 1 before disabling the USART or making the microcontroller enter low-power mode (See [Figure 16-4](#)).

The TRAC bit is cleared by the following software sequence:

1. A read to the USART_STS register
2. A write to the USART_DT register

Note: *The TRAC bit can also be cleared by writing a '0' with software. This clearing sequence is recommended only in multi-buffer communication mode.*

Figure 16-4 TRAC/TDE Behavior during Transmission



16.3.2.4 Break Frame

Setting the SBRK bit transmits a break frame. The length of break frame depends on the LEN bit (See [Figure 16-2](#)). If the SBRK = 1, a break frame is sent on the Tx line after completing the current data transmission. The SBRK bit is reset by hardware when the break character transmission is completed (during the stop bit of the break frame). The USART inserts a logic '1' at the end of the last break frame to guarantee the recognition of the start bit of the next frame.

Note: *If the software resets the SBRK bit before the break frame transmission, the break frame will not be transmitted. For two consecutive breaks, the SBRK bit should be set after the stop bit of the previous break.*

16.3.2.5 Idle Character

Setting the TEN bit drives the USART to send an idle frame before the first data frame.

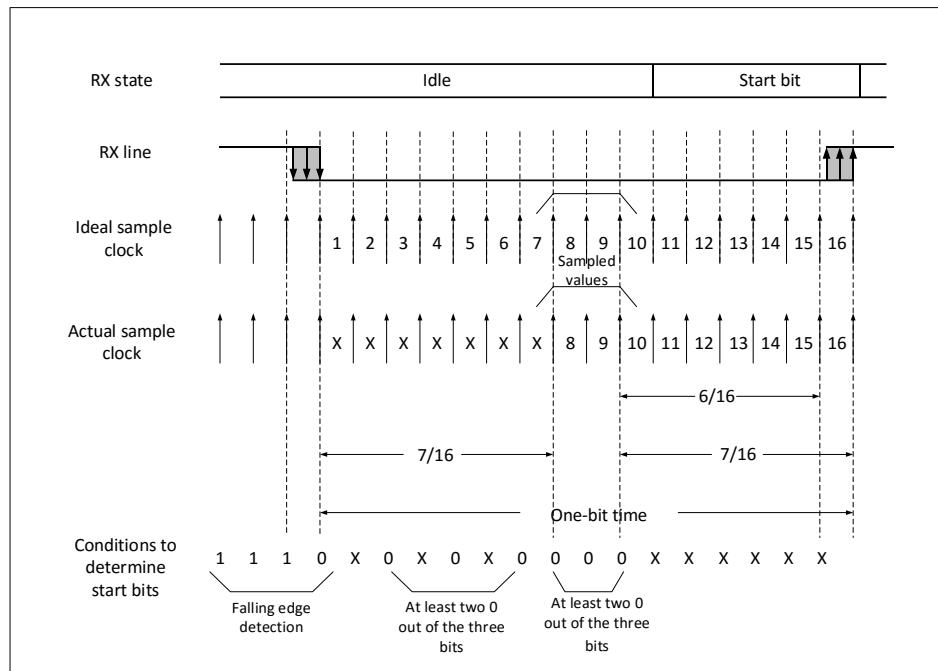
16.3.3 Receiver

The USART can receive data words of either 8 bits or 9 bits depending on the LEN bit in the USART_CTRL1 register.

16.3.3.1 Start Bit Detection

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1110X0X0X0000.

Figure 16-5 Start Bit Detection



Note: If the sequence is not complete, the receiver aborts the start bit detection and returns to idle state (no flag is set), where it waits for a falling edge.

1. The start bit is confirmed if the sampling on the bits 3, 5, and 7, as well as the bits 8, 9, and 10 are at 0 (that is, six '0'). The NERR bit, noise flag, is not set.
2. The start bit is confirmed if there are two bits at '0' in the sampling on the bits 3, 5, and 7, and at the same time, two bits at '0' in the sampling on the bits 8, 9, and 10. However, the NERR bit, noise flag, will be set.
3. The start bit is confirmed if there are two bits at '0' in the sampling on the bits 3, 5, and 7, and at the same time, three bits at '0' in the sampling on the bits 8, 9, and 10. However, the NERR bit, noise flag, will be set.
4. The start bit is confirmed if there are three bits at '0' in the sampling on the bits 3, 5, and 7, and at the same time, two bits at '0' in the sampling on the bits 8, 9, and 10. However, the NERR bit, noise flag, will be set.
5. If the sampling values of the bits 3, 5, 7, 8, 9, and 10 do not meet the above four requirements, then the USART receiver considers that no correct start bits are received, and it will abort the start bit detection and return to idle state waiting for a falling edge.

16.3.3.2 Character Reception

During a USART transmission, the least significant bits of data are shifted in first from the Rx pin. In this mode, the USART_DT register includes a buffer between the internal APB bus and the receive shift register.

Configuration procedure:

1. Enable the USART by writing 1 to the UEN bit in the USART_CTRL1 register.
2. Program the LEN bit in the USART_CTRL1 to define the word length.

3. Program the STOPB bit in the USART_CTRL2 to define stop bit.
4. If multi-buffer communication is adopted, program the DMA enable bit (DMAREN) in USART_CTRL3. The DMA register is configured according to the description of multi-buffer communication.
5. Program the USART_BAUDR register to define baud rate.
6. Program the REN bit in USART_CTRL1 (that is, write 1 to the REN bit), and make it begin to find start bit.

When a character is received:

- The RDNE 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 (as well as its associated error flags).
- An interrupt is generated if the RDNEIEN bit is set.
- The error flags will be set if a framing error, noise, or an overrun error is detected during reception.
- In multi-buffer communication, RDNE is set after every byte is received, and it is cleared when the DMA reads the data register.
- In single buffer mode, the RDNE bit is cleared by software reading to the USART_DT register. The RDNE flag can also be cleared by writing 0 to it. The RDNE bit must be cleared before the end of the next character reception to avoid an overrun error.

Note: *The REN bit should not be reset while receiving data. If the REN bit is cleared during reception, the reception of the current byte will be lost.*

16.3.3.3 Break Frame

When a break frame is received, the USART handles it as a framing error.

Note: In LIN mode, when a break frame is received, it is handled as a break frame, and LBDF will be set.

16.3.3.4 Idle Frame

When an idle frame is detected, it is handled as a normal data frame reception. However, an interrupt will be generated if the IDLEIEN bit is set.

16.3.3.5 Overrun Error

An overrun error occurs if a received character is not read yet, which causes RDNE not reset in time, and another character is received. Data cannot be transferred from the shift register to the RDR register until the RDNE bit is cleared. The RDNE flag is set by hardware after every byte is received. An overrun error occurs if the RDNE flag is set when the next data is received, or the previous DMA request is not responded.

When an overrun error occurs:

- The ORERR bit is set.
- The RDR content will not be lost. The previous data is still available when reading the USART_DT register.
- The shift register will be overwritten. After that, any data received will be lost.
- An interrupt is generated if either the RDNEIEN bit is set or both the ERRIEN and DMAREN bits are set.
- The ORERR bit is reset by reading the USART_STS and then the USART_DT registers in order.

Note: *When the ORERR bit is set, it indicates that at least 1 data is lost. There are two possibilities:*

- *If RDNE = 1, then the last valid data is still stored in the receive register, RDR, and can be read.*
- *If RDNE = 0, it means that the last valid data is already read, and thus there is nothing to be read in the RDR. This case can occur when the last valid data is read in the RDR, and at the same time the new (and lost) data is received. It may also occur when the new data is received during the reading sequence (between the USART_STS register read access and the USART_DT read access).*

For noise error, over-sampling techniques are used (except in synchronous mode) for data recovery by

discriminating between valid incoming data and noise.

Figure 16-6 Data Sampling for Noise Detection

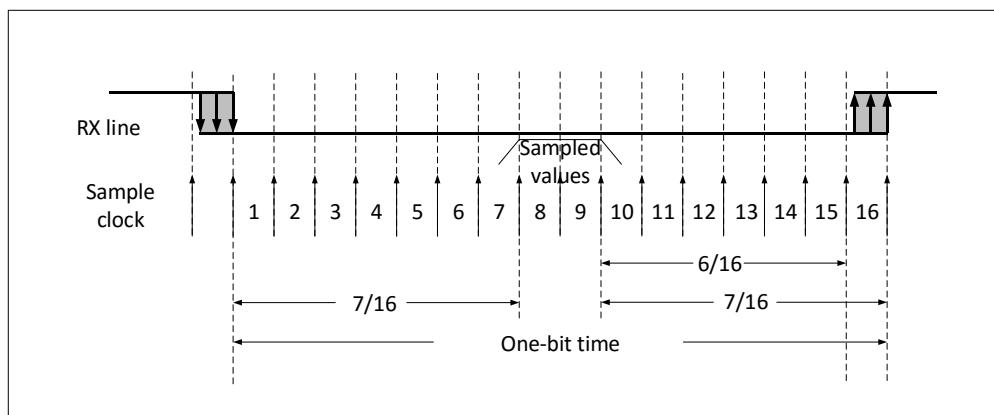


Table 16-1 Sampled Data from Noise Detection

Sampled Value	NERR Status	Received Bit	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

When noise is detected in a frame:

- The NERR flag is set at the rising edge of the RDNE bit.
- The invalid data is transferred from the shift register to the USART_DT register.
- No noise interrupt is generated in single byte communication. However, RDNE generates an interrupt since the NERR flag bit and the RDNE flag bit are set at the same time. In multi-buffer communication, an interrupt will be generated if the ERRIEN bit in the USART_CTRL3 register is set.

The NERR bit is reset by reading the USART_STS register followed by the USART_DT register in sequence.

16.3.3.6 Framing Error

A framing error is detected when: The stop bit is not recognized on reception at the expected time because of data desynchronization or excessive noise.

When a framing error is detected:

- The FERR bit is set by hardware.
- The invalid data is transferred from the shift register to the USART_DT register.
- No framing error interrupt is generated in single byte communication. However, this bit and the RDNE bit are set at the same time, and the latter will generate an interrupt. In multi-buffer communication, an interrupt will be generated if the ERRIEN bit is set in the USART_CTRL3 register.

The FERR bit is reset by reading the USART_STS register followed by the USART_DT register in sequence.

16.3.3.7 Configurable Stop Bits during Reception

The number of stop bits to be received can be configured through the control bits of the control register 2 (STOPB in USART_CTRL2). It can be either 1 or 2 in normal mode, and 0.5 or 1.5 in Smartcard mode.

1. 0.5 stop bit (reception in Smartcard mode): No sampling is done for 0.5 stop bit. Therefore, framing error and break frame cannot be detected when 0.5 stop bit is selected.
2. 1 stop bit: Sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
3. 1.5 stop bits (Smartcard mode): When transmitting in Smartcard mode, the device must check that the data is correctly sent. Thus, the receiver block must be enabled (REN = 1 in the USART_CTRL1 register), and sample the signals on the data line during stop bit transmission period. If parity error is detected, the Smartcard forces the data signal low when the transmitter samples the NACK signal, that is, within the corresponding time of stop bit on the bus, which indicates a framing error. Then, FERR is set at the end of the 1.5 stop bits. Sampling for 1.5 stop bits is done on the 16th, 17th, and 18th samples. For the Smartcard transmitter, the 1.5 stop bits can be decomposed into 2 parts: one 0.5 baud clock period during which nothing is done, followed by 1 normal stop bit period during which sampling occurs halfway through. For the Smartcard receiver, the 1.5 stop bits can also be decomposed into 2 parts: one 0.5 baud clock period during which parity check is done. If parity error occurs, the data signal is forced low in the following 1 period. If there is no parity error, release the data signal (which is at high level) in the following period. Please refer to Section 16.3.11 for more details.
4. 2 stop bits: Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. If a framing error is detected during the first stop bit, the framing error flag will be set. The second stop bit is not checked for framing error. The RDNE flag will be set at the end of the first stop bit.

16.3.4 Fractional Baud Rate Generation

The baud rate for the receiver and transmitter are both set to the same value as programmed in the Integer and Decimal values of USARTDIV.

$$\frac{\text{TX}}{\text{RX}} \text{baud} = \frac{f_{CK}}{16 * \text{USARTDIV}}$$

f_{CK} is the input clock to the peripheral (PCLK1 for USART2, 3, 4, 5, and PCLK2 for USART1)

USARTDIV is an unsigned fixed point number that is coded in the USART_BAUDR register.

Note: *The baud counters are updated with the new value of the baud registers after a write to USART_BAUDR. Therefore, the baud register value should not be changed during communication process.*

16.3.4.1 How to Derive USARTDIV from USART_BAUDR Register Values

Example 1:

If DIV_Integer = 27, DIV_Decimal = 12 (USART_BAUDR = 0x1BC), then

DIV_Integer (USARTDIV) = 27

DIV_Decimal (USARTDIV) = 12/16 = 0.75

Therefore, USARTDIV = 27.75

Example 2:

To program USARTDIV = 25.62, it leads to:

DIV_Decimal = 16*0.62 = 9.92

The nearest real number: 10 = 0x0A

DIV_Integer = DIV_Integer (25.620) = 25 = 0x19

Therefore, USART_BAUDR = 0x19A

Example 3:

To program USARTDIV = 50.99, it leads to:

DIV_Decimal = 16*0.99 = 15.84

The nearest real number: $16 = 0x10 \Rightarrow \text{DIV_Decimal}[3:0]$ overflow \Rightarrow carry must be added to the integer
 $\text{DIV_Integer} = \text{DIV_Integer} (0d50.990 + \text{carry}) = 51 = 0x33$
Therefore: $\text{USART_BAUDR} = 0x330$, $\text{USARTDIV} = 0d51.000$

Table 16-2 Error Calculation for Programming Baud Rates

Baud		fPCLK = 36 MHz			fPCLK = 72 MHz			fPCLK = 100 MHz		
Number	Kbps	Actual	Value program med in the baud register	Error %	Actual	Value program med in the baud register	Error %	Actual	Value program med in the baud register	Error %
1	2.4	2.4	937.5	0%	2.4	1875	0%	2.40004	2604.125	0%
2	9.6	9.6	234.375	0%	9.6	468.75	0%	9.60061	651	0%
3	19.2	19.2	117.1875	0%	19.2	234.375	0%	19.2012	325.5	0%
4	57.6	57.6	39.0625	0%	57.6	78.125	0%	57.6037	108.5	0%
5	115.2	115.384	19.5	0.15%	115.2	39.0625	0%	115.207	54.25	0%
6	230.4	230.769	9.75	0.16%	230.769	19.5	0.16%	230.415	27.125	0.01%
7	460.8	461.538	4.875	0.16%	461.538	9.75	0.16%	460.829	13.5625	0.01%
8	921.6	923.076	2.4375	0.16%	923.076	4.875	0.16%	925.926	6.75	0.47%
9	2250	2250	1	0%	2250	2	0%	2272.73	2.75	1%
10	4500	NA	NA	NA	4500	1	0%	4545.45	1.375	1%
11	6250	NA	NA	NA	NA	NA	NA	6250	1	0%

- Note:
1. The lower the CPU clock frequency, the lower the error of a particular baud rate.
The upper limit of the achievable baud rate can be obtained with this data.
 2. Only USART1 is clocked with PCLK2 (Max. of 100 MHz). Other USARTs are clocked with PCLK1 (Max. of 100 MHz).

16.3.5 USART Receiver's Tolerance to Clock Deviation

The USART asynchronous receiver can work normally only if the whole clock system deviation is within the range of the USART receiver tolerance. The factors that contribute to the deviation include:

- DTRA: Deviation caused by transmitter error (including the deviation of the transmitter's local oscillator)
- DQUANT: Error caused by the baud rate quantization of the receiver
- DREC: Deviation of the receiver's local oscillator
- DTCL: Deviation caused by the transmission line (generally due to the asymmetry between the low-to-high transition timing and the high-to-low transition timing of transceivers)

The following should be achieved:

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} < \text{USART receiver tolerance}$$

For normal data reception, the USART receiver tolerance is equal to the maximum tolerated deviation. This depends on the following choices:

- 10-bit or 11-bit character length defined by the LEN bit in the USART_CTRL1 register
- Whether to use fractional baud rate

Table 16-3 USART Receiver Tolerance When DIV_Decimal = 0

LEN bit	NF is considered an error	NF is not considered an error

0	3.75%	4.375%
1	3.41%	3.97%

Table 16-4 USART Receiver Tolerance When DIV_Decimal! = 0

LEN bit	NF is considered an error	NF is not considered an error
0	3.33%	3.88%
1	3.03%	3.53%

Note: The data specified in Table 16-3 and Table 16-4 may slightly differ in the special cases when the received frames contain exactly 10-bit idle frames when LEN = 0 (11-bit when LEN = 1).

16.3.6 Multiprocessor Communication

Multiprocessor communication can be achieved with the USART (connecting several USARTs in a network). For instance, one of the USARTs can be the master, and its Tx output is connected to the Rx input of other USARTs. Tx outputs of the USART slaves are logically ANDed together and connected to the Rx input of the master.

In multiprocessor configurations, usually only receivers to be addressed are enabled to receive data. This reduces redundant USART operation. Devices that are not addressed can be placed in mute mode.

In mute mode:

- None of the reception status bits can be set.
- All the receive interrupts are disabled.
- The RECMUTE bit in USART_CTRL1 register is set to 1. RECMUTE can be controlled automatically by hardware or written by the software under certain conditions.

The USART can enter or exit mute mode with the following two methods, depending on the WUMODE bit in the USART_CTRL1 register:

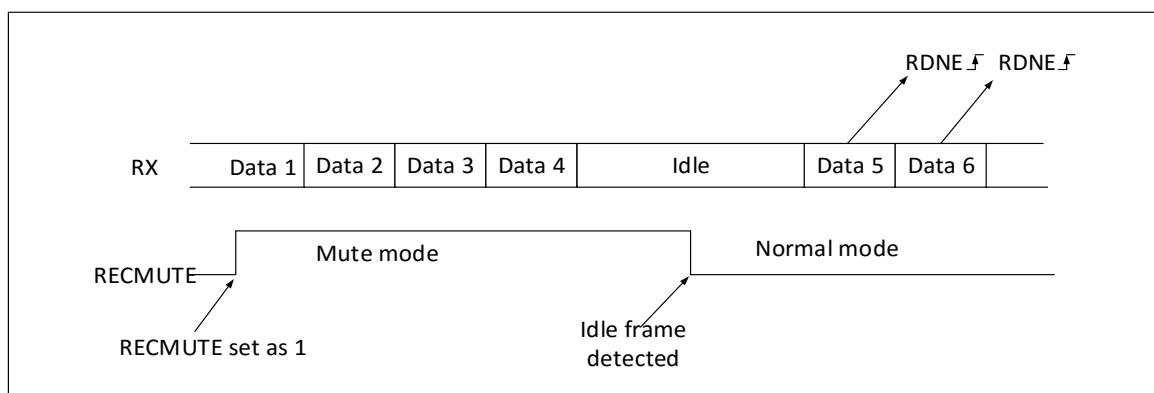
- If the WUMODE bit is reset: Perform idle line detection
- If the WUMODE bit is set: Perform address mark detection

16.3.6.1 Idle Line Detection (WUMODE = 0)

The USART enters mute mode when the RECMUTE bit is written to 1. It wakes up when an idle frame is detected. Then, the RECMUTE bit is cleared by hardware, but the IDLEF bit is not set in the USART_STS register. RECMUTE can also be written to 0 by software.

An example of mute mode behavior using idle line detection is shown in Figure 16-7.

Figure 16-7 Mute Mode Using Idle Line Detection



16.3.6.2 Address Mark Detection (WUMODE = 1)

In this mode, bytes are recognized as addresses if MSB is '1'; else, they are considered data. In an address byte, the address of the target receiver is put on the 4 LSB. This 4-bit address is compared by the receiver with its own address, which is programmed in the ADDR bits in the USART_CTRL2 register.

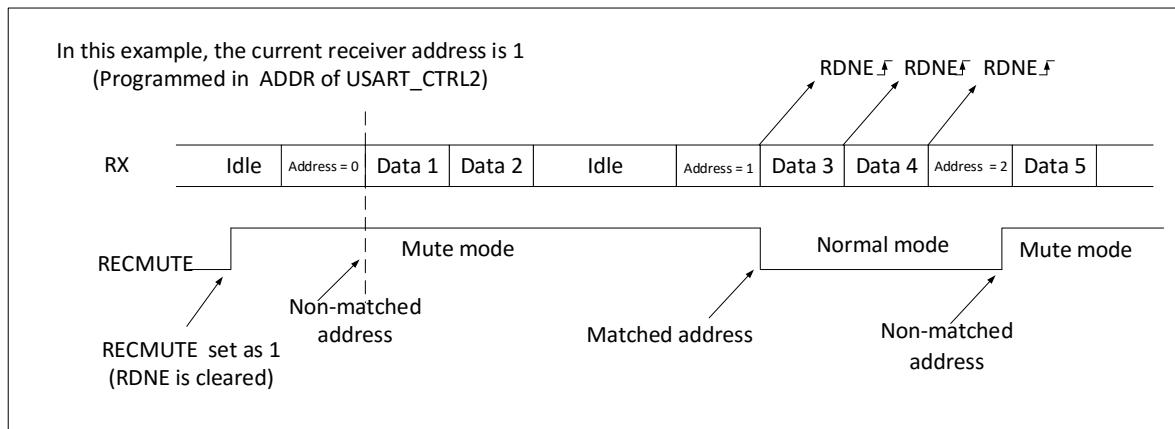
If the parity function is disabled, MSB is USART_DT[7] when LEN = 0, and MSB is USART_DT[8] when LEN = 1; if the parity function is enabled, MSB is USART_DT[6] when LEN = 0, and MSB is USART_DT[7] when LEN = 1.

The USART enters mute mode when a received address character does not match its programmed address. In this case, the RECMUTE bit is set by hardware. The RDNE flag is not set for this address byte, and no interrupt nor DMA request is issued, as the USART is in mute mode.

The USART exits from mute mode when a received address character matches the programmed address. Then, the RECMUTE bit is cleared, and the subsequent bytes are received normally. The RDNE bit is set for the address character since the RECMUTE bit is cleared.

The RECMUTE bit can be written as 0 or 1 when the receiver buffer does not contain data (RDNE = 0 in the USART_STS register). Otherwise, the write attempt is ignored. An example of mute mode behavior using address mark detection is shown in Figure 16-8.

Figure 16-8 Mute Mode Using Address Mark Detection



16.3.7 Parity Control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCEN bit in the USART_CTRL1 register. Depending on the frame length defined by the LEN bit, the possible USART frame formats are listed in Table 16-5.

Table 16-5 Frame Format

LEN bit	PCEN bit	USART Frame
0	0	Start bit 8-bit data Stop bit
0	1	Start bit 7-bit data Parity bit Stop bit
1	0	Start bit 9-bit data Stop bit
1	1	Start bit 8-bit data Parity bit Stop bit

Note: When an address mark wakes up the device, only the MSB bit of the data is taken into account for address matching, not the parity bit (MSB is the last one transmitted in data bits, and it is followed by the parity bit or the stop bit).

Even parity: The parity bit makes the 7 or 8 LSB bits in a frame and the number of '1' in the parity bit an even number. For example, there are four '1' in data = 00110101; if even parity is selected (PSEL = 0 in USART_CTRL1), the parity bit will be '0'.

Odd parity: The parity bit makes the 7 or 8 LSB bits in a frame and the number of '1' in the parity bit an odd number. For example, there are four '1' in data = 00110101; if odd parity is selected (PSEL = 1 in USART_CTRL1), the parity bit will be '1'.

Transmission mode: If the PCEN bit is set in USART_CTRL1, then the MSB bit of the data

written in the data register is transmitted after changed by the parity bit (even number of '1's if even parity is selected; odd number of '1's if odd parity is selected). If the parity check fails, the PERR flag is set in the USART_STS register, and an interrupt is generated if PERRIEN is already set in the USART_CTRL1 register.

16.3.8 LIN (Local Interconnection Network) Mode

The LIN mode is selected by setting the LINEN bit in the USART_CTRL2 register. In LIN mode, the following registers must be kept cleared:

- CLKEN in the USART_CTRL2 register, STOPB[1: 0]
- SCMEN, HALFSEL, and IRDAEN in the USART_CTRL3 register

16.3.8.1 LIN Transmission

The same procedure explained in Section 16.3.2 also applies to LIN master transmission, but for normal USART transmission, there are the following differences:

- Clear the LEN bit to configure 8-bit word length
- Set the LINEN bit to enter LIN mode. At the same time, setting the SBRK bit sends 13-bit '0' as a break frame. The break frame includes 13-bit '0' and 1-bit '1' to allow the detection on the next start bit.

16.3.8.2 LIN Reception

A break detection circuit is implemented when LIN mode is enabled. The detection is completely independent from the normal USART receiver. A break can be detected whenever it occurs, at idle state or during a frame (for example, a break is inserted while a frame is not completed).

When the receiver is enabled (REN = 1 in USART_CTRL1), the circuit monitors the Rx input for a start signal. The method for detecting start bits is the same as searching break characters or data. After a start bit is detected, the circuit samples the following bits on the 8th, 9th, and 10th samples. If 10 (when LBDLEN = 0 in USART_CTRL2) or 11 (when LBDLEN = 1 in USART_CTRL2) consecutive bits are detected as '0', and are followed by a delimiter character (which is the rising edge of Rx), the LBDF flag will be set in USART_STS. If the LBDIEN bit = 1, an interrupt is generated. Before validating the break, the delimiter should be checked, since it indicates that the Rx line returns to a high level.

If a '1' is sampled before the 10th or 11th sample, the break detection circuit cancels the current detection and searches for a start bit again. If LIN mode is disabled (LINEN = 1), the receiver continues working as normal USART, without detecting the break. If LIN mode is enabled (LINEN = 1), once a framing error occurs (i.e. the stop bit detects at '0', which will be the case for any break frame), the receiver stops until the break detection circuit receives either a '1' (if the break word is not complete) or a delimiter character (if a complete break is detected). The behavior of the break detector state machine and the break flag is shown in [Figure 16-9](#). Examples of break frames are shown in [Figure 16-10](#).

Figure 16-9 Break Detection in LIN Mode (11-bit break length - The LBDLEN bit is set)

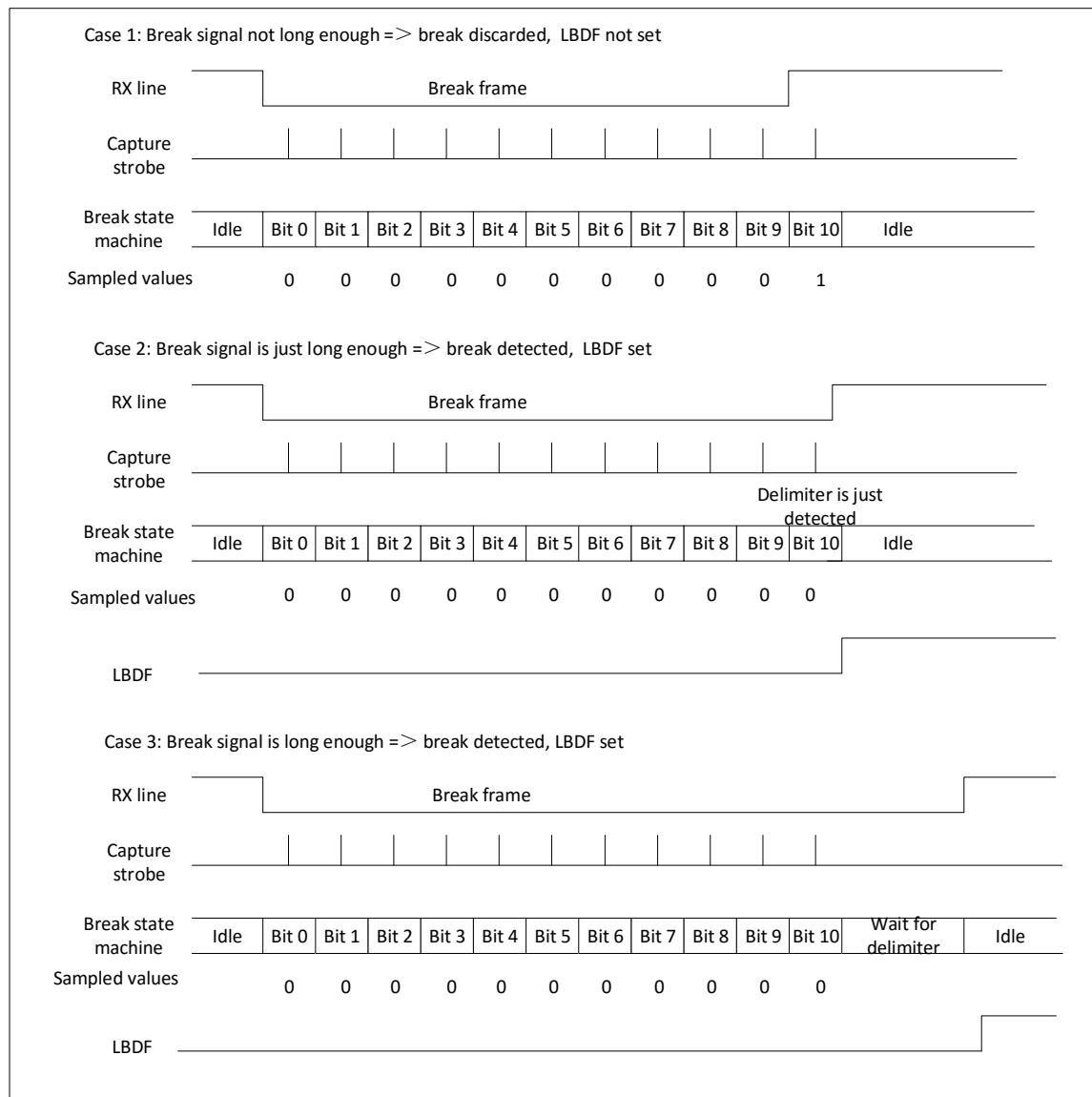
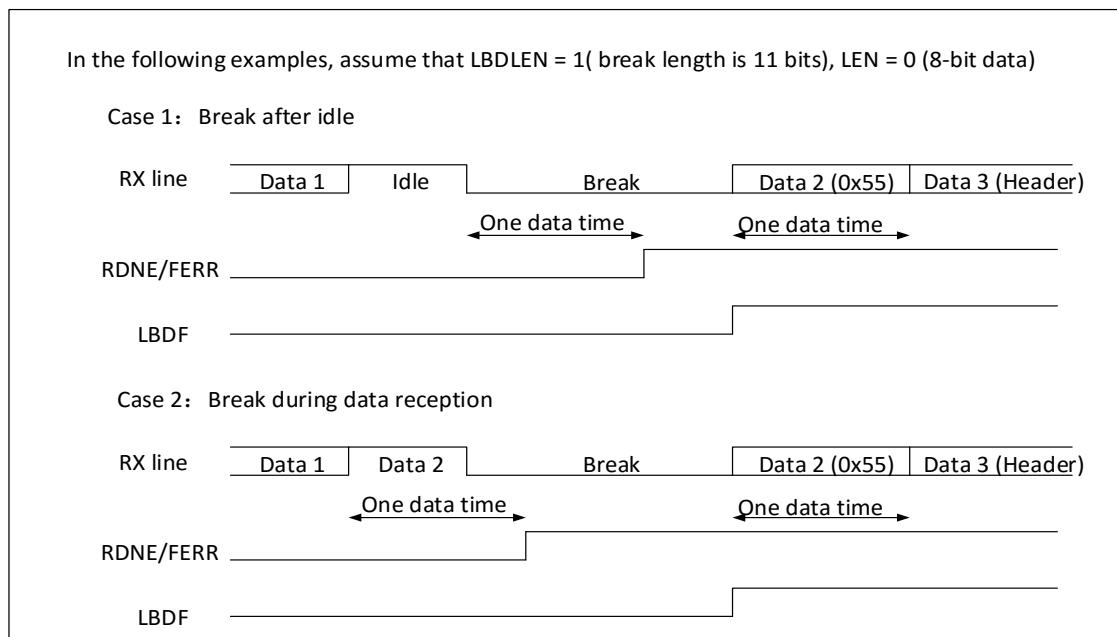


Figure 16-10 Break Detection and Framing Error Detection in LIN Mode



16.3.9 USART Synchronous Mode

The synchronous mode is selected by writing the CLKEN bit in the USART_ CTRL2 register. In synchronous mode, the following control bits must be kept cleared:

- LINEN in the USART_ CTRL2 register
- SCLEN, HALFSEL, and IRDAEN in the USART_ CTRL3 register

The USART allows users to control a bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit periods. Depending on the state of the LBCP bit in the USART_CTRL2 register, clock pulses will or will not be generated during the last valid data bit. The CLKPOL bit in the USART_CTRL2 register allows users to select the clock polarity, and the CLKPHA bit in the USART_CTRL2 register allows users to select the phase of the external clock (See [Figure 16-11](#), [Figure 16-12](#), and [Figure 16-13](#)). During idle, the external CK clock is not activated before the actual data arrives and when break is being sent. In synchronous mode, the USART transmitter works exactly like in asynchronous mode. But as CK is synchronized with Tx (according to CLKPOL and CLKPHA), the data on Tx is synchronous with CK.

In synchronous mode, the USART receiver works in a different manner compared to the asynchronous mode. If REN = 1, the data is sampled on CK (rising or falling edge, depending on CLKPOL and CLKPHA), without any oversampling. However, a setup and a hold time must be taken into account (which depends on the baud rate: 1/16-bit time).

- Note:**
1. *The CK pin works with the Tx pin. Thus, the clock is provided only if the transmitter is enabled (TEN = 1) and a data is being transmitted (writing data to USART_DT). This means that it is impossible to receive a synchronous data without transmitting data.*
 2. *The LBCP, CLKPOL, and CLKPHA bits should be configured when both the transmitter and the receiver are disabled. These bits should not be changed while the transmitter or the receiver is enabled.*
 3. *It is advised that TEN and REN should be set in the same instruction, to minimize the setup and the hold time of the receiver.*
 4. *The USART supports master mode only: It cannot receive or send data with input clocks from other devices (CK is always an output).*

Figure 16-11 Example of USART Synchronous Transmission

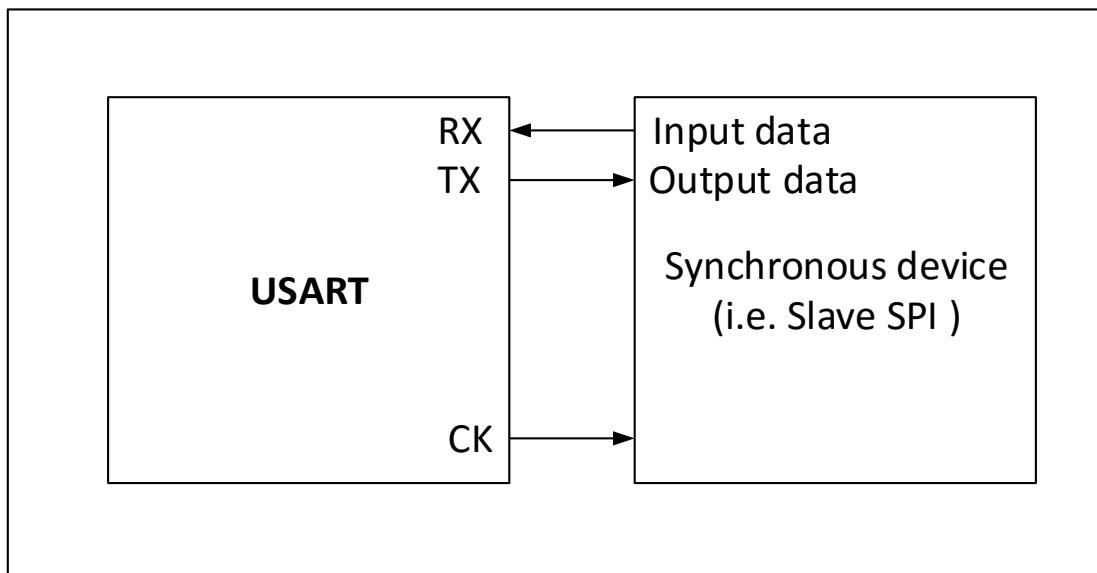


Figure 16-12 Example of USART Data Clock Timing (LEN = 0)

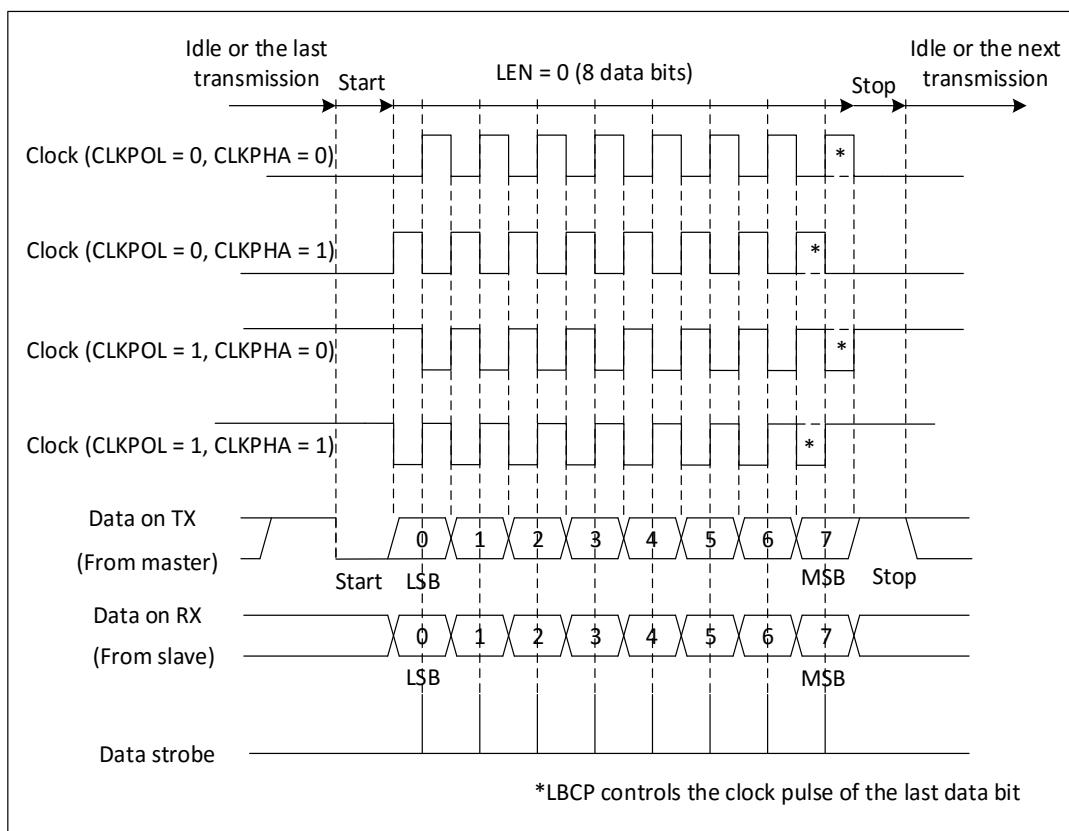


Figure 16-13 Example of USART Data Clock Timing (LEN = 1)

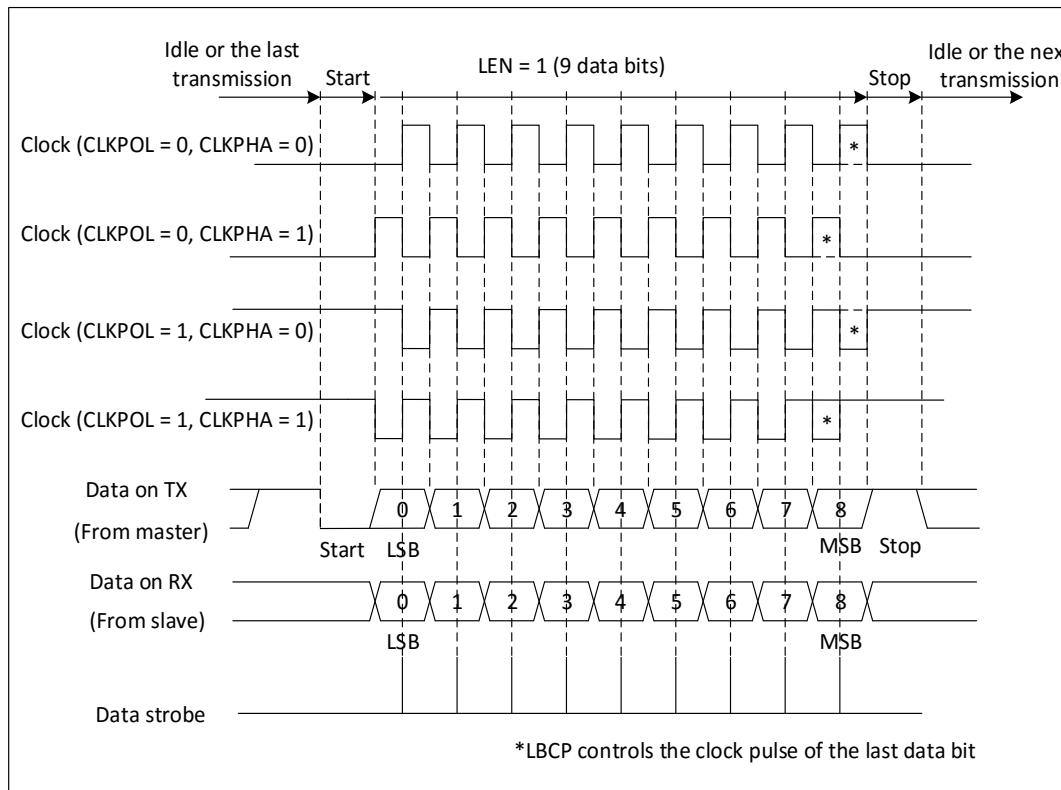
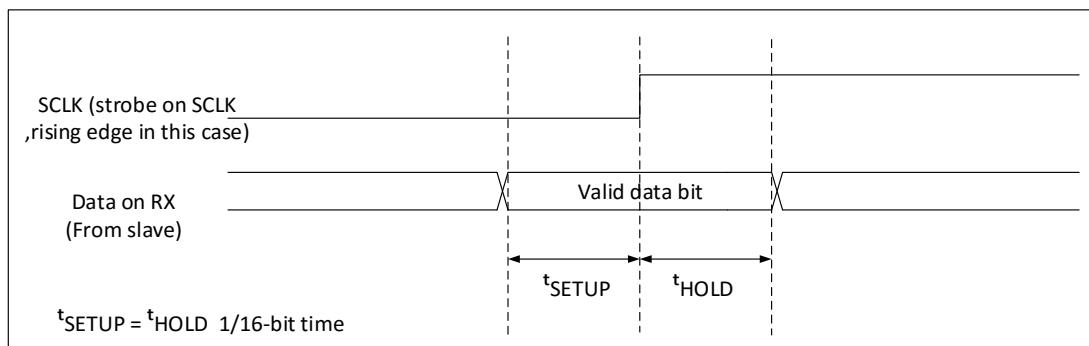


Figure 16-14 Rx Data Setup/Hold Time



Note: CK function is different in Smartcard mode, please refer to the section of Smartcard mode for more details.

16.3.10 Single-wire Half-duplex Communication

The single-wire half-duplex mode is selected by setting the HALFSEL bit in the USART_CTRL3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN in the USART_CTRL2 register
- SCMEN and IRDAEN in the USART_CTRL3 register

The USART can be configured to follow a single-wire half-duplex protocol. In single-wire half-duplex mode, the Tx and Rx pins are connected internally. The selection between half-duplex and full-duplex communication is done with the control bit, "HALFSEL" (in USART_CTRL3).

When HALFSEL is '1'

- Rx is no longer used.
- Tx is always released when no data is transmitted. Thus, it acts as a standard I/O at idle state or in reception. It means that this I/O must be set as floating input (or output high open-drain) when it is not driven by the USART. In addition, the communication is similar to normal USART mode. Conflicts on the line are managed by the software (by the use of a centralized arbiter, for instance). In particular, the transmission is never blocked by hardware; that is, the USART

cannot receive data when it is at transmission state. Namely, in half-duplex mode, transmission has higher priority over reception, and the conflicts between them should be handled by software. Transmission will continue as soon as data is written in the data register when the TEN bit is set.

16.3.11 Smartcard

The Smartcard mode is selected by setting the SCMEN bit in the USART_CTRL3 register. In Smartcard mode, the following bits must be kept cleared:

- LINEN in the USART_CTRL2 register
- HALFSEL and IRDAEN in the USART_CTRL3 register

In addition, the CLKEN bit can be set to provide a clock to the Smartcard.

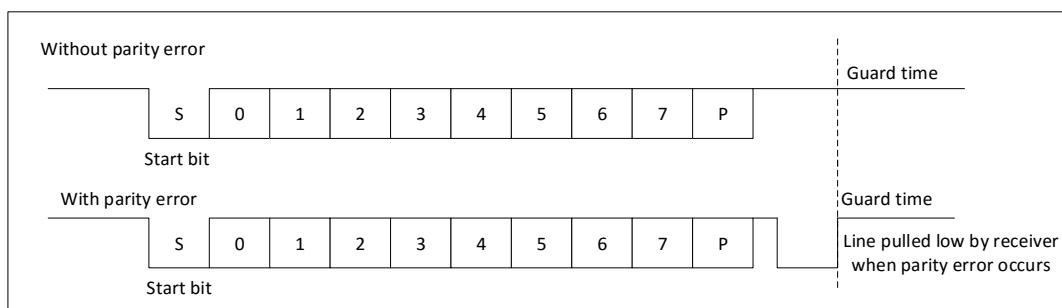
The Smartcard interface complies with ISO 7816-3 standard. It supports Smartcard asynchronous protocol. The USART should be configured as:

- 8 data bits plus parity bit: Where LEN = 1 and PCEN = 1 in the USART_CTRL1 register
- 1.5 stop bits when transmitting and receiving : Where STOPB = 11 in the USART_CTRL2 register

Note: *It is also possible to select 0.5 stop bit for receiving, but it is recommended that 1.5 stop bits be used for both transmitting and receiving to avoid switching between the two configurations.*

[Figure 16-15](#) shows examples of signals on the data line with and without parity error.

Figure 16-15 ISO7816-3 Asynchronous Protocol



When connected to a Smartcard, the USART Tx output drives a bidirectional line which is also driven by the Smartcard. The Tx pin must be configured as open drain.

Smartcard is a single-wire half-duplex communication protocol.

Transmission of data from the transmit shift register is delayed by a minimum of 1/2 baud clock. In normal operation, a full transmit shift register will start shifting data on the next baud clock edge. In Smartcard mode, this transmission is further delayed by a guaranteed 1/2 baud clock.

- If a parity error is detected during reception of a frame programmed with a 0.5 or 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame (that is, the stop bit ends). This is to notify the Smartcard that the data transmitted to USART is correctly received. This NACK signal (pulling transmit line low for 1 baud clock) will cause a framing error on the transmitter side (configured with 1.5 stop bits). The application can re-send data according to the protocol. A parity error is "NACKed" by the receiver if the NACKEN control bit is set; otherwise, a NACK is not transmitted.
- If parity error is detected, FERR of the transmitter is set when the 1.5 stop bits ends. For the Smartcard transmitter, the 1.5 stop bits can be decomposed into 2 parts: one 0.5 data bit period during which nothing is done, followed by 1 stop bit of data period during which sampling occurs halfway through. For the Smartcard receiver, the 1.5 stop bits can also be decomposed into 2 parts: one 0.5 data bit period during which parity check is done. If parity error occurs, the data signal is forced low in the following 1 period. If there is no parity error, release the data signal (which is in high level) in the following period.
- The assertion of the TRAC flag can be delayed by programming the guard time register (GTVAL). In normal operation, TRAC is asserted when the transmit shift register is empty and no further transmit requests occur. In Smartcard mode, an empty transmit shift register

triggers the guard time counter to count up to the programmed value in the guard time register (GTVAL). TRAC is forced low during this time (the time width of each bit in GTVAL). When the guard time counter reaches the programmed value, TC is asserted high (that is, set 1).

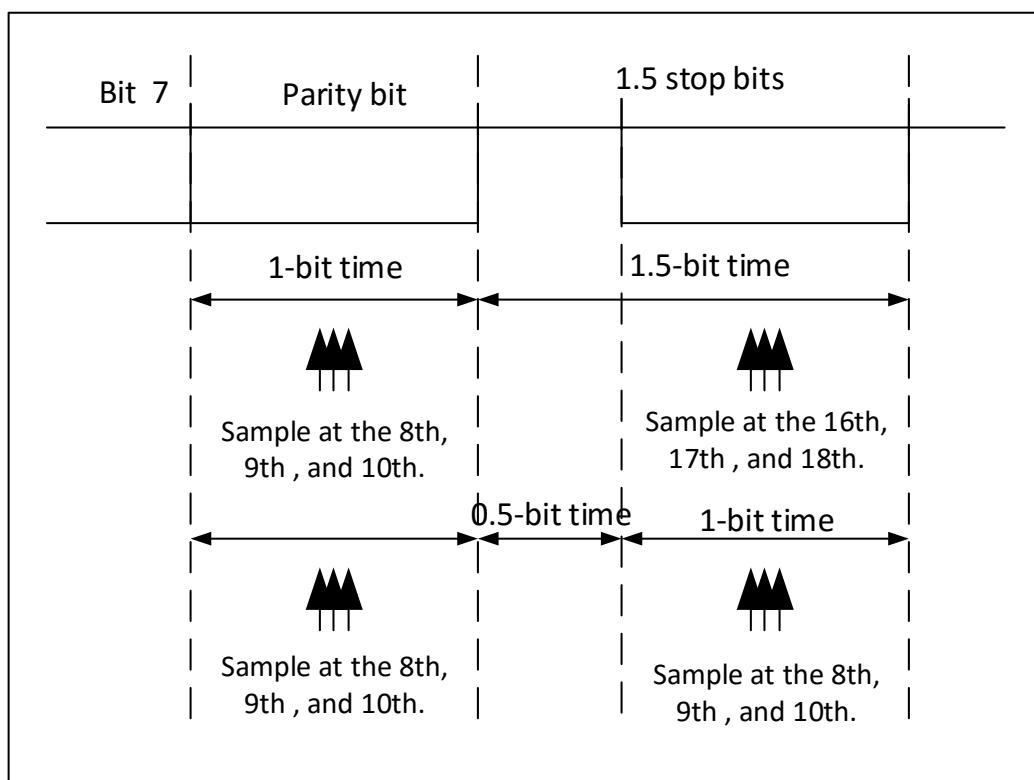
- The de-assertion of TRAC flag is not influenced by Smartcard mode.
- If a framing error is detected on the transmitter (a NACK signal from the receiver), the NACK signal will not be detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted, the receiver will not detect the NACK as a start bit.

Note:

1. A *break frame* is not significant in Smartcard mode. A 00h data with a framing error will be treated as data rather than a break.
2. No IDLE frame is transmitted when toggling the TEN bit. The IDLE frame is not defined by the ISO protocol.

Figure 16-16 details how the NACK signal is sampled by the USART. In this example, the USART transmits data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 16-16 Parity Error Detection Using 1.5 stop bits



The USART can provide a clock to the Smartcard through the CK output. In Smartcard mode, CK is not directly associated to the communication, but simply uses the internal peripheral input clock through a 5-bit prescaler to drive the clock of Smartcard. The division ratio is configured in the prescaler register, GTVAL. CK frequency can be programmed from $f_{CK}/2$ to $f_{CK}/62$, where f_{CK} is the peripheral input clock.

16.3.12 IrDA SIR ENDEC Block

The IrDA mode is selected by setting the IRDAEN bit in the USART_CTRL3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOPB, and CLKEN in the USART_CTRL2 register
- SCMEN and HALFSEL in the USART_CTRL3 register

The IrDA SIR physical layer specifies the use of a Return to Zero Inverted (RZI) modulation. This scheme uses an infrared light pulse to represent logic 0 (see [Figure 16-17](#)). The SIR transmit encoder

modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 Kbps for the SIR ENDEC. In normal mode, the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally HIGH (marking state) at the idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half-duplex communication protocol. If the transmitter is busy (i.e. the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder. If the receiver is busy (i.e. the USART is receiving decoded data from the USART), data on the Tx from the USART to IrDA is not encoded by IrDA. While receiving data, transmission should be avoided, or the data to be transmitted could be corrupted.
- According to SIR transmission logic, a '0' is transmitted as a high pulse, and a '1' is transmitted as a '0'. The width of the pulse is specified as 3/16 of the selected bit period in normal mode (see [Figure 16-18](#)).
- The SIR receive logic interprets a high state as a '1' and low pulses as '0'.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is at low state when idle.
- The SIR decoder converts the IrDA compatible receive signal into a bit stream for USART.
- The IrDA specification requires pulses greater than 1.41 us. The pulse width is programmable. The width of glitch detection logic filters on the receiver end should be less than 2 DIV periods (DIV is the prescaler value programmed in the IrDA low-power baud register, GTP). Pulse widths less than 1 DIV period are always rejected, but those of widths greater than one and less than two periods may be accepted or rejected, and those greater than 2 periods will be accepted as a pulse. The IrDA encoder/decoder doesn't work when DIV = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the STOPB bits in the USART_CTRL2 register must be configured to 1 stop bit.

IrDA low-power mode

For transmitters in low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the pulse width is 3 times of the low-power baud rate, which can be a minimum of 1.42 MHz. Generally this value is 1.8432 MHz ($1.42 \text{ MHz} < \text{DIV} < 2.12 \text{ MHz}$). A low-power mode programmable divisor divides the system clock to achieve this value. Receiver receiving in low-power mode is similar to receiving in normal mode. For glitch detection, the USART should discard pulses shorter than 1 DIV. A valid low signal is accepted only if its duration is greater than 2 periods of the IrDA low-power baud clock (DIV in USART_GTP).

- Note:**
1. *A pulse width less than two and greater than one DIV period(s) may or may not be rejected.*
 2. *The receiver set up time should be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half-duplex protocol).*

Figure 16-17 IrDA SIR ENDEC Block Diagram

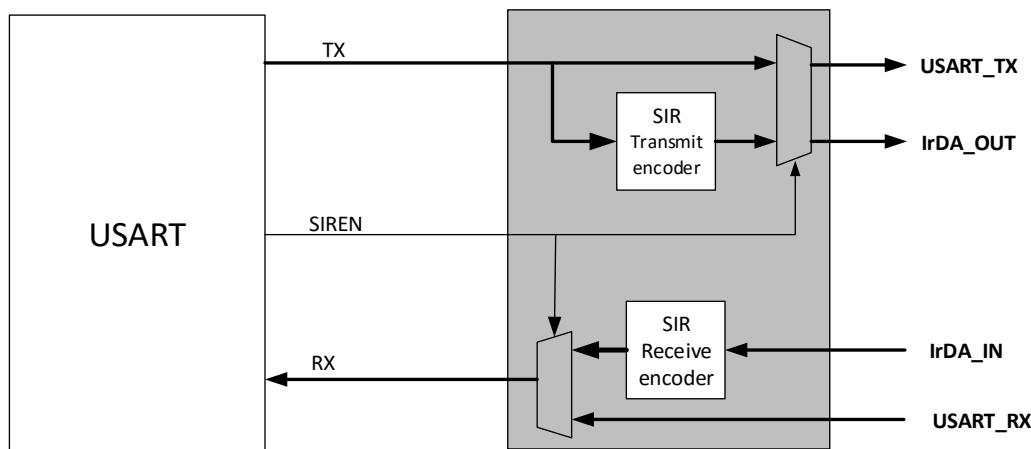
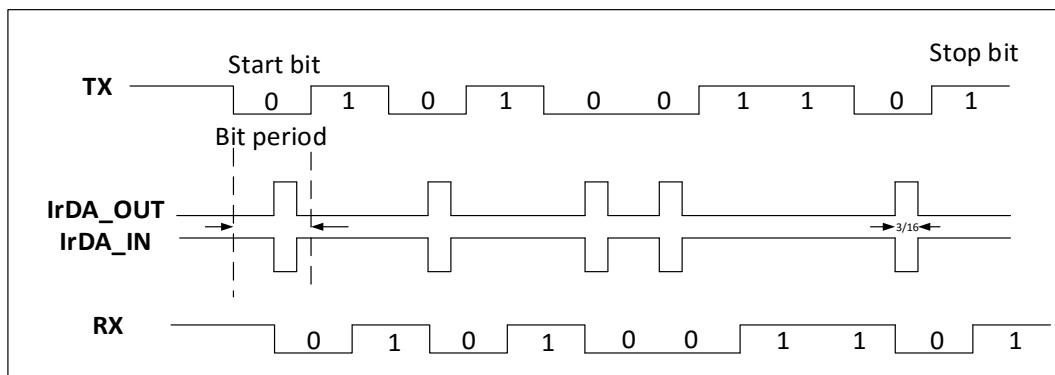


Figure 16-18 IrDA Data Modulation (3/16) - Normal Mode



16.3.13 Continuous Communication Using DMA

The USART is capable of continuing communication by using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: If DMA is not available in the product, users should use the USART as explained in [Section 16.3.2](#) or [Section 16.3.3](#). In the USART_STS register, users can clear the TDE/RDNE flags to achieve continuous communication.

16.3.13.1 Transmission Using DMA

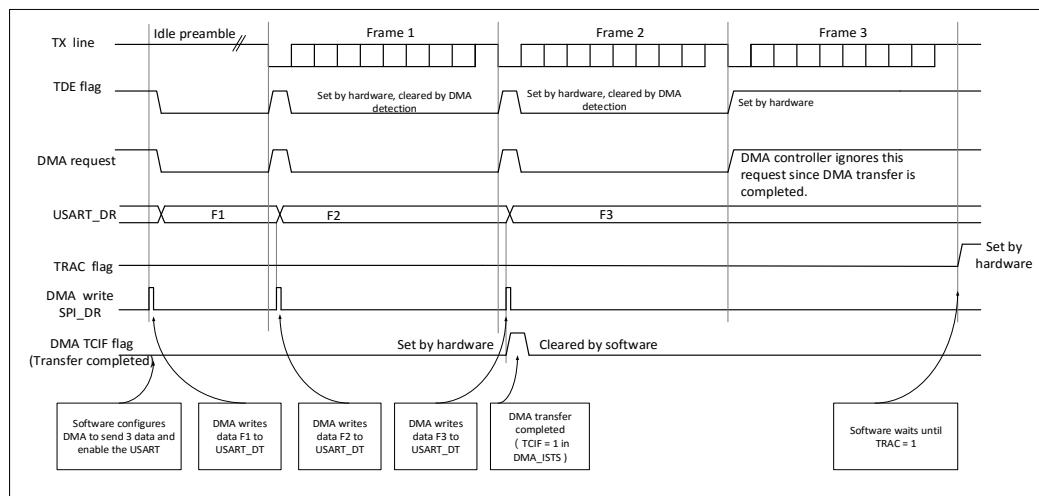
DMA mode can be enabled for transmission by setting DMATEN bit in the USART_CTRL3 register. Data is loaded from the specified SRAM area to the USART_DT register by DMA when the TDE bit is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Configure the USART_DT register address as the destination of DMA transfer in the DMA control register. Data will be sent to this address after each TDE event.
2. 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 this memory area after each TDE event.
3. Configure the total number of bytes to be transferred in the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half or full transfer as required by the application.
6. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector. In transmission mode, once the DMA finishes all the data to be transmitted, the TCIF flag is set in the DMA_ISTS register by the DMA controller. The TRAC flag can be monitored to ensure that the USART communication is completed. This avoids corrupting the last transmission before disabling the USART or entering the Stop mode. The software

must wait until TDE = 1, and then TRAC = 1.

Figure 16-19 Transmission Using DMA



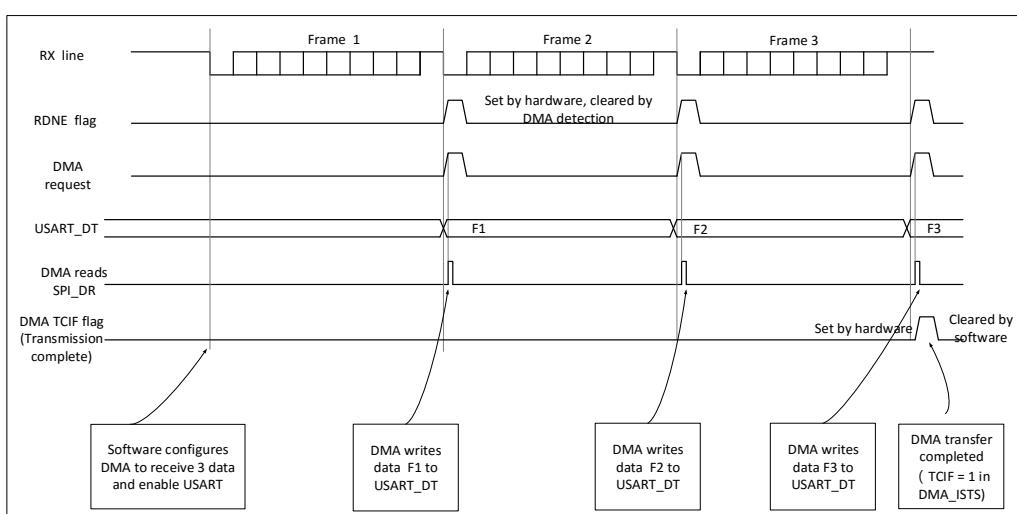
16.3.13.2 Reception Using DMA

DMA mode can be enabled for reception by setting the DMAREN bit in USART_CTRL3 register. Data is loaded from the USART_DT register to the specified SRAM area by DMA whenever a data byte is received (Please refer to the DMA descriptions). To map a DMA channel for USART reception, use the following procedure (x denotes the channel number):

1. Configure the USART_DT register address as the source of DMA transfer in the DMA control register. Data will be read from this address and sent to the memory after each RDNE event.
2. Configure the memory address as the destination of DMA transfer in the DMA control register. Data will be sent from the USART_DT register to this memory area after each RDNE event.
3. Configure the total number of bytes to be transferred in the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half or full transfer as required by the application.
6. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Figure 16-20 Reception Using DMA



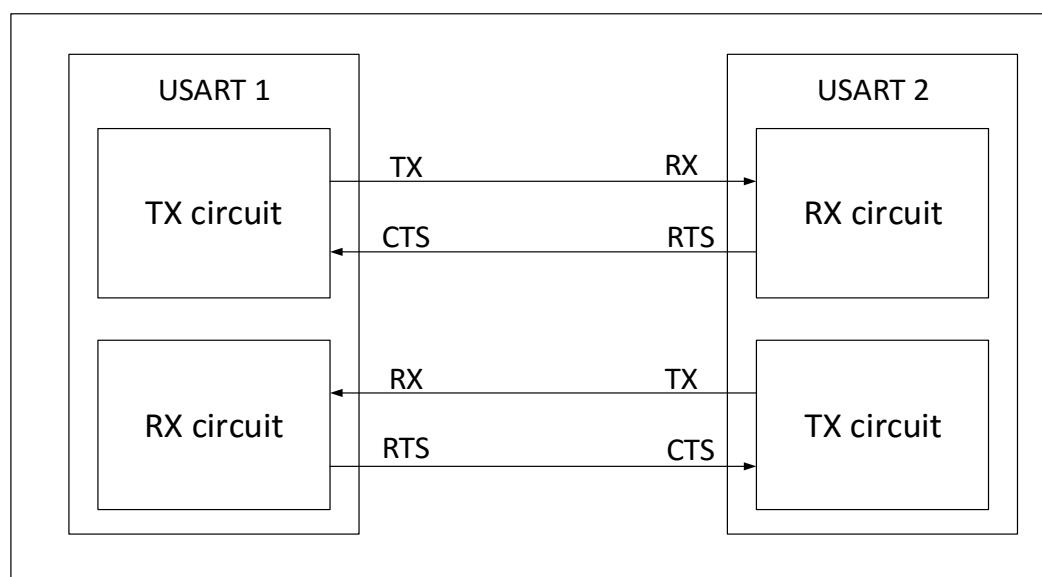
16.3.13.3 Error Flag and Interrupt Generation in Multi-buffer Communication

In multi-buffer communication, if any error occurs during the communication, the error flag will be asserted after the current byte transmission. An interrupt will be generated if the interrupt enable bit is set. In single byte reception, there will be separate error flag interrupt enable bit for framing error, overrun error, and noise flag asserted with RDNE; interrupts will be generated after the current byte transmission if these enable bits are set.

16.3.14 Hardware Flow Control

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. Figure 16-21 shows how to connect two devices in this mode:

Figure 16-21 Hardware Flow Control between Two USART

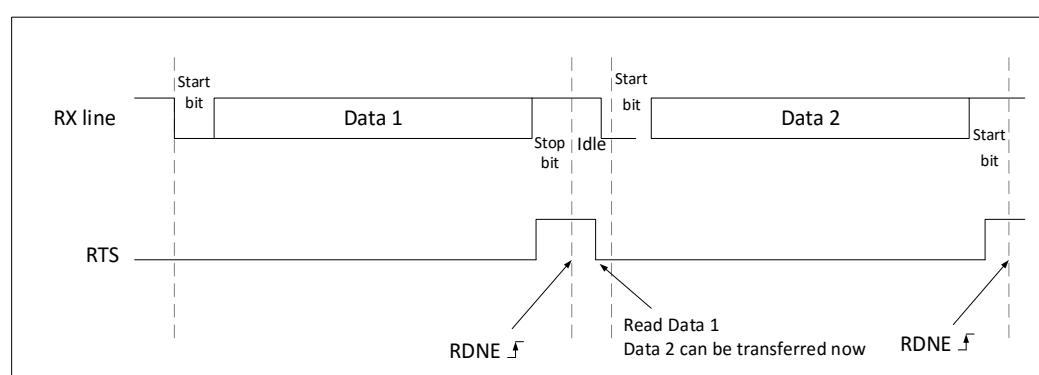


RTS and CTS flow control can be enabled independently by setting the RTSEN and CTSEN bits in the USART_CTRL3 register.

16.3.14.1 RTS Flow Control

If the RTS flow control is enabled ($RTSEN = 1$), then RTS is asserted (tied low) as long as the USART receiver is ready to receive new data. When the receive register is full (when each stop bit starts), RTS is set, indicating that the transmission is expected to stop at the end of the current frame. After RDNE is set, reading DT will clear RDNE, as well as RTS, indicating that receiver can receive the next data. Figure 16-22 shows an example of communication with RTS flow control enabled.

Figure 16-22 RTS Flow Control



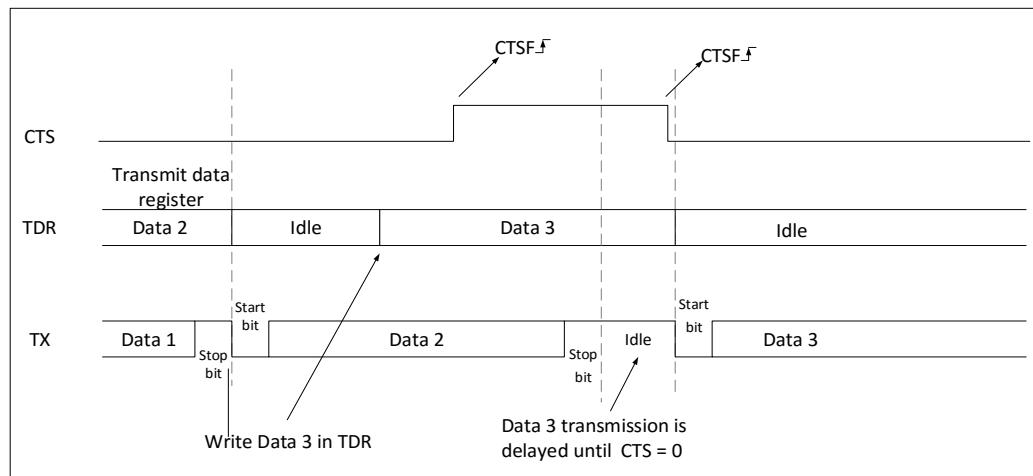
16.3.14.2 CTS Flow Control

If the CTS flow control is enabled ($CTSEN = 1$), then the transmitter checks the CTS input before transmitting the next frame. If CTS is asserted (tied low), then the next data is transmitted (assuming that the data is ready to be transmitted, in other words, if TDE = 0); if CTS is deasserted (high) during a

transmission, the transmitter stops after the current transmission is completed.

When CTSEN = 1, the CTSF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates whether the receiver is ready for communication or not. An interrupt is generated if the CTSIEN bit in the USART_CTRL3 register is set. Figure 16-23 shows an example of communication with CTS flow control enabled.

Figure 16-23 CTS Flow Control



16.4 USART Interrupt Request

Table 16-6 USART Interrupt Request

Interrupt Event	Event Flag	Enable Bit
Transmit data register empty	TDE	TDEIEN
CTS flag	CTSF	CTSIEN
Transmission completed	TRAC	TRACIEN
Received data ready to be read	RDNE	RDNEIEN
Overrun error detected	ORERR	
Idle line detected	IDLEF	IDLEIEN
Parity error	PERR	PERRIEN
Break flag	LBDF	LBDIEN
Noise flag, overrun error and framing error in multi-buffer communication	NERR or ORERR or FERR	ERRIEN ^(Note)

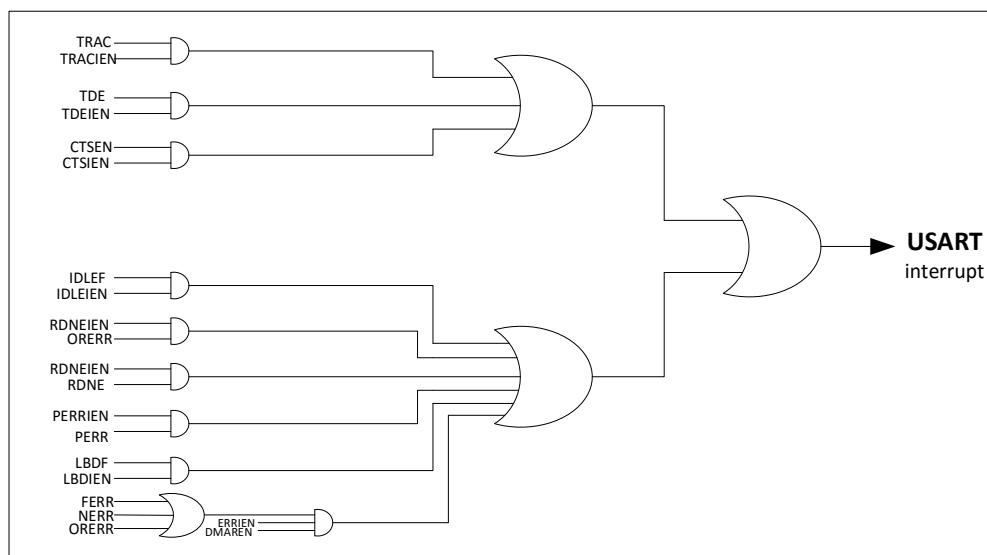
Note: This flag bit is used only when DMA is used for data reception.

The USART interrupt events are connected to the same interrupt vector (See Figure 16-24). These events include:

- During transmission: transmission completed, CTS changes, and transmit data register empty
- During reception: idle line detection, overrun error, receive data register not empty, parity error, LIN break detection, noise flag (only in multi-buffer communication), and framing error (only in multi-buffer communication)

These events generate an independent interrupt if the corresponding enable control bit is set.

Figure 16-24 USART Interrupt Mapping Diagram



16.5 USART Mode Configuration

Table 16-7 USART Mode Configuration (Note)

USART Mode	USART1	USART2	USART3	UART4	UART5
Asynchronous mode	X	X	X	X	X
Hardware flow control	X	X	X	NA	NA
Multi-buffer Communication (DMA)	X	X	X	X	NA
Multi-processor communication	X	X	X	X	X
Synchronous	X	X	X	NA	NA
Smartcard	X	X	X	NA	NA
Half-duplex (Single-wire mode)	X	X	X	X	X
IrDA	X	X	X	X	X
LIN	X	X	X	X	X

Note: X = Applicable, NA = Not applicable

16.6 USART Registers

For the abbreviations used in the register description, please refer to [Section 1.4.1](#).

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

16.6.1 USART Register Map

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_STS	Reserved																CTSF	LBDF	TDE	TRAC	RDNE	IDLEF	ORERR	NERR	FERR	PERR	0					
	0x00C0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x04	USART_DT	Reserved																DT[8:0]															
	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x08	USART_BAUDR	Reserved								DIV_Integer[13:0]												DIV_Decimal[3:0]											

16.6.2 Status Register (USART_STS)

Address offset: 0x00

Reset value: 0x00C0

Bit 6	TRAC: Transmission complete This bit is set by hardware if transmission of a frame containing data is completed and TDE = 1. An interrupt is generated if TRACIEN is '1' in the USART_CTRL1 register. It is cleared by software sequence (a read to the USART_STS register followed by a write to the USART_DT register). The TRAC bit can also be cleared by writing a '0'. This clearing sequence is recommended only for multi-buffer communication. 0: Transmission is not completed. 1: Transmission is completed.
Bit 5	RDNE: Read data register not empty This bit is set by hardware when the content of the RDR shift register is transferred to the USART_DT register. An interrupt is generated if RDNEIEN IS '1' in the USART_CTRL1 register. It is cleared by a read to the USART_DT register. The RDNE flag can also be cleared by writing '0'. This clearing sequence is recommended only for multi-buffer communication. 0: Data is not received. 1: Received data is ready to be read.
Bit 4	IDLEF: IDLE line detected This bit is set by hardware when an idle line is detected. An interrupt is generated if the IDLEIEN bit is '1' in the USART_CTRL1 register. It is cleared by a software sequence (a read to the USART_STS register followed by a read to the USART_DT register). 0: No idle line is detected, nor is idle frame. 1: Idle line is detected. Note: The IDLEF bit will not be set again until the RDNE bit is set (i.e. a new idle line is detected).
Bit 3	ORERR: Overrun error This bit is set by hardware when the data currently being received in the shift register needs to be transferred into the RDR register while RDNE = 1. An interrupt is generated if RDNEIEN = 1 in the USART_CTRL1 register. It is cleared by a software sequence (a read to the USART_STS register followed by a read to the USART_DT register). 0: No overrun error 1: Overrun error is detected. Note: When this bit is set, the RDR register content will not be lost, but the shift register will be overwritten. An interrupt is generated by ORERR flag if the ERRRIEN bit is set in multi-buffer communication mode.
Bit 2	NERR: Noise error flag This bit is set by hardware when noise is detected on a received frame. It is cleared by a software sequence (a read to the USART_STS register followed by a read to the USART_DT register). 0: No noise is detected. 1: Noise is detected. Note: This bit does not generate interrupt as it appears with the RDNE bit which generates an interrupt when its flag is set by hardware. If the ERRRIEN bit is set, an interrupt is generated when the NERR flag is set in multi-buffer communication.
Bit 1	FERR: Framing error This bit is set by hardware when a de-synchronization, excessive noise, or a break character is detected. It is cleared by a software sequence (a read to the USART_STS register followed by a read to the USART_DT register). 0: No framing error is detected. 1: Framing error or break frame is detected. Note: This bit does not generate interrupt as it appears with the RDNE bit which generates an interrupt when its flag is set by hardware. If the data currently being transferred causes both framing error and overrun error, the hardware will continue the data transmission and only set ORERR flag bit. If the ERRRIEN bit is set, an interrupt is generated when the FERR flag is set in multi-buffer communication.
Bit 0	PERR: Parity error This bit is set by hardware when parity error occurs in receiver mode. It is cleared by software sequence (a read to the USART_STS register followed by a read to the USART_DT register). The software must wait for the RDNE flag to be set before clearing the PERR bit. An interrupt is generated if PERRIEN is '1' in the USART_CTRL1 register. 0: No parity error 1: Parity error

16.6.3 Data Register (USART_DT)

Address offset: 0x04

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved																							
res																							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
Reserved								DT[8:0]															
res																							
Bit 31:9		Reserved. Forced to be '0' by hardware.																					
Bit 8:0		DT[8:0]: Data value These bits contain the received or transmitted data. It is composed of two registers, one for transmission (TDR) and one for reception (RDR). This register provides both read and write function. The TDR register provides the parallel interface between the internal bus and the output shift register (Please refer to Figure 16-1). The RDR register provides the parallel interface between the input shift register and the internal bus. When transmitting with the parity bit enabled (the PCEN bit is set in the USART_CTRL1 register), the value written in the MSB (bit 7 or bit 8, depending on the data length) will be replaced by the parity bit. When receiving with the parity bit enabled, the value read in the MSB bit is the received parity bit.																					

16.6.4 Baud Rate Register (USART_BAUDR)

Address offset: 0x08

Reset value: 0x0000

Note: If TE or RE is disabled respectively, the baud counter stops counting.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV_Integer[11:0]												DIV_Decimal[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:18		Reserved. Forced to be '0' by hardware.													
Bit 15:4		DIV_Integer[11:0]: Integer of USARTDIV The 12 bits define the integer of USART Divider (USARTDIV).													
Bit 3:0		DIV_Decimal[3:0]: Decimal of USARTDIV The 4 bits define the decimal of USART Divider (USARTDIV).													

16.6.5 Control Register 1 (USART_CTRL1)

Address offset: 0x0C

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	UEN	LEN	WUMODE	PCEN	PSEL	PERIEN	TDEIEN	TRA CIEN	RDN EIEN	IDLEI EN	TEN	REN	REC MUTE	SBRK	

res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:14	Reserved. Forced to be '0' by hardware.													
Bit 13	<p>UEN: USART enable When this bit is cleared, the USART prescalers and outputs are stopped at the end of the current byte transfer in order to reduce power consumption. This bit is set and cleared by software. 0: USART prescaler and outputs are both disabled. 1: USART is enabled.</p>													
Bit 12	<p>LEN: Word length This bit defines the word length. It is set or cleared by software. 0: 1 start bit, 8 data bits, n stop bit(s) 1: 1 start bit, 9 data bits, n stop bit(s) Note: This bit cannot be modified during a data transfer (both transmission and reception).</p>													
Bit 11	<p>WUMODE: Wakeup mode WAKE: Wakeup method This bit determines the way to wake up USART method. It is set or cleared by software. 0: Wakeup by idle line 1: Wakeup by address mark</p>													
Bit 10	<p>PCEN: Parity control enable This bit selects the hardware parity control (generation of parity bit for transmission; detection of parity bit for reception). When this bit is enabled, the computed parity is inserted at the MSB position (the 9th bit if LEN = 1; the 8th bit if LEN = 0), and parity is checked on the received data. This bit is set and cleared by software. Once it is set, parity control is active after the current byte transfer is completed. 0: Parity control is disabled. 1: Parity control is enabled.</p>													
Bit 9	<p>PSEL: Parity selection This bit selects the odd or even parity after the parity control is enabled. It is set and cleared by software. The parity selection is active after the current byte transfer is completed. 0: Even parity 1: Odd parity</p>													
Bit 8	<p>PERRIEN: PERR interrupt enable This bit is set and cleared by software. 0: Interrupt is disabled. 1: A USART interrupt is generated when PERR is '1' in the USART_STS register.</p>													
Bit 7	<p>TDEIEN: TDE interrupt enable This bit is set and cleared by software. 0: Interrupt is disabled. 1: A USART interrupt is generated when TDE is '1' in the USART_STS register.</p>													
Bit 6	<p>TRACIEN: TRAC interrupt enable This bit is set and cleared by software. 0: Interrupt is disabled. 1: A USART interrupt is generated when TDE is '1' in the USART_STS register.</p>													
Bit 5	<p>RDNEIEN: RDNE interrupt enable This bit is set and cleared by software. 0: Interrupt is disabled. 1: A USART interrupt is generated when RDNE or ORERR is '1' in the USART_STS register.</p>													
Bit 4	<p>IDLEIEN: IDLE interrupt enable This bit is set and cleared by software. 0: Interrupt is disabled. 1: A USART interrupt is generated when IDLEF is '1' in the USART_STS register.</p>													

Bit 3	TEN: Transmitter enable This bit enables the transmitter. It is set and cleared by software. 0: Transmitter is disabled. 1: Transmitter is enabled. Note 1: During transmission, except in Smartcard mode, if there is 0 pulse on the TEN bit (set '0' and then set '1'), a preamble (idle line, or idle frame) will be sent after the current byte transfer is completed., Note 2: After TEN is set, there is a 1-bit time delay before the transmission actually starts.
Bit 2	REN: Receiver enable This bit is set and cleared by software. 0: Receiver is disabled. 1: Receiver is enabled and begins searching for a start bit on Rx pin.
Bit 1	RECMUTE: Receiver wakeup This bit determines if the USART is in mute mode or not. It is set and cleared by software. It is cleared by hardware when a wakeup sequence is recognized. 0: Receiver is in active mode. 1: Receiver is in mute mode. Note 1: Before selecting mute mode (by setting the RECMUTE bit), the USART must first receive a data byte; otherwise, it cannot wake up by idle line detection in mute mode. Note 2: In address mark detection wakeup configuration (WUMODE bit = 1), the RECMUTE bit cannot be modified by software while the RDNE bit is set.
Bit 0	SBRK: Send break This bit set is used to send break characters. It can be set and cleared by software. It should be set by software and be reset by hardware during the stop bit of break. 0: No break character is transmitted. 1: Break character will be transmitted.

16.6.6 Control Register 2 (USART_CTRL2)

Address offset: 0x10

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	LIN EN	STOP B[1:0]	CLK EN	CLK POL	CLKP HA	LBCP	Reserved	LBD IEN	LBD LEN	Reserved	Reserved	ADDR[3:0]			
res	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	res	rw	rw	rw	rw
Bit 31:15		Reserved. Forced to be '0' by hardware.													
Bit 14		LINEN: LIN mode enable This bit is set and cleared by software. 0: LIN mode is disabled. 1: LIN mode is enabled. In LIN mode, the SBR bit in the USART_CTRL1 register can be used to send LIN synch breaks (13 low bits) and to detect LIN sync breaks.													
Bit 13:12		STOPB: STOP bits These bits are used for programming the number of stop bits. 00: 1 stop bit 01: 0.5 stop bit 10: 2 stop bits 11: 1.5 stop bits Note: The 0.5 stop bit and 1.5 stop bits are not available for UART4 and UART5.													

Bit 11	CLKEN: Clock enable This bit enables the CK pin. 0: CK pin is disabled. 1: CK pin is enabled. Note: This bit is not available for UART4 and UART5.
Bit 10	CLKPOL: Clock polarity This bit selects the polarity of the clock output on the SLCK pin in synchronous mode. It works in conjunction with the CLKPHA bit to produce the desired clock/data sample relationship 0: Stay low on CK pin outside transmission window. 1: Stay high on CK pin outside transmission window. Note: This bit is not available for UART4 and UART5.
Bit 9	CLKPHA: Clock phase This bit selects the phase of the clock output on the SLCK pin in synchronous mode. It works in conjunction with the CLKPOL bit to produce the desired clock/data sample relationship (Please refer to Figure 16-12 and Figure 16-13). 0: Data capture is done on the clock leading edge. 1: Data capture is done on the clock trailing edge. Note: This bit is not available for UART4 and UART5.
Bit 8	LBCP: Last bit clock pulse This bit selects whether the clock pulse of the last data bit transmitted (MSB) is outputted on the CK pin in synchronous mode. 0: The clock pulse of the last data bit is not outputted to the CK pin. 1: The clock pulse of the last data bit is outputted to the CK pin. Note 1: The last bit is the 8th or the 9th data bit transmitted (depending on the 8-bit or 9-bit format selected by the LEN bit in the USART_CTRL1 register.) Note 2: This bit is not available for UART4 and UART5.
Bit 7	Reserved. Forced to be '0' by hardware.
Bit 6	LBDIEN: LIN break detection interrupt enable Break interrupt mask (using break delimiter to detect breaks) 0: Interrupt is disabled. 1: An interrupt is generated when LBDF is '1' in the USART_STS register.
Bit 5	LBDLEN: LIN break detection length This bit is for the selection between 11-bit and 10-bit break detection. 0: 10-bit break detection 1: 11-bit break detection Note: LBDLEN can be used in break detection length control in LIN mode and other modes, and the detection length is the same as LIN mode.
Bit 4	Reserved. Forced to be '0' by hardware.
Bit 3:0	ADDR[3:0]: USART node address in this device This bit-field gives the address of the USART node. This is used in mute mode in multi-processor communication, for waking up a USART device with address mark detection.

Note: The three bits, CLKPOL, CLKPHA, and LBCP, cannot be modified after enabling transmission.

16.6.7 Control Register 3 (USART_CTRL3)

Address offset: 0x14

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
CTS IEN		CTS EN	RTS EN	DMA TEN	DMA REN	SCM EN	NAC KEN	HALF SEL	IRD ALP	IRD AEN	ERR IEN				

res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:11	Reserved. Forced to be '0' by hardware.										
Bit 10	<p>CTSIEN: CTSF interrupt enable 0: Interrupt is disabled. 1: An interrupt is generated when CTSF is '1' in the USART_STS register. Note: This bit is not available for UART4 and UART5.</p>										
Bit 9	<p>CTSEN: CTS enable 0: CTS hardware flow control is disabled. 1: CTS mode is enabled. Data can only be transmitted when the CTS input is asserted (tied low). If the CTS signal is deasserted while a data is being transmitted, then the transmission stops after this data is sent. If a data is written into the data register while CTS is deasserted, the transmission is postponed until CTS is asserted. Note: This bit is not available for UART4 and UART5.</p>										
Bit 8	<p>RTSEN: RTS enable 0: RTS hardware flow control is disabled. 1: RTS interrupt is enabled. Data is only requested when there is space in the receive buffer. The transmission is expected to stop after the current data is transmitted. The RTS output is asserted (tied low) when data can be received. Note: This bit is not available for UART5.</p>										
Bit 7	<p>DMATEN: DMA transmitter enable This bit is set and cleared by software. 0: DMA mode is disabled for transmission. 1: DMA mode is enabled for transmission. Note: This bit is not available for UART5.</p>										
Bit 6	<p>DMAREN: DMA receiver enable This bit is set and cleared by software. 0: DMA mode. DMA mode is disabled for reception. 1: DMA mode is enabled for reception. Note: This bit is not available for UART5.</p>										
Bit 5	<p>SCMEN: Smartcard mode enable This bit enables Smartcard mode. 0: Smartcard mode is disabled. 1: Smartcard mode is enabled. Note: This bit is not available for UART4 and UART5.</p>										
Bit 4	<p>NACKEN: Smartcard NACK enable 0: NACK is disabled when parity error occurs. 1: NACK is enabled when parity error occurs. Note: This bit is not available for UART4 and UART5.</p>										
Bit 3	<p>HALFSEL: Half-duplex selection Selection of single-wire half-duplex mode 0: Half-duplex mode is not selected. 1: Half-duplex mode is selected.</p>										
Bit 2	<p>IRDALP: IrDA low-power This bit is used for selecting between normal and low-power IrDA modes. 0: Normal mode 1: Low-power mode</p>										
Bit 1	<p>IRDAEN: IrDA mode enable This bit is set and cleared by software. 0: IrDA mode is disabled. 1: IrDA mode is enabled.</p>										

Bit 0	<p>ERRIEN: Error interrupt enable</p> <p>In multi-buffer communication mode, an interrupt is generated when framing error, overrun error, or noise error occurs (FERR = 1, or ORERR = 1, or NERR = 1 in USART_STS).</p> <p>0: Interrupt is disabled.</p> <p>1: An interrupt is generated when DMAREN = 1 in USART_CTRL3, and FERR = 1, or ORERR = 1, or NERR = 1 in USART_STS.</p>
-------	---

16.6.8 Guard Time and Prescaler Register (GTP)

Address offset: 0x18

Reset value: 0x0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GTVAL[7:0]								DIV[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:16	Reserved. Forced to be '0' by hardware.														
Bit 15:8	GTVAL[7:0]: Guard time value This bit-field specifies the guard time value in terms of baud clocks. This function is used in Smartcard mode. The transmission complete flag is set after this guard time. Note: This bit is not available for UART4 and UART5.														
Bit 7:0	DIV[7:0]: Prescaler value - In IrDA low-power mode: DIV[7:0] = IrDA low-power baud rate is used to divide the system clock to obtain the frequency in low-power mode: The source clock is divided by the value in the register (only 8 significant bits): 00000000: Reserved – Do not write this value. 00000001: Divide the source clock by 1 00000010: Divide the source clock by 2 - In IrDA normal mode: DIV must be set to 00000001 - In Smartcard mode: DIV[4:0]: Prescaler value is used to divide the system clock to provide clock for the Smartcard. The value in the register (5 low significant bits) is multiplied by 2, as the division factor for the source clock. 00000: Reserved – Do not write this value. 00001: Divide the source clock by 2 00010: Divide the source clock by 4 00011: Divide the source clock by 6 Note 1: Bit [7:5] have no effect in Smartcard mode. Note 2: This bit is not available for UART4 and UART5.														

17 Serial Peripheral Interface (SPI)

17.1 SPI Introduction

The SPI interface supports either the SPI protocol or the I²S audio protocol.

By default, the SPI function is selected. Users can switch from SPI mode to I²S mode by software.

The serial peripheral interface (SPI) allows half/full-duplex, synchronous, and serial communication with external devices. The interface can be configured as the master, and it provides the communication clock (SCK) for the external slave devices. The interface is also capable of operating in multimaster configuration. It may be used for a variety of purposes, including simplex synchronous transfers on two lines with a bidirectional data line or reliable communication with CRC check.

The I²S is also a 3-pin synchronous serial communication interface. It can address four different audio standards, including the I²S Philips standard, the MSB and LSB justified standards, and the PCM standard. It can operate as a slave or a master device in half-duplex mode. Master clock can be provided through the interface to an external slave device when the I²S functions as a master.

Warning: Since some SPI3/I2S3 pins are shared with JTAG pins (SPI3_NSS/I2S3_WS with JTDI and SPI3_SCK/I2S3_CK with JTDO), they are not controlled by the I/O controller and are reserved for JTAG usage (after each reset) by default. If users want to configure the SPI3/I2S3 pins, the JTAG must be disabled and switched to the SWD interface (when debugging), or disable both JTAG and SWD interfaces (during standard applications). For more information, please refer to [Section 7.4.4](#).

17.2 Main Features

17.2.1 SPI Main Features

- Full-duplex synchronous transfers on three lines
- Simplex synchronous transfers on two lines with or without a bidirectional data line
- 8-bit or 16-bit transfer frame format selection
- Master or slave operation
- Multimaster mode capability
- 8 master mode baud rate prescalers (Max. of f_{PCLK}/2)
- Slave mode frequency (Max. of f_{PCLK}/2)
- Fast communication for both master and slave
- NSS management by hardware or software in both master and slave mode: dynamic change of master/slave operations
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Dedicated transmission and reception flags with interrupt capability
- SPI bus busy status flag
- Hardware CRC for reliable communication:
 - CRC value can be transmitted as the last byte in Tx mode.
 - Automatic CRC error checking on the last received byte in full-duplex mode
- Master mode fault, overrun, and CRC error flags with interrupt capability
- 2-byte transmission and reception buffer with DMA capability: Tx and Rx requests generation

17.2.2 I²S Function Overview

- Half-duplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler to reach accurate audio sample frequencies

- (from 8 kHz to 192 kHz)
- Data format can be 16-bit, 24-bit, or 32-bit.
- Packet frame is fixed to 16-bit (16-bit data frame) or 32-bit (16-bit, 24-bit, or 32-bit data frame) by audio channel.
- Programmable clock polarity (steady state)
- Underrun flag in slave transmission mode and overrun flag in master/slave reception mode
- 16-bit data register for transmission and reception, one data register on each channel side
- I²S protocols supported
 - I²S Philips standard
 - MSB-justified standard (left-justified)
 - LSB-justified standard (right-justified)
 - PCM standard (with short and long frame synchronization on 16-bit channel frame or 16-bit data frame extended to 32-bit channel frame)
- Data direction is always MSB first.
- DMA capability for transmission and reception
- Master clock can output to an external audio component. The ratio is fixed at 256xFs (where Fs is the audio sampling frequency).

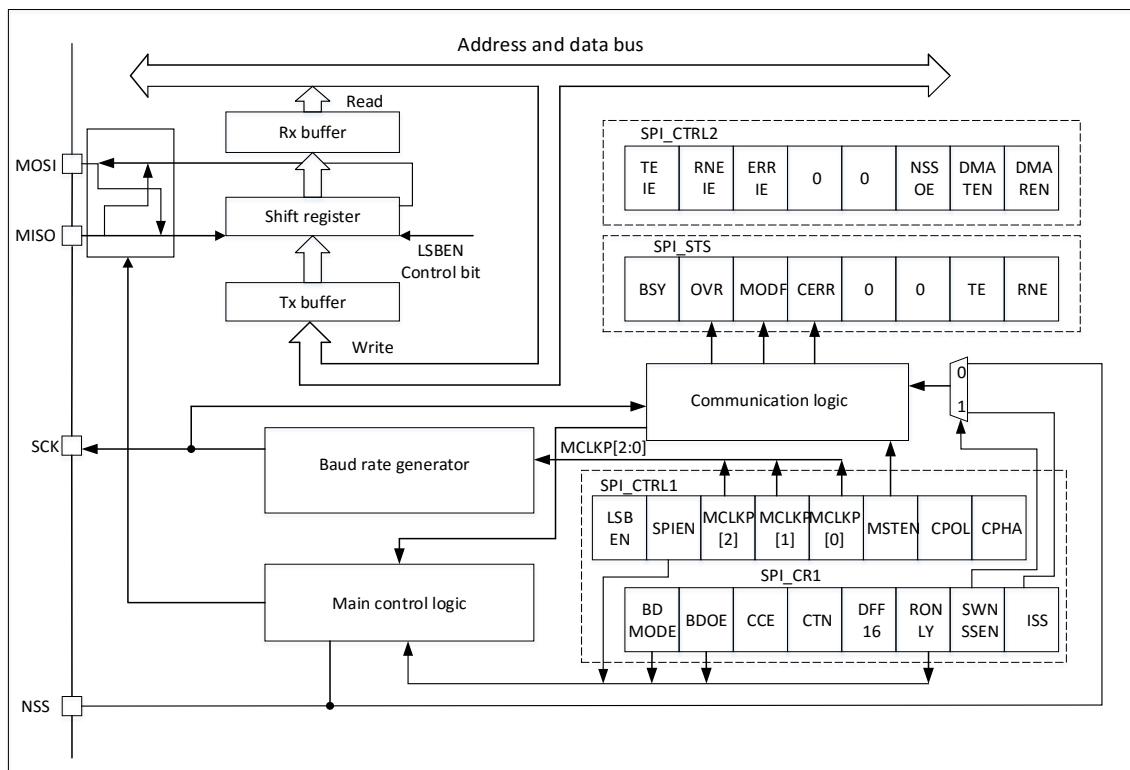
17.3 Function Overview

17.3.1 SPI Function Overview

17.3.1.1 Introduction

Figure 17-1 is the block diagram of SPI.

Figure 17-1 SPI Block Diagram

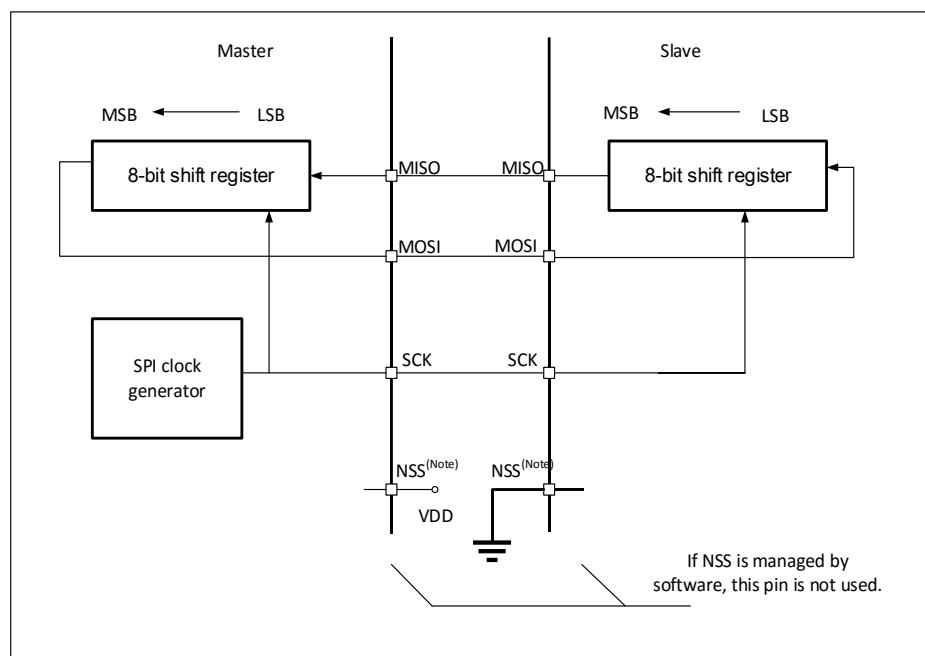


Usually, the SPI is connected to external devices through four pins:

- MISO: Master In/Slave Out. This pin transmits data in slave mode and receives data in master mode.
- MOSI: Master Out/Slave In. This pin transmits data in master mode and receives data in slave mode.
- SCK: Serial clock, as output for masters and input for slaves
- NSS: Slave select. This is an optional pin which selects master/slave devices. It acts as a “chip select” to allow the SPI master to independently communicate with certain slaves to avoid conflicts on the data lines. Slave NSS pin can be driven by a standard I/O pin on the master device. The NSS pin may also be used as an output if enabled (NSSOE bit) and driven low if the SPI is in master configuration. In this case, all NSS pins from SPI devices connected to the Master NSS pin are at a low level, and become slaves automatically when they are configured in NSS hardware mode. When configured in master mode with NSS configured as an input (MSTEN = 1, NSSOE = 0), the SPI enters fault state of the master mode if NSS is pulled low. In other words, the MSTEN bit is automatically cleared, and the device enters the slave mode.

Figure 17-2 shows an example of the interconnections between a single master and a single slave.

Figure 17-2 Single Master and Single Slave Application



Note: The NSS pin is configured as an input here.

The MOSI pins are connected together, and the MISO pins are connected together. In this way, data is transferred serially between master and slave (MSB first). The communication is always initiated by the master. When the master device transmits data to a slave device via the MOSI pin, the slave device responds via the MISO pin. This implies full-duplex communication, with synchronized data out and data in with the same clock signal, which is provided by the master device via the SCK pin.

Slave select (NSS) pin management

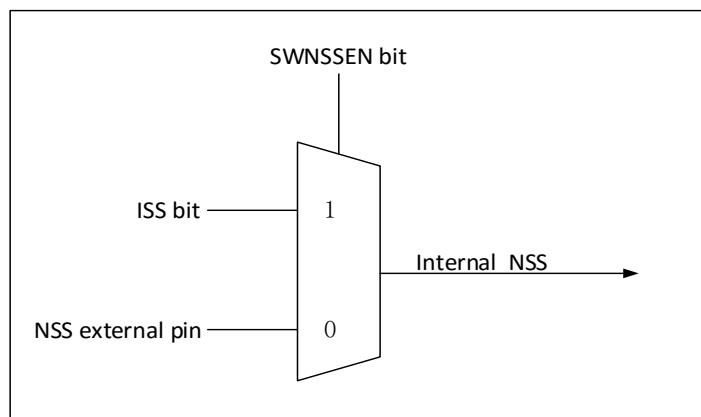
There are two NSS modes:

- Software NSS mode: This mode is enabled by setting the SWNSSEN bit in the SPI_CTRL1 register (See [Figure 17-3](#)). In this mode, NSS pin remains free for other application uses, while the internal NSS signal is driven by writing the ISS bit in the SPI_CTRL1 register.
- Hardware NSS mode, which includes two conditions:
 - NSS output enabled: When AT32F403 works as a master SPI, and NSS output is enabled by setting the NSSOE bit in the SPI_CTRL2 register, NSS pin is pulled low at this time; all SPI devices, of which NSS pins are connected to this master SPI's NSS

pins and configured as hardware NSS, will become slaves automatically. When a SPI device needs to send broadcast data, it must pull NSS signal low to notify other devices that it is the master. If it fails to pull NSS low, this means that there is another master processing communication on the bus, and a hardware failure fault (Hard Fault) will be generated.

- NSS output disabled: Allowed in multimaster mode.

Figure 17-3 Hardware/Software Slave Select Management



Clock phase and clock polarity

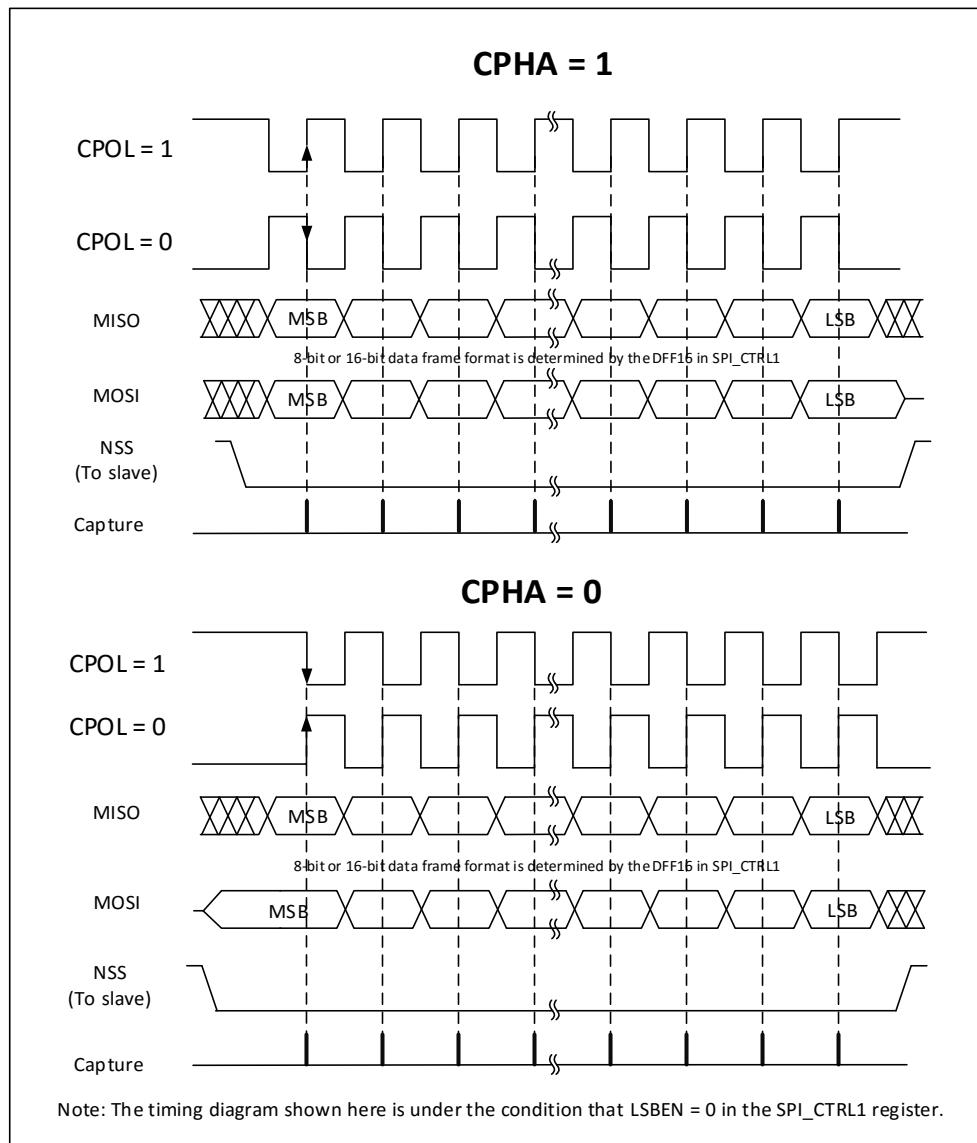
The CPOL and CPHA bits in the SPI_CTRL register can generate four possible timing relationships. The CPOL (clock polarity) bit controls the clock level at idle state when no data is being transferred. This bit takes effect both in master and slave modes. If CPOL is reset, the SCK pin stays low at idle state. If CPOL is set, the SCK pin stays high at idle state.

If the CPHA (clock phase) bit is set, the second edge on the SCK pin (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set) is the data bit capture strobe. Data are latched on the occurrence of the second clock transition. If the CPHA bit is reset, the first edge on the SCK pin (falling edge if CPOL bit is reset, rising edge if CPOL bit is set) is the data bit capture strobe. Data are latched on the occurrence of the first clock transition.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edge. [Figure 17-4](#) shows an SPI transfer with the four combinations of the CPHA and CPOL bits. This diagram may be interpreted as a master or slave timing diagram where the SCK pin, the MISO pin, the MOSI pin are directly connected between the master and the slave devices.

- Note:
1. Before modifying the CPOL/CPHA bits, the SPIEN bit should be cleared to disable SPI.
 2. Master and slave must be programmed in the same timing mode.
 3. The idle state of SCK must be consistent with the polarity selected in the SPI_CTRL1 register (at idle state, by pulling up SCK if CPOL = 1 or pulling down SCK if CPOL = 0).
 4. The data frame format (8-bit or 16-bit) is selected through the DFF16 bit in SPI_CTRL1 register, and it can also be used to determine the data length during transmission/reception.

Figure 17-4 Data Clock Timing Diagram



Data frame format

Data can be shifted out either MSB-first or LSB-first, depending on the value of the LSBEN bit in the SPI_CTRL1 register. Based on the size of the data programmed in the DFF16 bit in the SPI_CTRL1 register, each data frame is 8 bits or 16 bits long. The selected data frame format is applicable for transmission and/or reception.

17.3.1.2 Configure SPI in Slave Mode

In slave mode, the serial clock is received on the SCK pin from the master device. The value set in the MCLKP[4:0] bits in the SPI_CTRL1 register does not affect the data transfer rate.

Note: It is recommended enabling the SPI slave before the master sends the clock. Otherwise, undesired data transmission might occur. The data register of the slave should be ready before the first edge of the communication clock or before the end of the ongoing communication. The polarity of the communication clock must be set to a steady state value before the slave and the master are enabled.

Please follow the procedure below to configure SPI in slave mode:

Configuration procedure

1. Set the DFF16 bit to select 8-bit or 16-bit data frame format.
2. Configure the CPOL and CPHA bits to define the phase relationships between the data transfer and the serial clock (See [Figure 17-4](#)). For correct data transfer, the CPOL and CPHA bits must be configured in the same way in the slave device and

- the master device.
3. The frame format (MSB-first or LSB-first depending on the LSBEN bit in the SPI_CTRL1 register) must be the same as the master device.
 4. In hardware mode (Please refer to the section of slave select (NSS) pin management), the NSS pin must be connected to a low level signal during the whole byte (8-bit or 16-bit) transmit sequence. In NSS software mode, set the SWNSSEN bit and clear the ISS bit in the SPI_CTRL1 register.
 5. Clear the MSTEN bit and set the SPIEN bit (in the SPI_CTRL1 register) to assign the pins in SPI mode. In this configuration, the MOSI pin is data input, while the MISO pin is data output.

Data transmission procedure

The data byte is parallelly loaded into the Tx buffer during a write operation.

The transmit sequence begins when the slave device receives the clock signal, and the first data bit appears on its MOSI pin (Note: The first bit is sent at this time). The rest of the bits (the 7 bits in 8-bit data frame format; the 15 bits in 16-bit data frame format) are loaded into the shift register. The TE flag in the SPI_STS register is set when data is transferred from the Tx buffer to the shift register; an interrupt is generated if the TEIE bit in the SPI_CTRL2 register is set.

Data reception procedure

For the receiver, when data transfer is completed:

- Data in shift register is transferred to Rx buffer, and the RNE flag in the SPI_STS register is set.
- An interrupt is generated if the RNEIE bit is set in the SPI_CTRL2 register.

After the last sampling clock edge, the RNE bit is set, data byte received in the shift register is moved to the Rx buffer. When the SPI_DT register is read, the SPI peripheral returns this buffered value.

The RNE bit is cleared by reading the SPI_DT register.

17.3.1.3 Configure SPI in Master Mode

In the master configuration, the serial clock is generated on the SCK pin.

Configuration procedure

1. Select the MCLKP[3:0] bits in the SPI_CTRL1 register to define the serial clock baud rate.
2. Select the CPOL and CPHA bits to define one of the phase relationship between the data transfer and the serial clock (See [Figure 17-4](#)).
3. Set the DFF16 bit to define 8-bit or 16-bit data frame format.
4. Configure the LSBEN bit in the SPI_CTRL1 register to define the frame format.
5. If the NSS pin is required in input mode and hardware mode, connect the NSS pin to a high-level signal during the whole byte transmit sequence. In software mode, set the SWNSSEN and ISS bits in the SPI_CTRL1 register. If the NSS pin is required in output mode, only the NSSOE bit should be set.
6. The MSTEN and SPIEN bits must be set (They remain set only when the NSS pin is connected to a high-level signal). In this configuration, the MOSI pin is data output, and the MISO pin is data input.

Data transmission procedure

The transmit sequence begins when a byte is written in the Tx buffer. The data byte is parallelly loaded into the shift register (from the internal bus) during the first bit transmission, and then is shifted out serially to the MOSI pin; MSB-first or LSB-first depends on the LSBEN bit in the SPI_CTRL1 register. The TE flag is set when data is transferred from the Tx buffer to the shift register; an interrupt is generated if the TEIE bit in the SPI_CTRL2 register is set.

Data reception procedure

For the receiver, when data transfer is completed:

- Data-in shift register is transferred to Rx buffer, and the RNE flag is set.
- An interrupt is generated if the RNEIE bit is set in the SPI_CTRL2 register.

After the last sampling clock edge, the RNE bit is set, data byte received in the shift register is moved to the Rx buffer. When the SPI_DT register is read, the SPI peripheral returns this buffered value.

The RNE bit is cleared by reading the SPI_DT register. Once the transmission begins, a continuous transmit stream can be maintained if the next data to be transmitted is put into the Tx buffer. Please note that TE flag should be '1' before any attempt to write the Tx buffer is made.

Note: In NSS hardware mode, the NSS input of the slave device is controlled by the NSS pin, or by another GPIO pin driven by software.

17.3.1.4 Configure SPI for Half-duplex Communication

The SPI can operate in half-duplex mode with the following 2 configurations:

- 1 clock and 1 bidirectional data wire
- 1 clock and 1 data wire (receive-only or transmit-only)

1 clock and 1 bidirectional data wire (BDMODE = 1)

This mode is enabled by setting the BDMODE bit in the SPI_CTRL1 register. In this mode, the SCK pin is used as the clock; MOSI in master mode and MISO in slave mode are used for data communication. The transfer direction is selected by the BDOE bit in the SPI_CTRL1 register. When this bit is '1', the data line is output; otherwise, it is input.

1 clock and 1 unidirectional data wire (BDMODE = 0)

In this mode, SPI can either be transmit-only or receive-only.

- Transmit-only mode is similar to full-duplex mode (BDMODE = 0, RONLY = 0): Data is transmitted on the transmit pin (MOSI in master mode or MISO in slave mode), and the receive pin (MISO in master mode or MOSI in slave mode) can be used as a general-purpose I/O. In this case, the software can ignore data in the Rx buffer (If the data register is read, it does not contain any received data).
- In receive-only mode, the SPI output function can be disabled by setting the RONLY bit in the SPI_CTRL2 register. In this case, the transmit pin (MOSI in master mode or MISO in slave mode) is released, so it can be used for other purposes.

To start the communication in receive-only mode, configure the SPI as follows:

- In master mode, the communication starts immediately once the SPI is enabled. The current reception will not stop until it is finished when the SPIEN bit is cleared. There is no need to read the BSY flag in this mode. It always remains set when SPI communication is ongoing.
- In slave mode, the SPI continues to receive as long as the NSS is pulled down (or the ISS bit is cleared in NSS software mode), and at the same time the SCK has clock pulse.

17.3.1.5 Data Transmission and Reception

Rx and Tx buffers

In reception, received data are stored into an internal Rx buffer; in transmission, data are first stored into an internal Tx buffer before being transmitted.

A read access to the SPI_DT register returns the Rx buffered value; data written to the SPI_DT will be written into the Tx buffer.

Start sequence in master mode

- In full-duplex mode (BDMODE = 0 and RONLY = 0)
 - The sequence begins when data is written into the SPI_DT register (Tx buffer).
 - Data is parallelly loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then shifted out serially to the MOSI pin.
 - At the same time, the received data on the MISO pin is serially shifted into the 8-bit shift register and then parallelly loaded into the SPI_DT register (Rx buffer).
- In unidirectional receive-only mode (BDMODE = 0 and RONLY = 1)
 - The sequence begins as soon as SPIEN = 1.

- Only the receiver is activated, and the received data on the MISO pin are shifted serially into the 8-bit shift register and then are parallelly loaded into the SPI_DT register (Rx buffer).
- In bidirectional mode, when transmitting (BDMODE = 1 and BDOE = 1)
 - The sequence begins when data are written into the SPI_DT register (Tx buffer).
 - The data are then parallelly loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then are shifted out serially to the MOSI pin.
 - No data is received.
- In bidirectional mode, when receiving (BDMODE = 1 and BDOE = 0)
 - The sequence begins as soon as SPIEN = 1 and BDOE = 0.
 - The received data on the MOSI pin are serially shifted into the 8-bit shift register and then are parallelly loaded into the SPI_DT register (Rx buffer).
 - The transmitter is not activated, and no data is shifted out serially to the MOSI pin.

Start sequence in slave mode

- In full-duplex mode (BDMODE = 0 and RONLY = 0)
 - The sequence begins when the slave device receives the clock signal, and the first data bit appears on its MOSI pin. The remaining bits are loaded into the shift register one after another.
 - At the same time, the data are parallelly loaded from the Tx buffer into the 8-bit shift register during the first bit transmission, and then are shifted out serially to the MISO pin. The software must write the data to be sent into the Tx register before the SPI master device initiates the transfer.
- In unidirectional receive-only mode (BDMODE = 0 and RONLY = 1)
 - The sequence begins when the slave device receives the clock signal, and the first data bit appears on its MOSI pin. Then, the remaining bits are loaded into the shift register one after another.
 - The transmitter is not activated, and no data is shifted out serially to the MISO pin.
- In bidirectional mode, when transmitting (BDMODE = 1 and BDOE = 1)
 - The sequence begins when the slave device receives the clock signal, and the first data bit in the Tx buffer is sent to the MISO pin.
 - When the first data bit is sent to the MISO pin, the data to be sent in the Tx buffer are parallelly loaded into the 8-bit shift register, and then are shifted out serially to the MISO pin. The software must write the data to be sent into the Tx register before the SPI master device initiates the transfer.
 - No data is received.
- In bidirectional mode, when receiving (BDMODE = 1 and BDOE = 0)
 - The sequence begins when the slave device receives the clock signal, and the first data bit appears on its MISO pin.
 - The received data on the MISO pin is serially shifted into the 8-bit shift register and then are parallelly loaded into the SPI_DT register (Rx buffer).
 - The transmitter is not activated, and no data are serially shifted out to the MISO pin.

Handling data transmission and reception

The TE flag (Tx buffer empty) is set when the data is transferred from the Tx buffer to the shift register. It indicates that the internal Tx buffer is ready to be loaded with the next data. An interrupt will be generated if the TEIE bit in the SPI_CTRL2 register is set. Writing to the SPI_DT register can clear the TE bit.

Note: *The software must ensure that the TE flag is set to 1 before attempting to write to the Tx buffer. Otherwise, it overwrites the data that is previously written to the Tx buffer.*

The RNE flag (Rx buffer not empty) is set on the last sampling clock edge, when the data are transferred from the shift register to the Rx buffer. It indicates that data are ready to be read from the SPI_DT register.

An interrupt will be generated if the RNEIE bit in the SPI_CTRL2 register is set. Reading the SPI_DT register can clear the RNE bit.

For some configurations, the BSY flag can be used during the last data transfer to wait for completion of the transfer.

Full-duplex transmit and receive procedure in master or slave mode (BDMODE = 0 and RONLY = 0)

The software should follow the procedure below to transmit and receive data (See [Figure 17-5](#) and [Figure 17-6](#)):

1. Enable the SPI by setting the SPIEN bit to 1.
2. Write the first data to be transmitted into the SPI_DT register, and this clears the TE flag.
3. Wait until TE = 1, and then write the second data to be transmitted. Then, wait until RNE = 1 and read the SPI_DT register to get the first received data; at the same time, reading the SPI_DT register will clear the RNE bit. Repeat this procedure to transmit the remaining data while receiving the n - 1 data.
4. Wait until RNE = 1, and read the last received data.
5. Wait until TE = 1, and then disable the SPI after BSY = 0.

This procedure can also be implemented to handle the interrupts generated at rising edges of the corresponding RNE or TE flags.

Figure 17-5 TE/RNE/BSY Behavior during Continuous Output in Master/Full-duplex Mode (BDMODE = 0 and RONLY = 0)

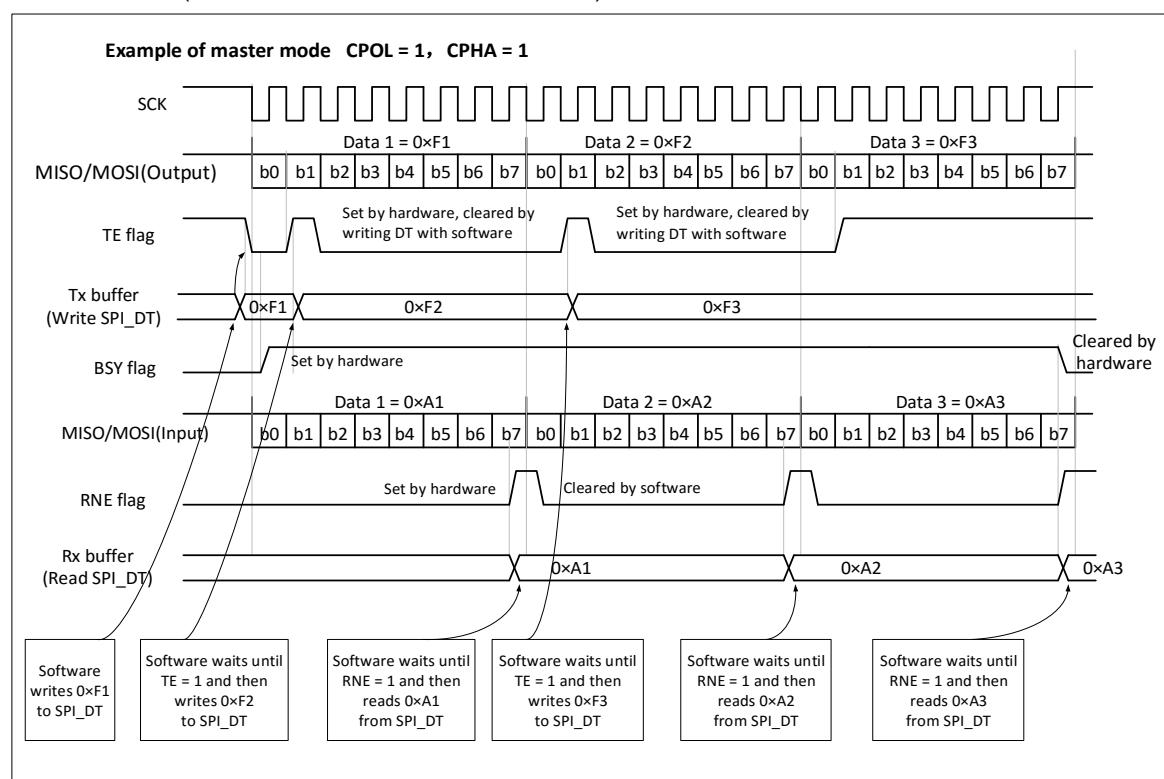
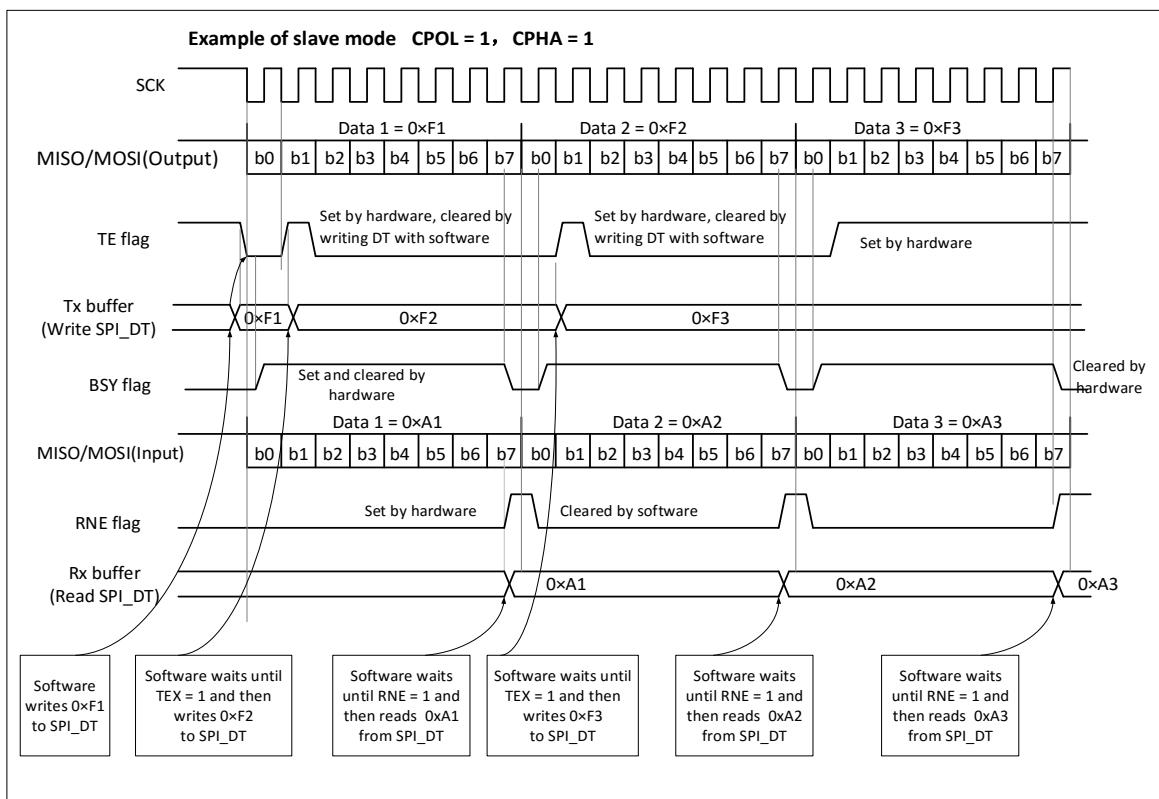


Figure 17-6 TE/RNE/BSY Behavior during Continuous Transfer in Slave/Full-duplex Mode (BDMODE = 0 and RONLY = 0)



Transmit-only procedure (BDMODE = 0 and RONLY = 0)

In this mode, the procedure can be briefly described as follows, and the BSY bit can be used to wait until the end of the transmission (See [Figure 17-7](#) and [Figure 17-8](#)):

1. Enable the SPI by setting the SPIEN bit to '1'.
2. Write the first data item to be sent into the SPI_DT register, and this clears the TE flag.
3. Wait until TE = 1, and then write the next data item to be transmitted. Repeat this step for each data item to be transmitted.
4. After writing the last data item into the SPI_DT register, wait until TE = 1; then, wait until BSY = 0, which indicates that the transmission of the last data is completed.

This procedure can also be implemented to handle the interrupts generated at rising edges of the corresponding TE flags.

- Note:**
1. During discontinuous communications, there is a 2-APB clock period delay between the write operation to the SPI_DT register and setting the BSY bit. As a consequence, in transmit-only mode, it is recommended that users wait until TE = 1, and then wait until BSY = 0 after writing the last data.
 2. After transmitting two data items in transmit-only mode, the OVR bit becomes '1' in the SPI_STS register since the received data will not be read (Note: Software can ignore this OVR flag bit).

Figure 17-7 TE/BSY Behavior during Continuous Transfer in Master Transmit-only Mode (BDMODE = 0 and RONLY = 0)

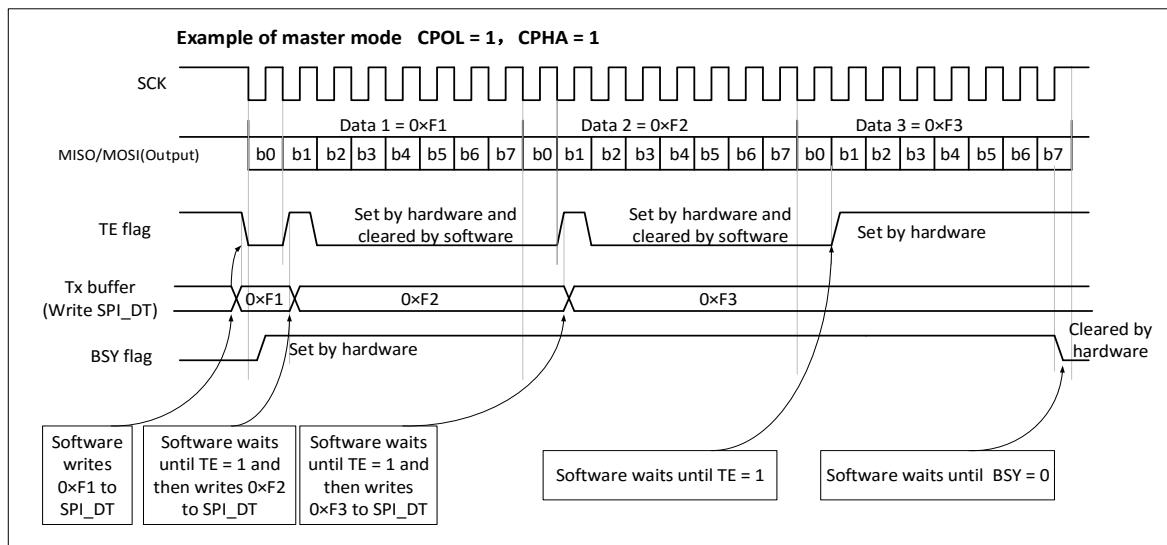
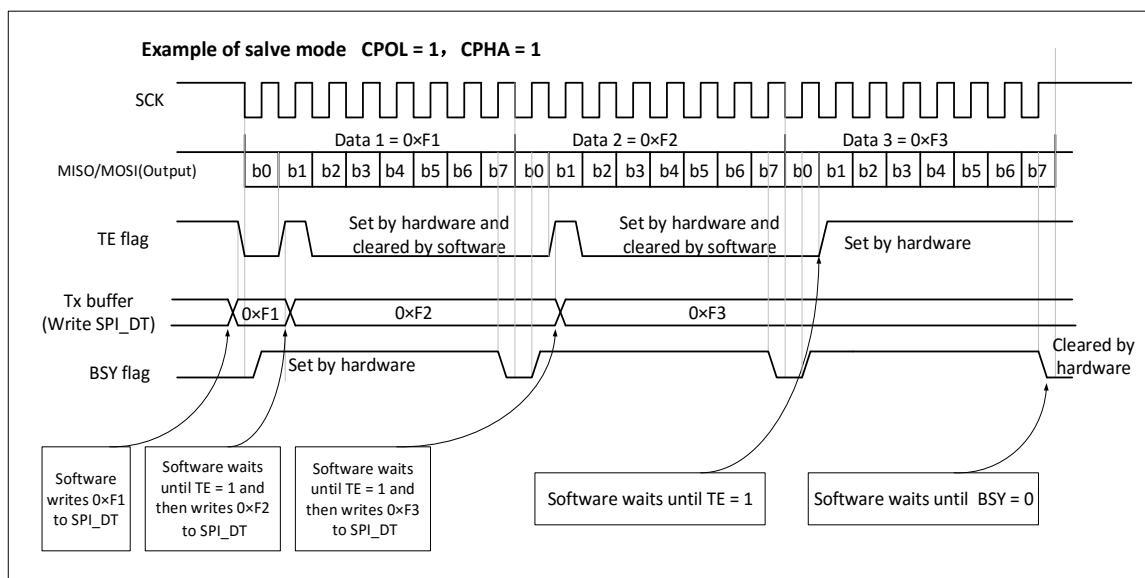


Figure 17-8 TE/BSY Behavior during Continuous Transfer in Slave Transmit-only Mode (BDMODE = 0 and RONLY = 0)



Bidirectional transmit procedure (BDMODE = 1 and BDOE = 1)

In this mode, the procedure is similar to the procedure in transmit-only mode, except that the BDMODE and BDOE bits have to be set at the same time in the SPI_CTRL2 register before enabling the SPI.

Unidirectional receive-only procedure (BDMODE = 0 and RONLY = 1)

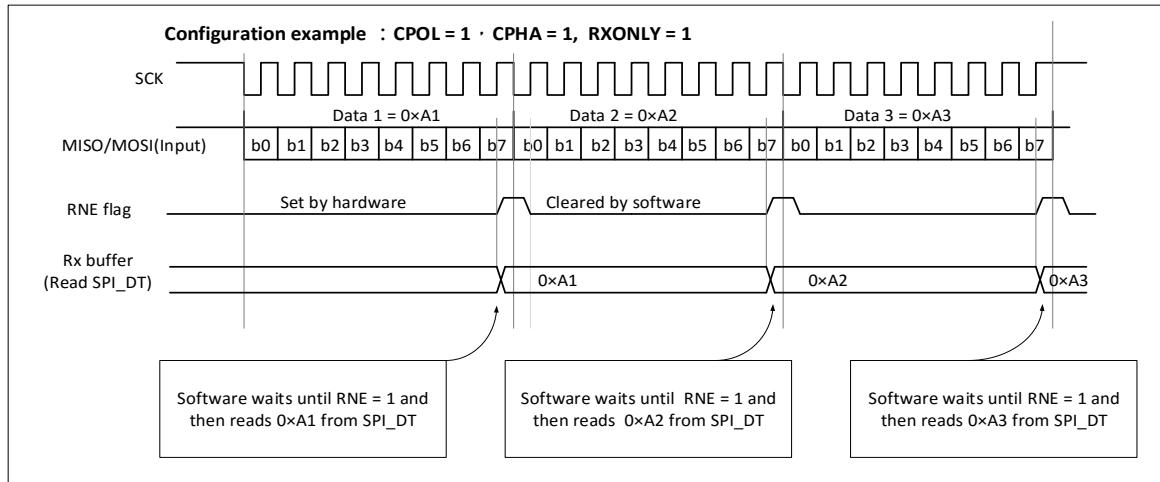
In this mode, the procedure can be briefly described as follows, (See [Figure 17-9](#)):

1. Set RONLY = 1 in the SPI_CTRL2 register.
2. Set SPIEN = 1 to enable the SPI:
 - a) In master mode, this immediately generates the SCK clock, and data will be continuously received until the SPI is disabled (SPIEN = 0).
 - b) In slave mode, serial data will be received when the SPI master device drives the NSS signal low and generates the SCK clock.
3. Wait until RNE = 1, and then read the SPI_DT register to obtain the received data (this clears the RNE bit). Repeat this operation for each data item to be received.

This procedure can also be implemented to handle the interrupts generated at rising edges of the corresponding RNE flags.

Note: If the SPI is disabled after the last transfer, please follow the recommended procedure in [Section 17.3.1.9](#).

Figure 17-9 RNE Behavior during Continuous Transfer in Receive-only Mode (BDMODE = 0 and RONLY = 1)



Bidirectional receive procedure (BDMODE = 1 and BDOE = 0)

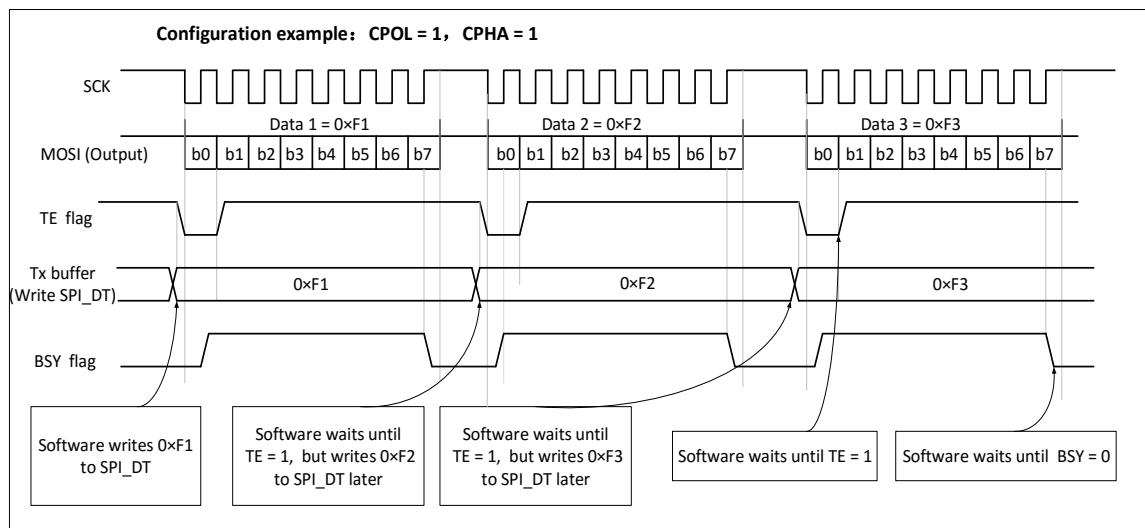
In this mode, the procedure is similar to the receive-only mode procedure, except that the BDMODE bit has to be set '1', and the BDOE bit is cleared in the SPI_CTRL2 register before enabling the SPI.

Continuous and discontinuous transfers

When transmitting data in master mode, if the software is fast enough to detect each rising edge of TE (or TE interrupts) and to immediately write to the SPI_DT register before the end of ongoing data transfer, then continuous communication can be achieved. In this case, continuity of the SPI clock between each data transfer is maintained, and the BSY bit will not be cleared.

On the contrary, if the software is not fast enough, then it leads to discontinuities in the communication. In this case, the BSY bit is cleared between each data transmission (See Figure 17-10). In master receive-only mode (RONLY = 1), the communication is always continuous, and the BSY flag is always '1'. In slave mode, the continuity of the communication is determined by the SPI master device. In any case, even if the communication is continuous, the BSY flag goes low between each transfer for at least one SPI clock cycle (See [Figure 17-8](#)).

Figure 17-10 TE/BSY Behavior during Discontinuous Transfer (BDMODE = 0 and RONLY = 0)



17.3.1.6 CRC Calculation

CRC calculation is implemented for full-duplex communication reliability. Separate CRC calculators are used for the transmitted data and received data. The CRC is calculated through a programmable polynomial computation on each bit. It is performed on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CTRL1 register.

Note: *This SPI offers two kinds of CRC calculation methods which depend on the data frame format selected for the transmission and/or reception. 8-bit data frame adopts CRC8; 16-bit data frame adopts CRC16.*

CRC calculation is enabled by setting the CCE bit in the SPI_CTRL1 register, and this action also resets the CRC registers (SPI_RCRC and SPI_TCRC). After the CTN bit in SPI_CTRL1 is set, the content in SPI_TCRC will be transmitted after the current byte is sent.

When transferring the content of SPI_TCRC, if the data received in the shift register does not match with the content of SPI_RCRC, then the CERR bit in the SPI_STS register is set.

If data is still present in the Tx buffer, the CRC value is transmitted only after the end of data byte transmission. During CRC transmission, the CRC calculator is switched off, and the register value remains unchanged.

Note: *Please refer to the product instruction to check if there is CRC calculation (This function is not available on every model).*

SPI communication with CRC is enabled through the following procedure:

- Program the CPOL, CPHA, LSBEN, MCLKP, SWNSSEN, ISS, and MSTEN values.
- Program the polynomial in the SPI_CPOLY register.
- Enable CRC calculation by setting the CCE bit in the SPI_CTRL1 register. This also clears the SPI_RCRC and SPI_TCRC registers.
- Enable the SPI by setting the SPIEN bit in the SPI_CTRL1 register.
- Start the communication and keep the communication until the last byte or half-word.
- When writing the last byte or half-word to the Tx buffer, set the CTN bit in the SPI_CTRL1 register to indicate that the CRC value will be transmitted after the hardware transmission of the last byte. During CRC value transmission, CRC calculation is stopped.
- After the transfer of the last byte or half-word, the SPI sends the CRC value, and the CTN bit is cleared. Likewise, the received CRC is compared to the SPI_RCRC value. If the values do not match, the CERR flag in SPI_STS is set, and an interrupt is generated when the ERRIE bit in the SPI_CTRL2 register is set.

Note: 1. *When the SPI is in slave mode, please note that CRC calculation is only enabled after the clock becomes stable. Otherwise, the CRC calculation may be wrong. In fact, once the CCE bit is set, the CRC calculation is performed if input clock is present on the SCK pin, no matter what the SPIEN bit status is.*

2. *With high SPI clock frequencies, be careful when transmitting the CRC. During CRC transmission, make the time to use CPU as short as possible; it is forbidden to call software functions in the CRC transmission sequence to avoid errors in the last data and CRC reception. The CTN bit must be set before the end of the last data transmission/reception.*

3. *For high SPI clock frequencies, it is advised to use the DMA mode to avoid the degradation of the SPI speed performance, since CPU accesses would affect the SPI bandwidth.*

4. *When AT32F403 is configured as slave mode, and NSS hardware mode is used, the NSS pin should remain low during data transfer and CRC transfer.*

When the SPI is configured in slave mode with the CRC function enabled, CRC calculation is performed even if a high level is applied on the NSS pin. (Note: When the NSS signal is high, if clock pulse is present on the SCK pin, then CRC calculation is kept being performed.) For example, this may happen in a multislave environment where the communication master addresses several slaves alternately. (Note: In this case, CRC misoperation should be avoided.)

After each CRC value transmission or reception, the CRC value will be cleared.

17.3.1.7 Status Flag

Three status flags are provided for the application to completely monitor the state of the SPI bus.

Tx buffer empty flag (TE)

When this flag is set, it indicates that the Tx buffer is empty, and the next data to be transmitted can be loaded into the buffer. The TE flag is cleared when writing to the SPI_DT register.

Rx buffer not empty (RNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is cleared when the SPI data register is read.

Busy flag

The BSY flag is set and cleared by hardware (writing to this flag has no effect). The BSY flag indicates the state of the SPI communication layer.

When BSY is set, it indicates that the SPI is busy communicating. There is one exception: In bidirectional master receive mode (MSTEN = 1, BDM = 1, and BDOE = 0), the BSY flag is kept low during reception.

The BSY flag is useful to detect the end of a transfer if the software wants to disable the SPI and enter Halt mode (or disable the peripheral clock). This avoids corrupting the last transfer. Therefore, the procedure described below must be strictly followed.

The BSY flag is also useful to avoid write collisions in a multimaster system.

The BSY flag is set when a transfer starts, except for bidirectional receive mode of master mode (MSTEN = 1, BDM = 1, and BDOE = 0).

This flag is cleared under the following conditions:

- When a transfer is finished. (Except for the continuous communication in master mode)
- When the SPI is disabled.
- When a master mode fault occurs (MODF = 1).

If the communication is not continuous, the BSY flag is low between each data item transfer.

When the communication is continuous:

- In master mode: The BSY flag is kept high during the whole transfer procedure.
- In slave mode: The BSY flag goes low for one SPI clock cycle between each data transfer.

Note: *Do not use the BSY flag to handle every data transmission or reception. It is better to use the TE and RNE flags instead.*

17.3.1.8 Disabling SPI

When communication is completed, disabling the SPI peripheral can stop the communication. This is done by clearing the SPIEN bit. For some configurations, disabling the SPI and entering the Halt mode while a transfer is ongoing may corrupt the current transfer, and/or the BSY flag might become unreliable.

To avoid this situation, it is recommended that users follow the procedure below when disabling the SPI:

In master or slave full-duplex mode (BDMODE = 0, RONLY = 0)

1. Wait until RNE = 1 and receive the last data
2. Wait until TE = 1
3. Wait until BSY = 0
4. Disable the SPI (SPIEN = 0), and finally enter the Halt mode (or disable the peripheral clock).

In master or slave unidirectional transmit-only mode (BDMODE = 0, RONLY = 0) or bidirectional transmit mode (BDMODE = 1, BDOE = 1)

After the last data is written into the SPI_DT register:

1. Wait until TE = 1
2. Wait until BSY = 0
3. Disable the SPI (SPIEN = 0), and finally enter the Halt mode (or disable the peripheral clock).

In master unidirectional receive-only mode (MSTEN = 1, BDMODE = 0, RONLY = 1) or bidirectional receive mode (MSTEN = 1, BDMODE = 1, BDOE = 0)

This case should be managed in a particular way to ensure that the SPI does not initiate a new transfer:

1. Wait for the second to last ($n - 1$) occurrence of RNE = 1
2. Wait for one SPI clock cycle before disabling the SPI (SPIEN = 0) (using a software delay)
3. Wait for the last RNE = 1 before entering the Halt mode (or disabling the peripheral clock)

Note: In master unidirectional receive-only mode (MSTEN = 1, BDM = 1, BDOE = 0), the BSY flag is always kept low during the transfers.

In slave receive-only mode (MSTEN = 0, BDMODE = 0, RONLY = 1) or bidirectional receive mode (MSTEN = 0, BDMODE = 1, BDOE = 0)

1. Users can disable the SPI (SPIEN = 0) at any time, and the SPI will be disabled after the current transfer is finished.
2. If users want to enter the Halt mode, users must first wait until BSY = 0 before entering the Halt mode (or disabling the peripheral clock).

17.3.1.9 SPI Communication Using DMA

To reach the maximum of communication speed, the SPI transmit buffer needs to be filled with data in time; likewise, the data received on the Rx buffer should be read in time to avoid overrun. To facilitate effective transfers, the SPI features a DMA capability which adopts a simple request/acknowledge protocol.

A DMA access is requested from the DMA when the corresponding enable bit in the SPI_CTRL2 register is set. Separate requests must be issued to the Tx and Rx buffers.

- In transmission, a DMA request is issued each time when TE is set to '1'. The DMA controller then writes data to the SPI_DT register, which clears the TE flag.
- In reception, a DMA request is issued each time when RNE is set to '1'. The DMA controller then reads the SPI_DT register, which clears the RNE flag.

When only the SPI is used to transmit data, only the SPI Tx DMA channel needs to be enabled. In this case, the OVR flag is set because the data received is not read (Note: Software can ignore this flag). When only the SPI is used to receive data, only the SPI Rx DMA channel needs to be enabled.

In transmission mode, when the DMA transfers all the data to be transmitted (The TCIF flag becomes '1' in the DMA_ISTS register), the BSY flag can be monitored to ensure that the SPI communication is completed. This can avoid corrupting the last transmission before disabling the SPI or entering the Stop mode. Therefore, the software must first wait until TE = 1 and then BSY = 0.

Note: During discontinuous communications, there is a 2-APB clock period delay between the write operation to SPI_DT and setting the BSY bit. As a consequence, it is necessary to wait first until TE = 1, and then until BSY = 0 after writing the last data.

Figure 17-11 Transmission using DMA

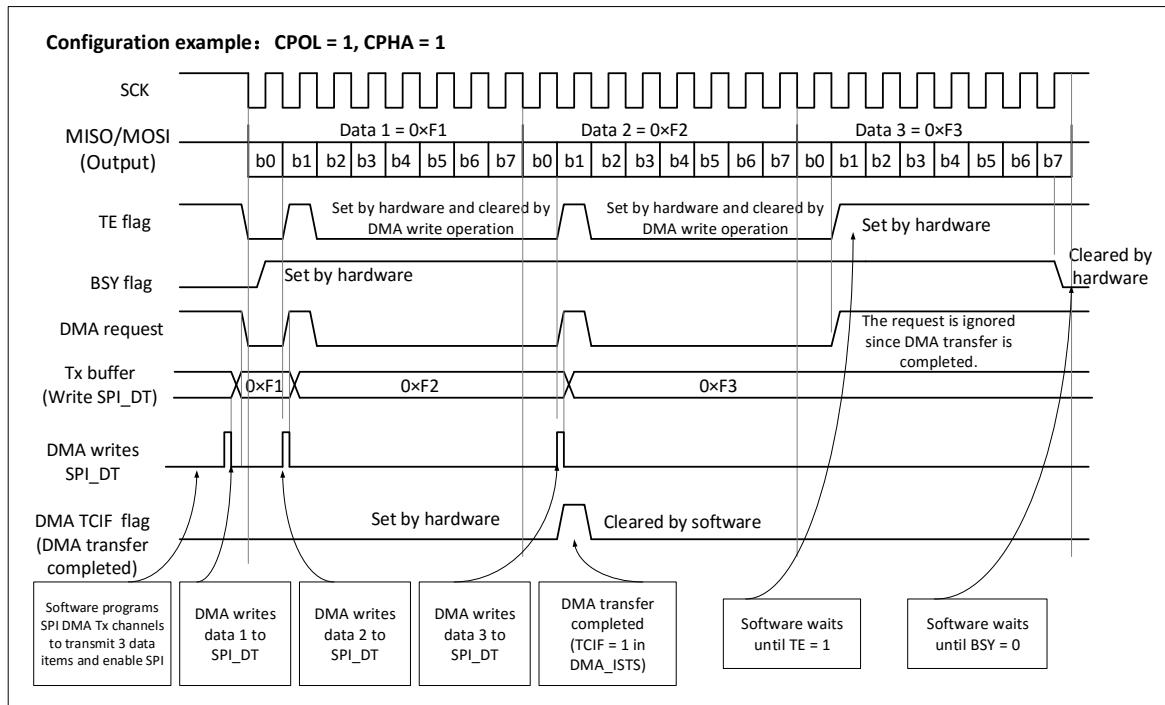
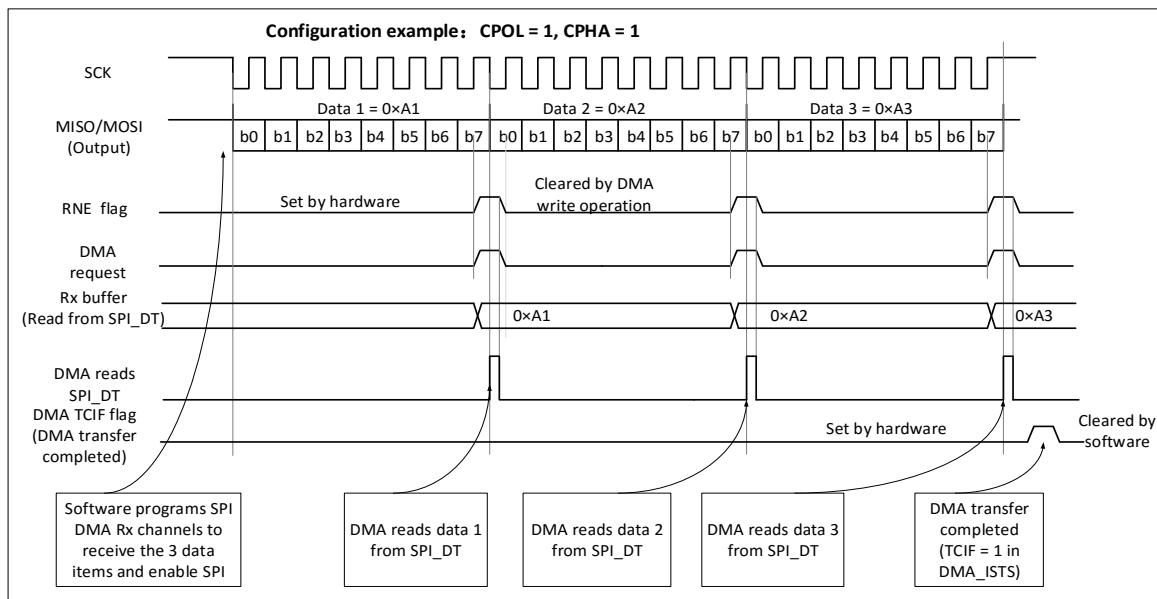


Figure 17-12 Reception using DMA



DMA capability with CRC

When the SPI is enabled with CRC check and DMA mode, the transmission and reception of the CRC are automatically completed at the end of communication.

At the end of data and CRC transfers, the CERR flag in SPI_STS is '1' if corruption occurs during the transfer.

17.3.1.10 Error Flag

Master mode fault (MODF)

Master mode fault occurs only when: In NSS pin hardware mode, the master device has its NSS pin pulled low; or in NSS pin software mode, the ISS bit is set as '0'; this automatically sets the MODF bit. Master mode fault affects the SPI peripheral in the following ways:

- The MODF bit is set, and an SPI interrupt is generated if the ERRIE bit is set.
- The SPIEN bit is cleared. This blocks all outputs from the device and disables the SPI interface.
- The MSTEN bit is cleared, thus forcing the device into slave mode.

Use the following sequence to clear the MODF bit:

1. Perform a read or write access to the SPI_STS register when the MODF bit is set.
2. Then write to the SPI_CTRL1 register.

To avoid multiple slave conflicts in a system with several MCUs, the NSS pin of the master device must be pulled high before clearing the MODF bit. The SPIEN and MSTEN bits can be restored to their original states after the clearing sequence.

For security purposes, hardware does not allow the setting of the SPIEN and MSTEN bits when the MODF bit is set.

In normal configuration, the MODF bit in a slave device cannot be set. However, in a multimaster configuration, the device can be in slave mode with this MODF bit set. In this case, the MODF bit indicates that there might be a multimaster conflict. An interrupt routine can be used to recover from fault state by resetting or returning to default state.

Overrun error

An overrun condition occurs when the master device sends data bytes, but the slave device does not clear the RNE bit resulting from the previous data byte. When an overrun condition occurs:

- The OVR bit is set '1'; an interrupt is generated if the ERRIE bit is set.

In this case, the contents in Rx buffer will not be updated with the newly received data from the master device. A read from the SPI_DT register returns the previous unread byte. All other subsequently transmitted bytes are lost.

Reading the SPI_DT register and SPI_STS register one after another can clear the OVR bit.

CRC error

When the CCE bit is set in the SPI_CTRL register, the CRC error flag is used to verify the validity of the value received. If the value received in the shift register (the SPI_TCRC value sent by the transmitter) does not match with the value of the receiver (the SPI_RCRC register), the CRCEN bit in the SPI_STS register will be set.

17.3.1.11 SPI Interrupt

Table 17-1 SPI Interrupt Request

Interrupt Event	Event Flag	Enable Control Bit
Tx buffer empty flag	TE	TEIE
Rx buffer not empty flag	RNE	RNEIE
Master mode fault event	MODF	
Overrun error	OVR	ERRIE
CRC error flag	CERR	

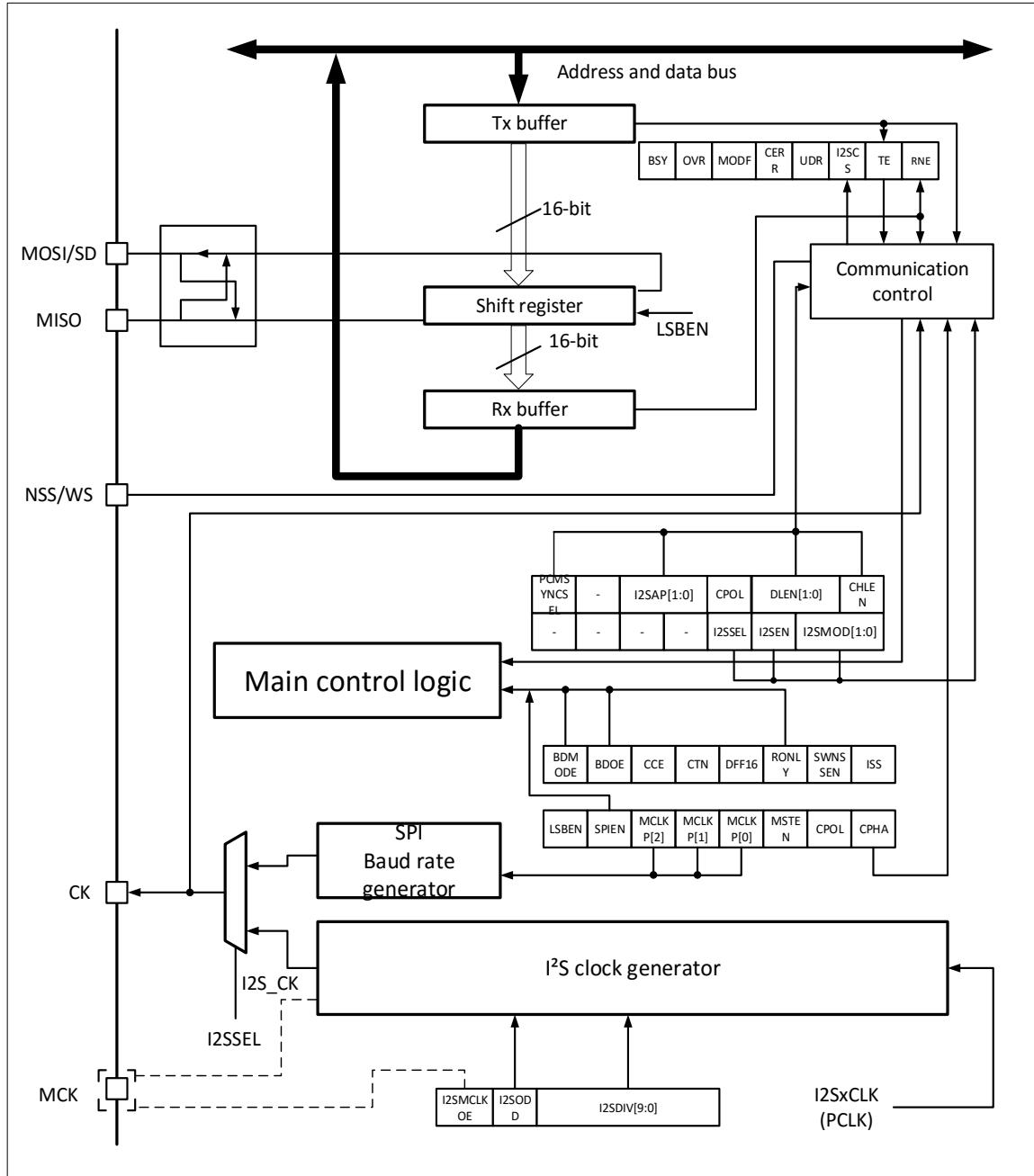
17.3.2 I²S Function Overview

All AT32F403 products support I²S audio protocol.

17.3.2.1 I²S Function Overview

Figure 17-13 is the block diagram of I²S:

Figure 17-13 I²S Block Diagram



Setting the I2SEL bit in the SPI_I2SCTRL register can enable I²S function. In this case, the SPI functions as an audio I²S interface. Generally speaking, this I²S interface uses the same pins, flags, and interrupts as the SPI.

The I²S shares three common pins with the SPI:

- SD: Serial data (mapped on the MOSI pin) for transmitting or receiving the data of two time-multiplexed channels
 - WS: Word select (mapped on the NSS pin) as data control signal output in master mode; input in slave mode
 - CK: Serial clock (mapped on the SCK pin) as clock signal output in master mode;

input in slave mode

- For some external audio devices which require master clock, an additional pin could be used to output clock.
- MCLK: Master clock (mapped independently) is used to output additional clock signal when I²S is configured as master mode and the I2SMCLKOE bit in the SPI_I2SCLKP register is set. The frequency of output clock signal is set to 256×Fs by default, where Fs is the audio signal sampling frequency.

When set in master mode, I²S uses its own clock generator to produce the communication clock signal. This clock generator is also the source of the master clock output. Two additional registers are available in I²S mode; one is SPI_I2SCLKP, which is used to configure the clock generator, and the other is generic I²S configuration register, SPI_I2SCTRL (which configures audio standard, slave/master mode, data format, packet frame, and clock polarity, etc.).

The SPI_CTRL1 register and all CRC registers are not used in the I²S mode. Likewise, the NSSOE bit in the SPI_CTRL2 register and the MODF and CERR bits in the SPI_STS are not used. I²S uses the same register, SPI_DR, as SPI for data transfer in 16-bit wide mode.

17.3.2.2 Supported Audio Protocol

The three-line bus supports audio data time-multiplexed on two channels: the right channel and the left channel. However, there is only one 16-bit register for the transmission and the reception. Hence, the software should write the corresponding value of the current transmission channel into the data register; likewise, identify the corresponding channel by checking the I2SCS bit in the SPI_STS register when reading data from the data register. The left channel is always sent first, followed by the right channel (The I2SCS bit has no meaning for the PCM protocol).

Four data and packet frames are available. Data can be sent with the following four formats:

- 16-bit data packed in 16-bit frame
- 16-bit data packed in 32-bit frame
- 24-bit data packed in 32-bit frame
- 32-bit data packed in 32-bit frame

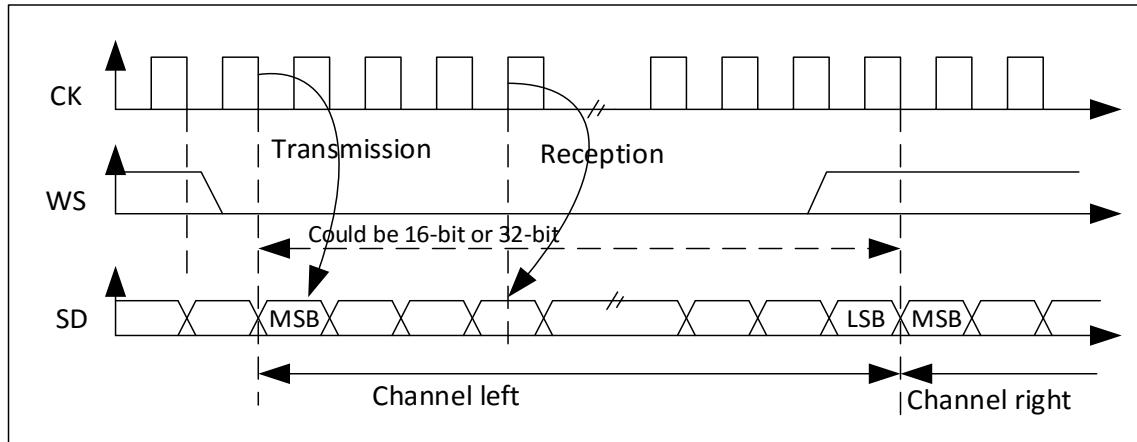
When using 16-bit data extended on 32-bit packet, the first 16 bits (MSB) are the significant bits, and the subsequent 16-bit (LSB) is forced to 0 without software action or DMA request (only one read/write operation is needed).

The 24-bit and 32-bit data frames require two CPU read or write operations to/from the SPI_DT register or two DMA transfers if the DMA is used. For 24-bit data frame, when it is extended to 32-bit, the 8 least significant bits are set '0' by hardware. For all data formats and communication standards, the most significant bit (MSB) is always sent first.

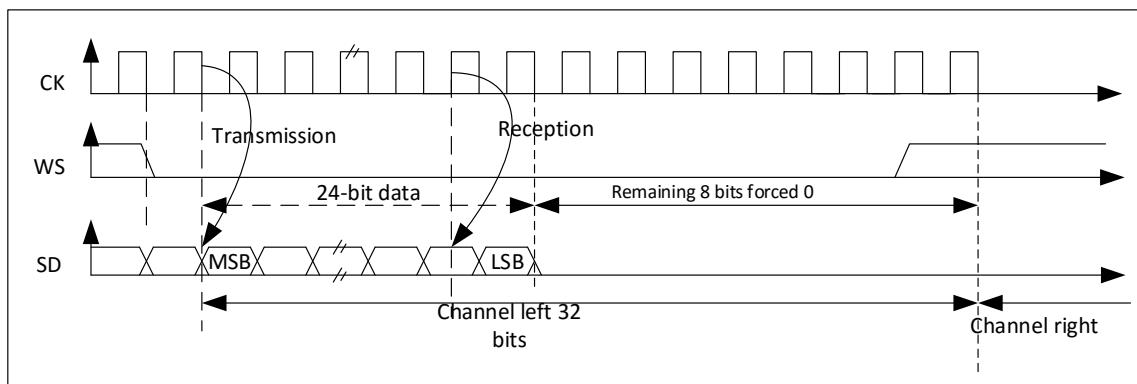
The I²S interface supports four audio standards, which are configurable with the I2SAP[1:0] and PCMSYNCSEL bits in the SPI_I2SCTRL register.

I²S Philips standard

In this standard, the WS pin is used to indicate the channel of which data is being transmitted. It is activated one clock cycle before the first bit (MSB) is sent.

Figure 17-14 I²S Philips Protocol Waveforms (16bit/32-bit full accuracy, CPOL = 0)

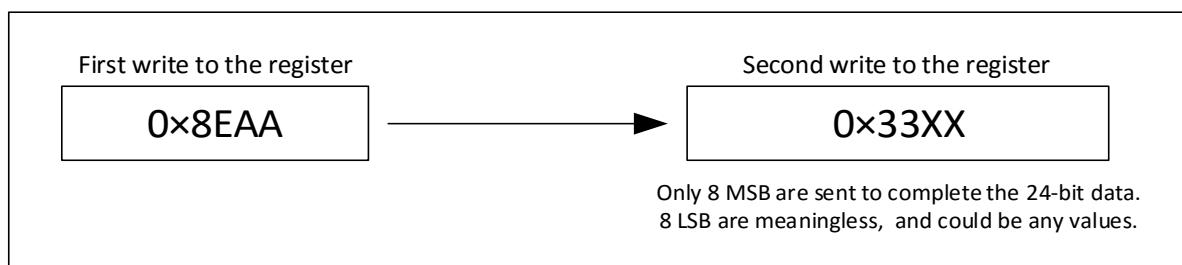
Data are transmitted on the falling edge of CK by the transmitter, and are read on the rising edge by the receiver. The WS signal is also latched on the falling edge of CK.

Figure 17-15 I²S Philips Protocol Standard Waveforms (24-bit frame, CPOL = 0)

This mode needs two write or read operations to/from the SPI_DT register.

- In transmission mode: If 0x8EAA33 has to be sent (24-bit):

Figure 17-16 Transmitting 0x8EAA33



- In reception mode: If 0x8EAA33 has to be received:

Figure 17-17 Receiving 0x8EAA33

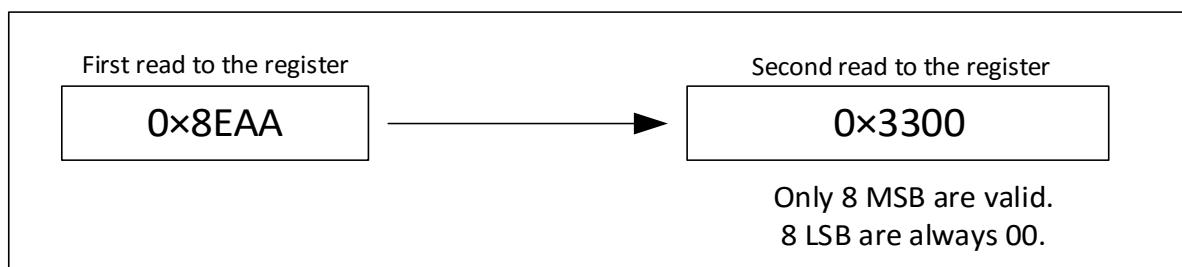
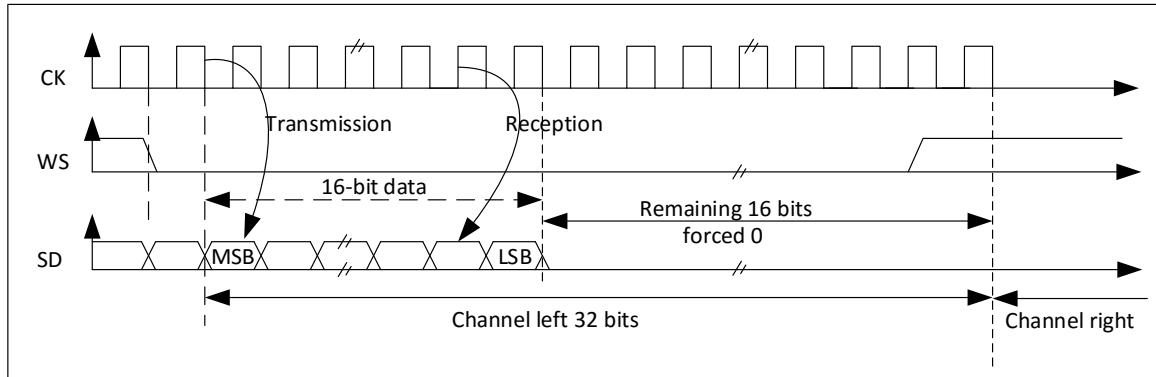
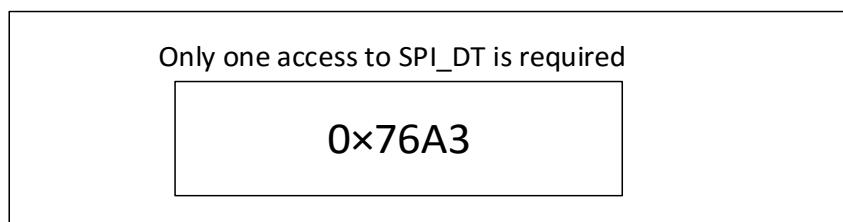


Figure 17-18 I²S Philips Protocol Standard Waveforms (16-bit extended to 32-bit frame, CPOL = 0)

When 16-bit data frame extended to 32-bit channel frame is selected at the I²S configuration phase, only one access to SPI_DT is required. The 16 remaining bits are forced by hardware to 0x0000 to extend the data to 32-bit format.

If data to be transmitted or the received data are 0x76A3 (0x76A30000 if extended to 32-bit), the operation shown in Figure 17-19 is required.

Figure 17-19 Example



During transmission, MSB should be written to SPI_DT; the TE flag bit is set, indicating that new data can be written, and its interrupt, if allowed, is generated. The transmission is done by hardware; TE is set, and the corresponding interrupt is generated even if 16-bit 0x0000 is not yet sent.

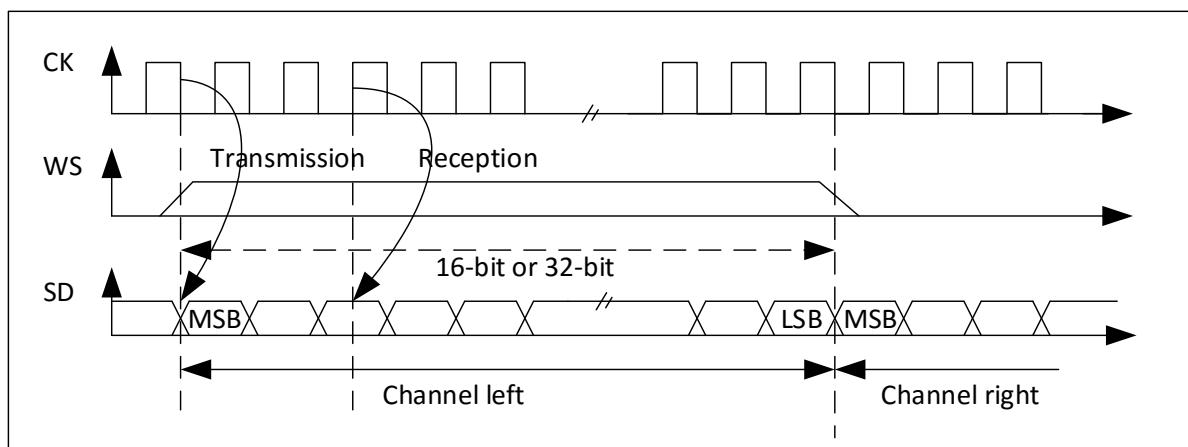
For reception, the RNE flag is set, and its interrupt, if allowed, is generated each time when the first 16 half-word (MSB) is received.

In this way, more time is provided between two write and read operations, which prevents underrun or overrun conditions.

MSB-justified standard

In this standard, the WS signal and the first data bit, which is the MSB bit, are generated at the same time.

Figure 17-20 MSB-justified 16-bit or 32-bit Full-accuracy, CPOL = 0



Data are changed on the falling edge of clock signal for transmitter and are read on the rising edge for the receiver.

Figure 17-21 MSB-justified 24-bit Data, CPOL = 0

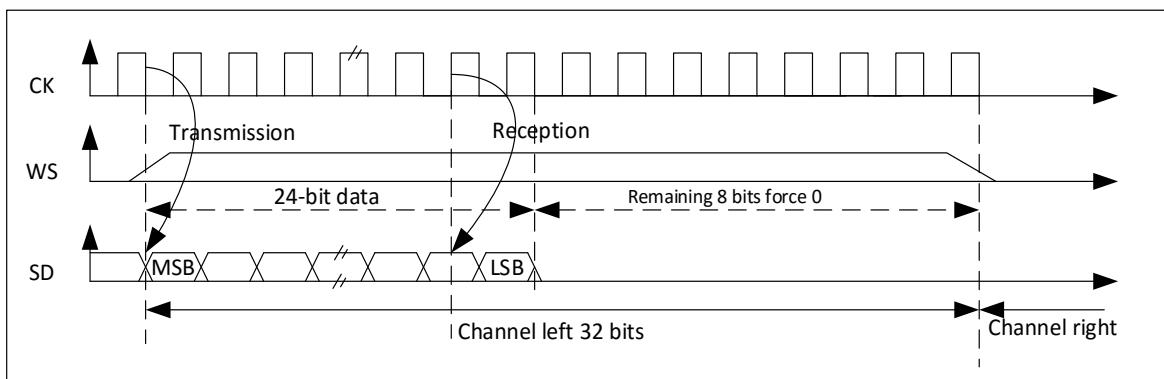
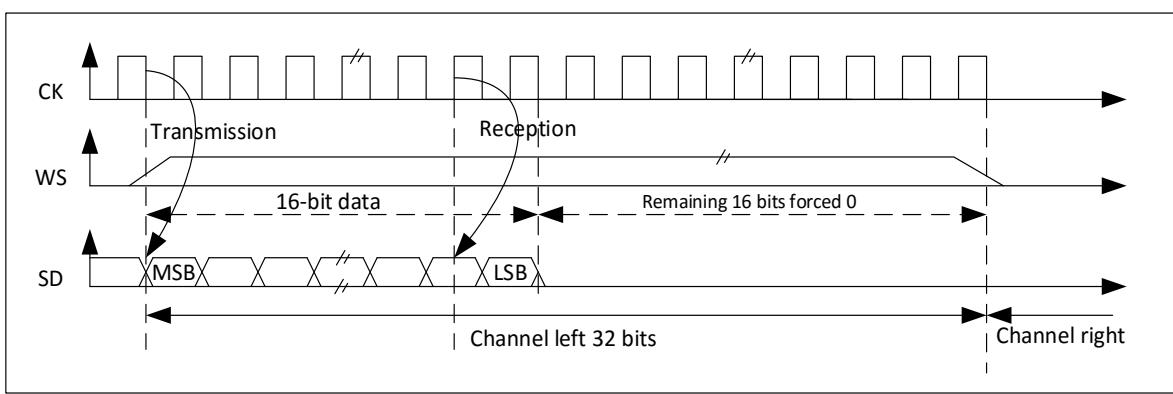


Figure 17-22 MSB-justified 16-bit Data Extended to 32-bit Frame, CPOL = 0



LSB-justified standard

This standard is similar to the MSB-justified standard (no difference between the 16-bit and 32-bit full-accuracy frame formats).

Figure 17-23 LSB-justified 16-bit or 32-bit Full-accuracy, CPOL = 0

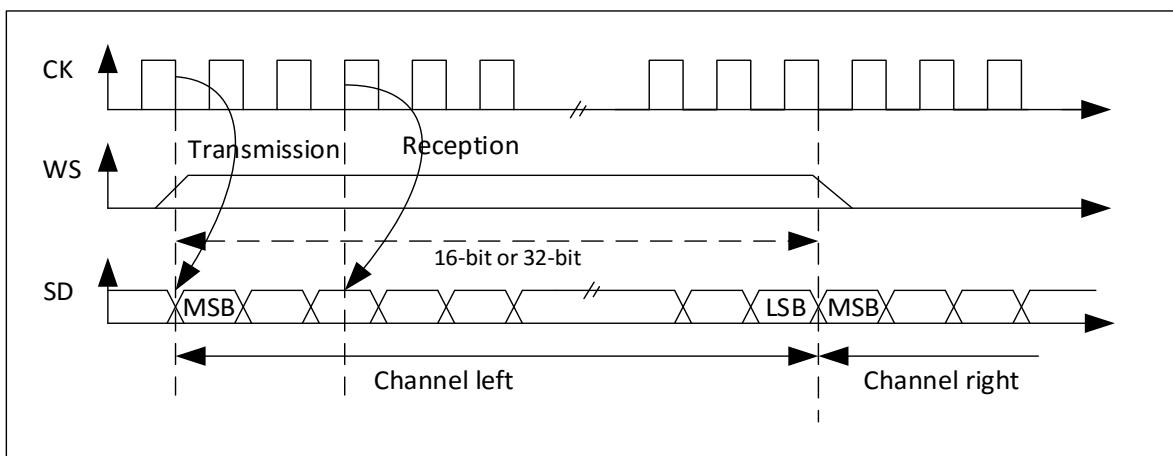
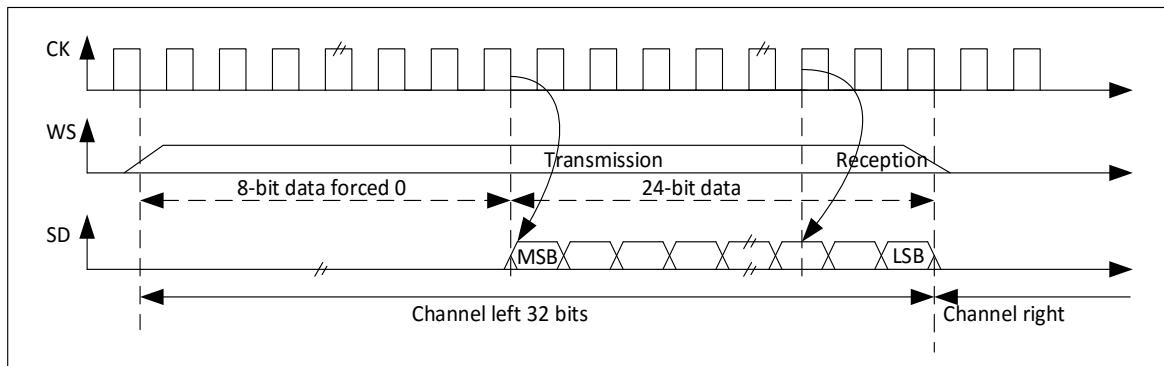


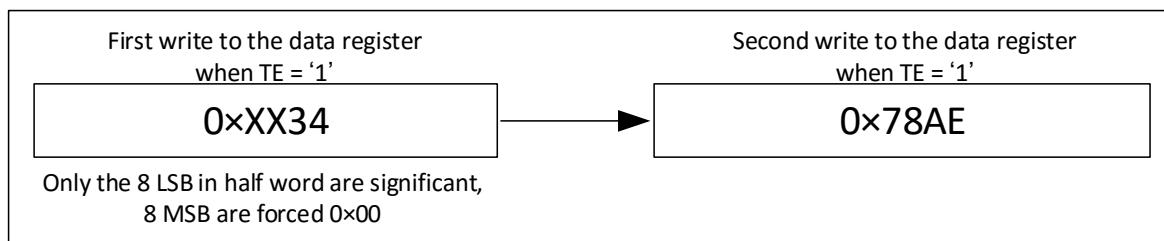
Figure 17-24 LSB-justified 24-bit Data, CPOL = 0



- In transmission mode

If data 0x3478AE is to be transmitted, two write operations to the SPI_DT register are required through software or by DMA. The operations are shown in Figure 17-25.

Figure 17-25 Operations Required to Transmit 0x3478AE



- In reception mode

If data 0x3478AE is to be received, two read operations from SPI_DT are required on two consecutive RNE events, respectively.

Figure 17-26 Operations Required to Receive 0x3478AE

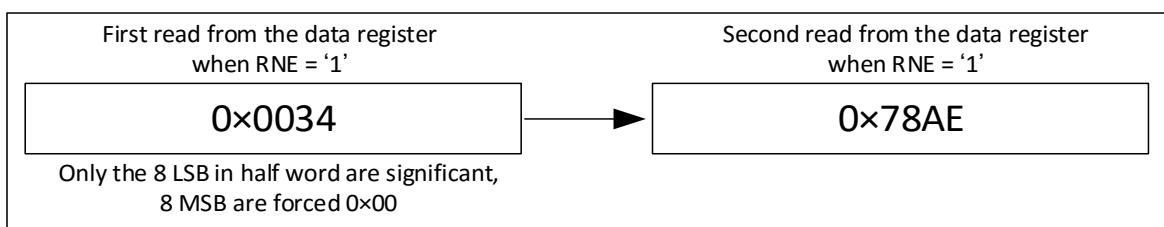
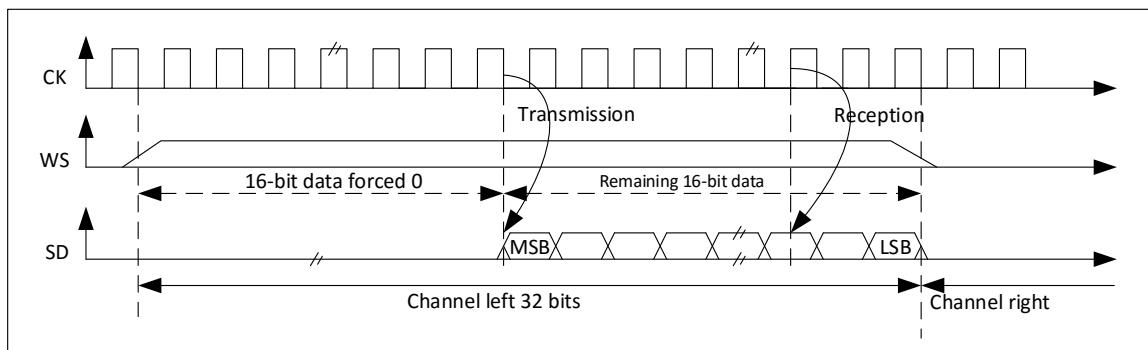


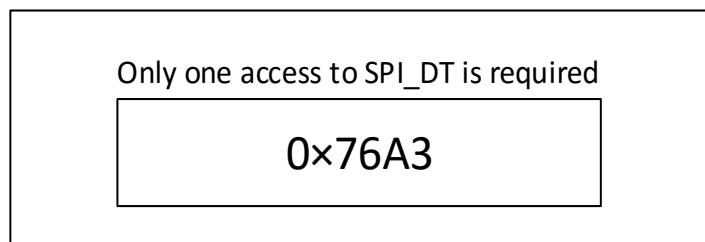
Figure 17-27 LSB-justified 16-bit Data Extended to 32-bit Frame, CPOL = 0



When 16-bit data frame extended to 32-bit channel frame is selected at the I²S configuration phase, only one access to SPI_DT is required. The first half-word (16-bit MSB) are forced to 0x0000 by hardware to extend the data to 32-bit format.

If the data to be transmitted or the received data are 0x76A3 (0x000076A3 if extended to 32-bit), the operation shown in Figure 17-28 is required.

Figure 17-28 Example

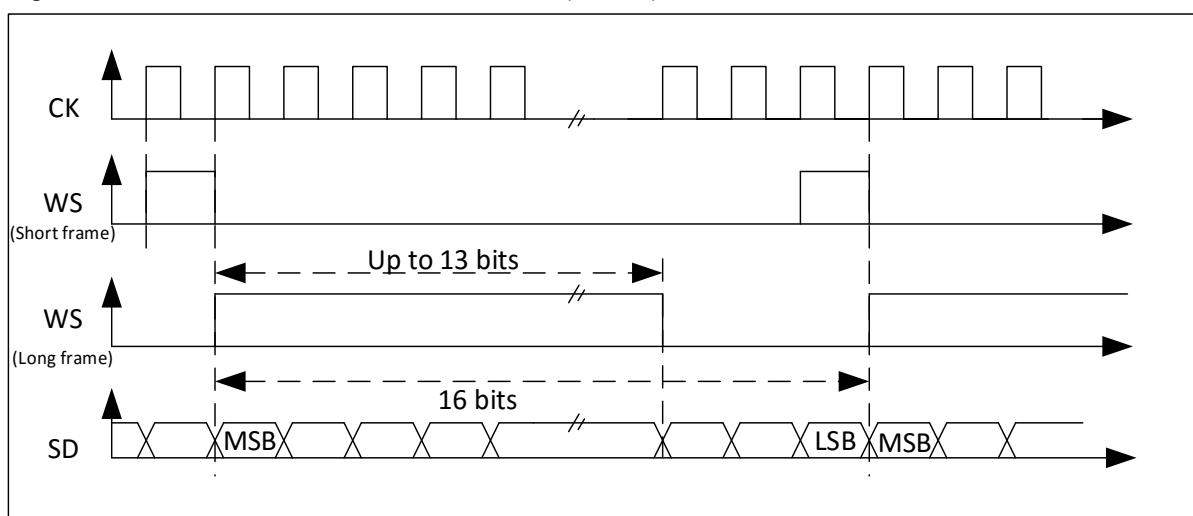


During transmission, if TE is '1', users have to write the data to be transmitted (0x76A3 in this case). The 0x0000 field used for extension on 32-bit is transmitted first. The next TE event occurs as soon as valid data begins to be sent from the SD pin. In reception mode, RNE is asserted as soon as the significant half-word is received (instead of the 0x0000 field). In this way, more time is provided between two write or read operations to prevent underrun or overrun conditions.

PCM standard

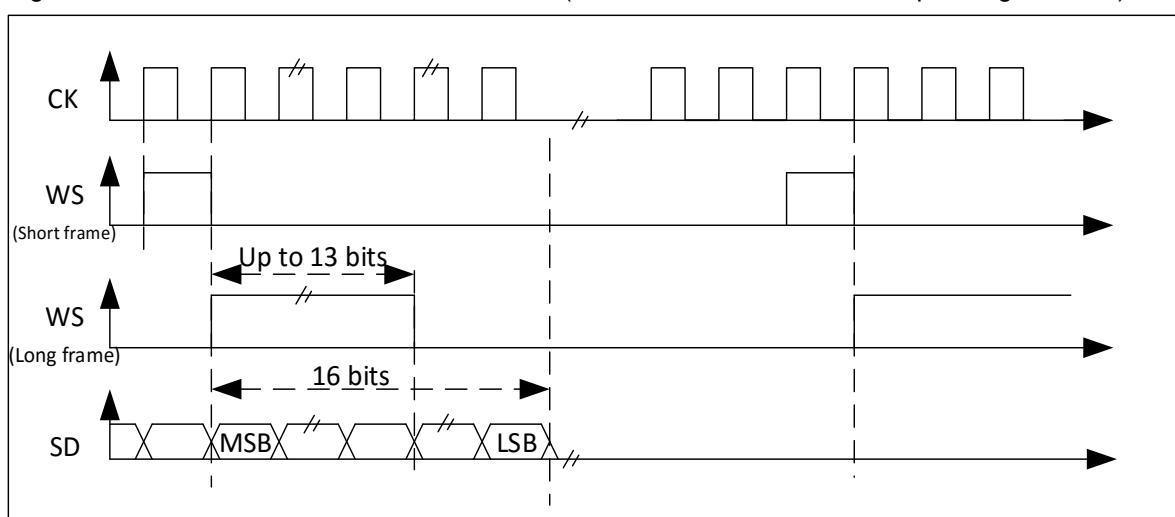
In PCM standard, there is no information on channel selection. Two frames, short and long frame, are available for PCM standard, and they can be configured with the PCMSYNCSEL bit in the SPI_I2SCTRL register.

Figure 17-29 PCM Standard Waveform (16-bit)



For long frame in slave mode, the WS signal assertion time is fixed at 13 bits for synchronization. As for short frame, the WS signal assertion time is only 1 bit for synchronization.

Figure 17-30 PCM Standard Waveform (16-bit extended to 32-bit package frame)



Note: For both modes (master and slave) and for both synchronizations (short and long), the number of bits between two consecutive pieces of data and two synchronization signals needs to be specified by the DLEN bit and the CHLEN bit in the SPI_I2SCTRL register (even in slave mode).

17.3.2.3 Clock Generator

The I²S bitrate determines the dataflow on the I²S data line and the I²S clock signal frequency.

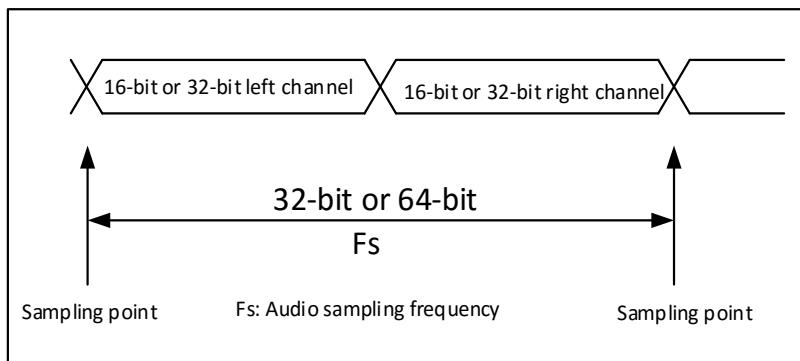
I²S bitrate = number of bits per channel × number of channels × sampling audio frequency

For a 16-bit audio signal with left and right channels, the I²S bitrate is calculated as follows:

$$\text{I}^2\text{S bitrate} = 16 \times 2 \times F_s$$

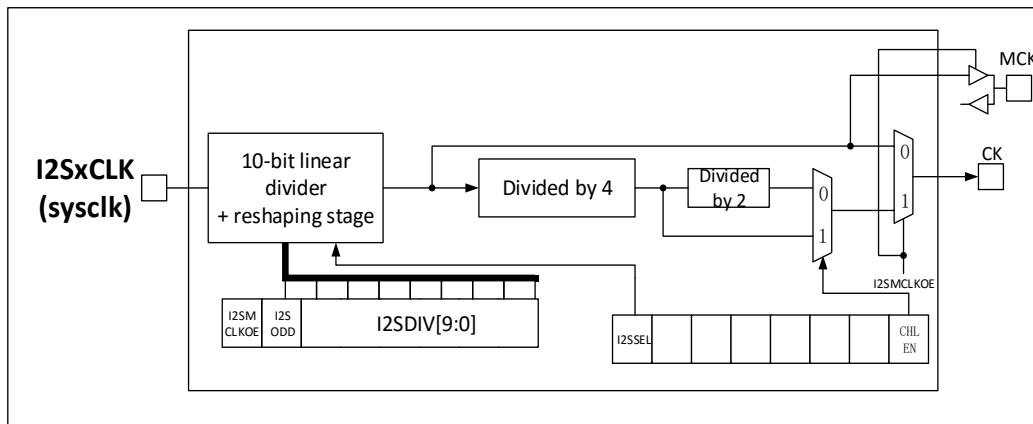
$$\text{If the packet length is 32-bit wide, then: } \text{I}^2\text{S bitrate} = 32 \times 2 \times F_s$$

Figure 17-31 Definition of Audio Sampling Frequency



In master mode, the linear divider should be configured correctly and properly to obtain the desired audio frequency.

Figure 17-32 I²S Clock Generator Architecture



In Figure 17-32, the I2SxCLK source is the system clock (HSI, HSE, or PLL that drives the AHB clock).

The audio sampling frequency can be 192 kHz, 96 kHz, 48 kHz, 44.1 kHz, 32 kHz, 22.05 kHz, 16 kHz, 11.025 kHz, or 8 kHz (or any other values within this range). In order to reach the desired frequency, the linear divider needs to be programmed according to the formulas below:

When the master clock is generated (the I2SMCLKOE bit is '1' in the SPI_I2SCLKP register):

When the channel frame is 16-bit wide, $F_s = I2SxCLK / [(16*2)^* ((2*I2SDIV) + I2SODD)*8]$

When the channel frame is 32-bit wide, $F_s = I2SxCLK / [(32*2)^* ((2*I2SDIV) + I2SODD)*4]$

When the master clock is disabled (the I2SMCLKOE bit is '0'):

When the channel frame is 16-bit wide, $F_s = I2SxCLK / [(16*2)^* ((2*I2SDIV) + I2SODD)]$

When the channel frame is 32-bit wide, $F_s = I2SxCLK / [(32*2)^* ((2*I2SDIV) + I2SODD)]$

Table 17-2 provides an example of the precision values for different clock configurations.

Note: Other configurations can also be used to achieve the optimum clock precision.

Table 17-2 Audio-frequency Precision Using System Clock

SysCLK (MHz)	MCLK	Target Fs (Hz)	16 bits				32 bits			
			I2sDI V	I2S O DD	Real Fs	Error	I2sDI V	I2S O DD	Real Fs	Error
200	No	192000	16	1	189393.9	1.36%	8	0	195312.5	1.73%
200	No	96000	32	1	96153.85	0.16%	16	1	94696.97	1.36%
200	No	48000	65	0	48076.92	0.16%	32	1	48076.92	0.16%
200	No	44100	71	0	44014.08	0.19%	35	1	44014.08	0.19%
200	No	32000	97	1	32051.28	0.16%	49	0	31887.76	0.35%
200	No	22050	141	1	22084.81	0.16%	71	0	22007.04	0.19%
200	No	16000	195	1	15984.65	0.10%	97	1	16025.64	0.16%
200	No	11025	283	1	11022.93	0.02%	141	1	11042.4	0.16%
200	No	8000	390	1	8002.561	0.03%	195	1	7992.327	0.10%
200	Yes	192000	3	0	130208.3	32.18%	3	0	130208.3	32.18%
200	Yes	96000	4	0	97656.25	1.73%	4	0	97656.25	1.73%
200	Yes	48000	8	0	48828.13	1.73%	8	0	48828.13	1.73%
200	Yes	44100	9	0	43402.78	1.58%	9	0	43402.78	1.58%
200	Yes	32000	12	0	32552.08	1.73%	12	0	32552.08	1.73%
200	Yes	22050	17	1	22321.43	1.23%	17	1	22321.43	1.23%
200	Yes	16000	24	1	15943.88	0.35%	24	1	15943.88	0.35%
200	Yes	11025	35	1	11003.52	0.19%	35	1	11003.52	0.19%
200	Yes	8000	49	0	7971.939	0.35%	49	0	7971.939	0.35%
100	No	192000	8	0	195312.5	1.73%	4	0	195312.5	1.73%
100	No	96000	16	1	94696.97	1.36%	8	0	97656.25	1.73%
100	No	48000	32	1	48076.92	0.16%	16	1	47348.48	1.36%
100	No	44100	35	1	44014.08	0.19%	17	1	44642.86	1.23%
100	No	32000	49	0	31887.76	0.35%	24	1	31887.76	0.35%
100	No	22050	71	0	22007.04	0.19%	35	1	22007.04	0.19%
100	No	16000	97	1	16025.64	0.16%	49	0	15943.88	0.35%
100	No	11025	141	1	11042.4	0.16%	71	0	11003.52	0.19%
100	No	8000	195	1	7992.327	0.10%	97	1	8012.821	0.16%
100	Yes	96000	2	0	97656.25	1.73%	2	0	97656.25	1.73%
100	Yes	48000	4	0	48828.13	1.73%	4	0	48828.13	1.73%
100	Yes	44100	4	1	43402.78	1.58%	4	1	43402.78	1.58%
100	Yes	32000	6	0	32552.08	1.73%	6	0	32552.08	1.73%
100	Yes	22050	9	0	21701.39	1.58%	9	0	21701.39	1.58%
100	Yes	16000	12	0	16276.04	1.73%	12	0	16276.04	1.73%
100	Yes	11025	17	1	11160.71	1.23%	17	1	11160.71	1.23%
100	Yes	8000	24	1	7971.939	0.35%	24	1	7971.939	0.35%
72	No	192000	6	0	187500	2.34%	3	0	187500	2.34%
72	No	96000	11	1	97826.09	1.90%	6	0	93750	2.34%
72	No	48000	32	1	34615.38	27.88%	11	1	48913.04	1.90%
72	No	44100	25	1	44117.65	0.04%	13	0	43269.23	1.88%
72	No	32000	35	0	32142.86	0.45%	17	1	32142.86	0.45%
72	No	22050	51	0	22058.82	0.04%	25	1	22058.82	0.04%
72	No	16000	70	1	15957.45	0.27%	35	0	16071.43	0.45%
72	No	11025	102	0	11029.41	0.04%	51	0	11029.41	0.04%
72	No	8000	140	1	8007.117	0.09%	70	1	7978.723	0.27%
72	Yes	96000	2	0	70312.5	26.76%	2	0	70312.5	26.76%
72	Yes	48000	3	0	46875	2.34%	3	0	46875	2.34%
72	Yes	44100	3	0	46875	6.29%	3	0	46875	6.29%
72	Yes	32000	4	1	31250	2.34%	4	1	31250	2.34%

72	Yes	22050	6	1	21634.62	1.88%	6	1	21634.62	1.88%
72	Yes	16000	9	0	15625	2.34%	9	0	15625	2.34%
72	Yes	11025	13	0	10817.31	1.88%	13	0	10817.31	1.88%
72	Yes	8000	17	1	8035.714	0.45%	17	1	8035.714	0.45%

17.3.2.4 I²S Master Mode

When I²S is configured in master mode, the serial clock is generated on the CK pin, and the Word Select signal is generated on the WS pin. Whether the master clock (MCLK) is output or not can be configured by the I2SMCLKOE bit in the SPI_I2SCLKP register.

Procedure

1. Set I2SDIV[9:0] in the SPI_I2SCLKP register to define the serial clock baud rate corresponding to audio sample frequency. The I2SODD bit in the SPI_I2SCLKP register also needs to be defined.
2. Set the CPOL bit to define the steady level for the communication clock. Set the MCLKOE bit in the SPI_I2SCLKP register if the master clock, MCLK, needs to be provided to the external DAC/ADC audio component (The I2SDIV and I2SODD values should be computed depending on the state of the MCLK output. For more details, please refer to [Section 17.4.3](#)).
3. Set the I2SEL bit in the SPI_I2SCTRL register to activate the I²S function, and select the I²S standard through the I2SAP[1:0] and PCMSYNCSEL bits. Set the data length through the DATLEN[1:0] bits. Select the number of bits per channel by configuring the CHLEN bit. Also set I2SMOD[1:0] in the SPI_I2SCTRLI2S register to select I²S master mode and direction (transmitter or receiver).
4. If needed, enable the desired interrupt and the DMA capabilities by writing the SPI_CTRL2 register.
5. The I2SEN bit in the SPI_I2SCTRL register must be set.
6. WS and CK pins need to be configured in output mode. The MCLK pin is also set as an output, if the I2SMCLKOE bit in SPI_I2SCLKP is set.

Transmission sequence

The transmission sequence begins when a half-word (16-bit) is written into the Tx buffer. Assume that the first data written into the Tx buffer corresponds to the left channel data. When data are transferred from the Tx buffer to the shift register, TE is set, and at this time data corresponding to the right channel have to be written into the Tx buffer. The I2SCS flag indicates which channel corresponds to the current data to be transmitted. The value of I2SCS is updated when TE is '1', so it will be meaningful when the TE is '1'.

A full frame is recognized at the end of a left channel data transmission followed by a right channel data transmission. It is not possible to have a partial frame where only the left channel is sent.

The half-word data is parallelly loaded into the 16-bit shift register during the first bit transmission, and then the following bits are shifted out serially, MSB first, to the MOSI/SD pin. The TE flag is set after each data transfer from the Tx buffer to the shift register. An interrupt is generated if the TEIE bit in the SPI_CTRL2 register is set.

For more details about the write operations depending on the I²S standard mode selected, please refer to [Section 17.4.2](#).

To ensure a continuous audio data transmission, it is advised to write the SPI_DT register with the next data to be transmitted before the end of the current transmission. When writing the last data, wait until TE = 1 and BSY = 0, and then disable I²S function by clearing the I2SEN bit.

Reception sequence

The operation of reception sequence mode is the same as the transmission mode, except for the point 3 (Please refer to the previous section of "transmission sequence"), where the master reception mode is selected by configuring I2SMOD[1:0]. Whatever the data or channel length is, the audio data are received as 16-bit packets. This means that each time when the Rx buffer is full, the RNE flag is set, and an interrupt is generated if the RNEIE bit is set in the SPI_CTRL2 register. According to the data and channel length configured, the audio value received for a right or left channel may require one or two receptions into the Rx buffer.

Reading the SPI_DT register can clear the RNE flag bit.

I2SCS is updated after each reception. It is sensitive to the WS signal generated by the I²S cell.

For more details about the read operations depending on the selected I²S standard mode, please refer to [Section 17.4.2](#).

If new data is received while the previously received data is not read yet, an overrun is generated, and the OVR flag is set. If the ERRIE bit is set in the SPI_CTRL2 register, an interrupt is generated to indicate the error.

To switch off the I²S, specific actions are required to ensure that the I²S completes the transfer cycle properly without initiating a new data transfer. The sequence depends on the configuration of the data and channel lengths, also the audio protocol mode selected. Please refer to the following:

- 16-bit data extended on 32-bit channel length (DLEN = 00 and CHLEN = 1), in LSB-justified mode (I2SAP = 10)
 - a) Wait for the second to last (n - 1) RNE = 1
 - b) Then, wait 17 I²S clock cycles (using a software loop)
 - c) Disable I²S (I2SEN = 0)
- 16-bit data extended on 32-bit channel length (DLEN = 00 and CHLEN = 1), in MSB-justified, I²S, or PCM modes (I2SAP = 00, or I2SAP = 01, or I2SAP = 11)
 - a) Wait for the last RNE = 1
 - b) Then, wait 1 I²S clock cycle (using a software loop)
 - c) Disable the I²S (I2SEN = 0)
- For all other combinations of DLEN and CHLEN, whatever the audio mode is selected through I2SAP, carry out the following sequence to switch off I²S:
 - a) Wait for the second to last (n - 1) RNE = 1
 - b) Then, wait 1 I²S clock cycle (using a software loop)
 - c) Disable I²S (I2SEN = 0)

Note: The BSY flag is always kept low during transfers.

17.3.2.5 I²S Slave Mode

In slave mode, the I²S can be configured in transmission or reception mode. The operating mode mainly follows the same rules as described in the I²S master configuration. In slave mode, I²S interface does not need to provide the clock. The clock and WS signals are provided by the external I²S master, connected to the corresponding pins. Therefore, users do not have to configure the clock.

The configuration steps are as follows:

1. Set the I2SEL bit in the SPI_I2SCTRL register to activate the I²S function; configure I2SAP[1:0] to select desired I²S standard; program DLEN[1:0] to select the bit number of data; configure CHLEN to choose number of bit per channel. Select data direction (transmission or reception) in I²S slave mode through I2SMOD[1:0] in the SPI_I2SCTRL register.
2. If needed, enable interrupt and DMA functions by setting the SPI_CTRL2 register.
3. The I2SEN bit in SPI_I2SCTRL register must be set.

Transmission sequence

The transmission sequence begins when the external master device sends the clock, and when the NSS_WS signal requests data transfer. The slave has to be enabled and the I²S data register has to be loaded before the external master starts the communication.

For the I²S MSB justified and LSB justified modes, the first data item written into the data register corresponds to the left channel data. When the communication starts, the data is transferred from the Tx buffer to the shift register. The TE flag is then set, and the data corresponding to the right channel is written into the I²S data register.

The I2SCS flag indicates that data to be transmitted corresponds to which channel. Compared to the master transmission mode, in slave mode, I2SCS is sensitive to the WS signal from the external I²S master. This means that the I²S slave needs to be ready to transmit the first data before the clock is generated by the master. For I²S MSB-justified and LSB-justified mode, WS signal assertion means the left channel transmitted first.

Note: The I2SEN bit has to be set at least two PCLK cycles before the first clock signal of the I²S master on the CK pin.

Half-word data is parallelly loaded into the 16-bit shift register from the I²S internal bus during the first bit transmission, and then is serially shifted out to the MOSI/SD pin, MSB first. The TE flag is set after each transfer from the Tx buffer to the shift register, and an interrupt is generated if the TEIE bit in the SPI_CTRL2 register is set.

Note: Before any attempt to write the Tx buffer, ensure that the TE flag is '1'.

For more details about the write operations depending on the I²S standard mode selected, please refer to [Section 17.4.2](#). To secure a continuous audio data transmission, it is advised to write the SPI_DT register with the next data to be transmitted before the end of the current transmission. An underrun flag is set and an interrupt may be generated if the new data is not written into the SPI_DT register before the first clock edge of the next data communication. This indicates that there is software data transmission error. If the ERRIE bit is set in the SPI_CTRL2 register, an interrupt is generated when the UDR flag in the SPI_STS register goes high. In this case, it is suggested to switch off the I²S and to restart a data transfer starting from the left channel.

When the last data is written, clear the I2SEN bit after waiting until TE = 1 and BSY = 0 to switch off the I²S.

Reception sequence

The configuration procedure is the same as the transmission mode except for the point 1, where the I2SMOD[1:0] bits are configured to select master reception mode. Whatever the data length or the channel length is, the audio data is always received as 16-bit packets. This means that each time the Rx buffer is full, the RNE flag is set, and an interrupt is generated if the RNEIE bit is set in the SPI_CTRL2 register. Depending on the data length and channel length configuration, the audio value received for a right or left channel may need one or two receptions into the Rx buffer.

The I2SCS flag is updated at each data reception (to be read from SPI_DT). It is sensitive to the external WS signal generated by I²S cell.

Reading the SPI_DT register, will clear the RNE bit.

For more details about the read operations depending on the selected I²S standard, please refer to [Section 17.4.2](#).

If new data is received while the precedent received data is not read, an overrun is generated, and the OVR flag is set. If the ERRIE bit is set in the SPI_CTRL2 register, an interrupt is generated to indicate the error.

To switch off I²S in reception mode, the I2SEN bit has to be cleared immediately after receiving the last RNE = 1.

Note: The external I²S master components should have the capability of sending/receiving data in 16-bit or 32-bit packets via an audio channel.

17.3.2.6 Status Flag

Four status flags are provided for users to monitor the state of the I²S bus.

Busy flag (BSY)

The BSY flag is set and cleared by hardware (writing to this flag has no effect). It indicates the state of the I²S communication layer.

When BSY is set, it indicates that I²S is busy communicating, but here is one exception: In master receive mode (I2SMOD = 11), the BSY flag is always kept low during reception.

The BSY flag can be used to detect the end of a transfer before the software disables SPI. This avoids corrupting the last transfer. Therefore, the procedure described below must be strictly followed.

The BSY flag is set when a transfer starts, unless the I²S is in master receiver mode.

The BSY flag is cleared:

- When a transfer completes (except for master transmit mode, in which the communication is continuous).
- When I²S is disabled.

When communication is continuous:

- In master transmit mode, the BSY flag is kept high during all the transfers.
- In slave mode, the BSY flag goes low for one I²S clock cycle between each transfer.

Note: Do not use the BSY flag for each data transmission or reception. It is better to use the TE and RNE flags.

Tx buffer empty flag (TE)

When set, this flag indicates that the Tx buffer is empty, and the next data to be transmitted can then be

loaded into it. The TE flag is reset when the Tx buffer already contains data to be transmitted. It is also reset when the I²S is disabled (the I2SEN bit is '0').

Rx buffer not empty flag (RNE)

When set, this flag indicates that there are valid received data in the Rx buffer. It is reset when the SPI_DT register is read.

Channel side flag (I2SCS)

In transmission mode, this flag is refreshed when TE goes high. It indicates the channel side to which the data from the SD pin is transferred. If an underrun error occurs in slave transmission mode, the value of this flag is invalid. The I²S needs to be switched off and then switched on before resuming the communication.

In reception mode, this flag is refreshed when data is received into SPI_DT. It indicates from which channel side data is received.

Note: If errors occur (such as overrun, OVR), this flag becomes meaningless, and I²S should be reset and then enabled again (also, change the I²S configuration if needed).

This flag is meaningless in the PCM standard for both short and long frame modes.

When the OVR or UDR flag in SPI_STS is set and the ERRIE bit in SPI_CTRL2 is also set, an interrupt is generated. This interrupt can be cleared by reading the SPI_STS register (once the interrupt source is cleared).

17.3.2.7 Error Flag

There are two error flags for the I²S cell.

Underrun flag (UDR)

In slave transmission mode, this flag is set when the first clock edge for data transmission arrives, but new data is not yet loaded into the SPI_DT register. This bit is valid when the I2SEL bit in SPI_I2SCTRL is set. An interrupt is generated if the ERRIE bit in SPI_CTRL2 is set.

The UDR bit is cleared by a read operation to the SPI_STS register.

Overrun flag (OVR)

Overrun errors occur and the flag is set when new data is received, but the previous received data is not yet read. An interrupt is generated if the ERRIE bit is set in SPI_CTRL2, to indicate that an error occurs.

In this case, the contents of Rx buffer are not updated with the newly received data from the transmitter device. A read operation to the SPI_DT register returns the previous correctly received data. All other subsequently transmitted 16-bit data after the overrun event are lost.

The bit is cleared by a read operation to the SPI_DT register followed by a read operation to the SPI_STS register.

17.3.2.8 I²S Interrupt

Table 17-3 lists all the I²S interrupts.

Table 17-3 I²S Interrupt Request

Interrupt Event	Event Flag	Enable Flag
Tx buffer empty flag	TE	TEIE
Rx buffer not empty flag	RNE	RNEIE
Overrun Flag	OVR	ERRIE
Underrun Flag	UDR	

17.3.2.9 DMA Function

DMA works in exactly the same way as in the I²S mode and in SPI mode, except that the CRC feature is not available in I²S mode since there is no data transfer protection system.

17.4 SPI Registers

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

Table 17-4 SPI Register Map and Reset Values

17.4.1 SPI Control Register 1 (SPI_CTRL1) (Not Used in I²S Mode)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BD MO DE	BD OE	CCE	CTN	DFF 16	RO NLY	SW NSS EN	ISS	LSB EN	SPI EN	MCLKP[2:0]	MST EN	CP OL	CPH A		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15	BDMODE: Bidirectional data mode enable 0: "2-line unidirectional" mode is selected. 1: "1-line bidirectional" mode is selected. Note: This bit is not used in I ² S mode.
Bit 14	BDOE: Output enable in bidirectional mode This bit and the BDMODE bit together determine the data output direction in "1-line bidirectional" mode. 0: Output is disabled. (Receive-only mode) 1: Output is enabled. (Transmit-only mode) The data line is MOSI pin in master mode, and the MISO pin in slave mode. Note: This bit is not used in I ² S mode.
Bit 13	CCE: Hardware CRC calculation enable 0: CRC calculation is enabled. 1: CRC calculation is disabled. Note 1: This bit should be written only when SPI is disabled (SPIEN = 0), else an error may occur. It can only be used in full-duplex mode. Note 2: This bit is not used in I ² S mode.
Bit 12	CTN: Transmit CRC next 0: Next transmitted value comes from Tx buffer. 1: Next transmitted value comes from Tx CRC register. Note 1: This bit is set once the last data is written into the SPI_DT register Note 2: This bit is not used in I ² S mode.
Bit 11	DFF16: Data frame format 0: 8-bit data frame format is selected for transmission/reception. 1: 16-bit data frame format is selected for transmission/reception. Note 1: This bit should be written only when SPI is disabled (SPIEN = 0), else an error may occur. Note 2: This bit is not used in I ² S mode.
Bit 10	RONLY: Receive only This bit and the BDMODE bit together determine the transfer direction in "2-line bidirectional" mode. In multi-slave configuration, this bit is set on the slave that is never accessed. Thus, only the accessed slaves have output, which avoids conflicts on data lines. 0: Full-duplex (Transmission and reception) 1: Output is disabled (Receive-only mode). Note: This bit is not used in I ² S mode.
Bit 9	SWNSSEN: Software slave management When SWNSSEN is set, NSS pin level is determined by the ISS bit. 0: Software slave management is disabled. 1: Software slave management is enabled. Note: This bit is not used in I ² S mode.
Bit 8	ISS: Internal slave select This bit has meaning only when the SWNSSEN bit is set. It determines the level on the NSS, and I/O operations on the NSS pin are invalid. Note: This bit is not used in I ² S mode.
Bit 7	LSBEN: Frame format 0: MSB first 1: LSB first Note 1: This bit cannot be changed when communication is ongoing. Note 2: This bit is not used in I ² S mode.

Bit 6	SPIEN: SPI enable 0: SPI is disabled. 1: SPI is enabled. Note 1: This bit is not used in I ² S mode. Note 2: When SPI is to be disabled, follow the procedure in Section 17.3.1.8 .
Bit 5:3	MCLKP[2:0]: Baud rate control MCLKP[3] bit is in SPI_CTRL2 register, MCLKP[3:0]: 0000: f _{PCLK} /2 0001: f _{PCLK} /4 0010: f _{PCLK} /8 0011: f _{PCLK} /16 0100: f _{PCLK} /32 0101: f _{PCLK} /64 0110: f _{PCLK} /128 0111: f _{PCLK} /256 1000: f _{PCLK} /512 1001: f _{PCLK} /1024 These bits cannot be changed when communication is ongoing. They can be changed only when SPIEN is disabled. Note: This bit is not used in I ² S mode.
Bit 2	MSTEN: Master selection 0: Slave configuration 1: Master configuration Note 1: This bit cannot be changed when communication is ongoing. Note 2: This bit is not used in I ² S mode.
Bit 1	CPOL: Clock polarity 0: SCK keeps low at idle state. 1: SCK keeps high at idle state. Note 1: This bit cannot be changed when communication is ongoing. Note 2: This bit is not used in I ² S mode.
Bit 0	CPHA: Clock phase 0: Data capture starts from the first clock edge. 1: Data capture starts from the second clock edge. Note 1: This bit cannot be changed when communication is ongoing. Note 2: This bit is not used in I ² S mode.

17.4.2 SPI Control Register 2 (SPI_CTRL2)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved		MCL KP[3]	TEIE	RNEI E	ERRI E	Reserved	NSS OE	DMA TEN	DMA REN	
					res		rw	rw	rw	rw	res	rw	rw	rw	

Bit 15:9	Reserved. Forced to be '0' by hardware.
Bit 8	MCLKP[3]: Baud rate control Please refer to MCLKP[2:0] in the SPI_CTRL1 register.
Bit 7	TEIE: Tx buffer empty interrupt enable 0: TE interrupt is disabled. 1: TE interrupt is enabled. An interrupt is generated when the TE flag is set.
Bit 6	RNEIE: Rx buffer not empty interrupt enable 0: RNE interrupt is disabled. 1: RNE interrupt is enabled. An interrupt is generated when the RNE flag is set.
Bit 5	ERRIE: Error interrupt enable This bit controls interrupt generation when errors occur (CERR, OVR, or MODF). 0: Error interrupt is disabled. 1: Error interrupt is enabled.
Bit 4:3	Reserved. Forced to be '0' by hardware.

Bit 2	NSSOE: SS output enable 0: SS output is disabled in master mode, and the device can work in multi-master mode. 1: SS output is enabled in master mode when the device is activated. The device cannot work in multimaster mode. Note: This bit is not used in I ² S mode.
Bit 1	DMATEN: DMA Tx enable When this bit is set, DMA request will be sent once the TE flag is set. 0: Tx buffer DMA is disabled. 1: Tx buffer DMA is enabled.
Bit 0	DMAREN: DMA Rx enable When this bit is set, DMA request will be sent once the RNE flag is set. 0: Rx buffer DMA is disabled. 1: Rx buffer DMA is enabled.

17.4.3 SPI Status Register (SPI_STS)

Address offset: 0x08

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								BSY	OVR	MO DF	CERR	UD R	I ² S CS	TE	RNE
Reserved								r	r	r	r_w0	r	r	r	r
res															

Bit 15:8	Reserved. Forced to be '0' by hardware.
Bit 7	BSY: Busy flag 0: SPI is not busy. 1: SPI is busy, or the Tx buffer is not empty. This bit is set or reset by hardware. Note: Special attention must be given to the use of this flag, please refer to Section 17.3.1.7 and Section 17.3.1.8 .
Bit 6	OVR: Overrun flag 0: No overrun error 1: Overrun error occurs. This bit is set by hardware and reset by software sequence. For more details about the software sequence, please refer to Section 17.3.2.7 .
Bit 5	MODF: Mode fault 0: No mode fault 1: Mode fault occurs. This bit is set by hardware and reset by software sequence. For more details about the software sequence, please refer to Section 17.3.2.7 . Note: This bit is not used in I ² S mode.
Bit 4	CERR: CRC error flag 0: The received CRC value matches with the value in the SPI_RCRC register. 1: The received CRC value does not match with the value in the SPI_RCRC register. This bit is set by hardware and reset by software writing '0'. Note: This bit is not used in I ² S mode.
Bit 3	UDR: Underrun flag 0: No underrun error 1: Underrun error occurs. This bit is set by hardware and reset by software sequence, please refer to Section 17.3.2.7 . Note: This bit is not used in SPI mode.
Bit 2	I²SCS: Channel side 0: Channel left is to be transmitted or received. 1: Channel right is to be transmitted or received. Note: This bit is not used in SPI mode, and has no meaning in PCM mode.

Bit 1	TE: Transmit buffer empty 0: Tx buffer is not empty. 1: Tx buffer is empty.
Bit 0	RNE: Receive buffer not empty 0: Rx buffer is not empty. 1: Rx buffer is empty.

17.4.4 SPI Data Register (SPI_DT)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:0															
<p>DT[15:0]: Data register Data to be transmitted or data already received The data register includes 2 buffers: One for writing (Tx buffer) and the other one for reading (Rx buffer). A write operation will send data into the Tx buffer; a read operation will return the data held in the Rx buffer. Note for SPI mode: Depending on the data frame format selection, the DFF16 bit in the SPI_CTRL1 register, the data sent or received is either 8-bit or 16-bit. This selection has to be made before enabling the SPI to ensure correct operation. For 8-bit data frame, the buffers are 8-bit, and only the SPI_DT[7:0] is used for transmission/reception. In reception mode, SPI_DT[15:8] is forced to 0. For 16-bit data frame, the buffers are 16-bit and the entire data register, SPI_DT[15:0], is used for transmission/reception.</p>															

17.4.5 SPICRC Polynomial Register (SPI_CPOLY) (Not Used in I²S Mode)

Address offset: 0x10

Reset value: 0x0007

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
<p>Bit 15:0</p> <p>CPOLY[15:0]: CRC polynomial register This register contains the polynomial used for CRC calculation. Its reset value is 0x0007, which can be changed according to the application. Note: This bit is not used in I²S mode.</p>															

17.4.6 SPIRxCRC Register (SPI_RCRC) (Not Used in I²S Mode)

Address offset: 0x14

Reset value: 0x0000

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

Bit 15:0	<p>RCRC[15:0]: Rx CRC register</p> <p>When CRC calculation is enabled, RCRC[15:0] contains the CRC value computed based on the received bytes. This register is reset when the CCE bit in SPI_CTRL1 register is written '1'. The CRC is calculated with the polynomial programmed in the SPI_CPOLY register.</p> <p>Only the 8 LSB bits are considered when the data frame format is set to 8-bit data. CRC calculation is done based on CRC8 standard. The entire 16 bits of this register are considered when 16-bit data frame format is selected. CRC calculation is done based on CRC16 standard.</p> <p>Note 1: A read to this register when the BSY flag is set may return an incorrect value.</p> <p>Note 2: These bits are not used in I²S mode.</p>
----------	---

17.4.7 SPITxCRC Register (SPI_TCRC)

Address offset: 0x18

Reset value: 0x0000

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

Bit 15:0	<p>TCRC[15:0]: Tx CRC register</p> <p>When CRC calculation is enabled, TCRC[15:0] contains the CRC value computed based on the bytes to be transmitted. This register is reset when the CCE bit of SPI_CTRL1 is written '1'. The CRC is calculated with the polynomial programmed in the SPI_CPOLY register.</p> <p>Only the 8 LSB bits are considered when the data frame format is set to 8-bit. CRC calculation is done based on CRC8 standard. The entire 16 bits of this register are considered when a 16-bit data frame format is selected. CRC calculation is done based on CRC16 standard.</p> <p>Note: A read to this register when the BSY flag is set may return an incorrect value.</p> <p>Note: These bits are not used in I²S mode.</p>
----------	--

17.4.8 SPI_I2S Configuration Register (SPI_I2SCTRL)

Address offset: 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	I2SS EL	I2SEN	I2SMOD[1:0]	PCMS YNCSEL	Reserve d	I2SAP[1:0]	CPO L	DLEN[1:0]	CHL EN						
res	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 15:12	Reserved. Forced to be '0' by hardware.														
Bit 11	<p>I2SEL: I²S mode selection</p> <p>0: SPI mode is selected. 1: I²S mode is selected.</p> <p>Note: This bit can be configured only after SPI or I²S is disabled.</p>														
Bit 10	<p>I2SEN: I²S enable</p> <p>0: I²S is disabled. 1: I²S is enabled.</p> <p>Note: This bit is not used in SPI mode.</p>														

Bit 9:8	I2SMOD[1:0]: I ² S configuration mode 00: Slave transmission 01: Slave reception 10: Master transmission 11: Master reception Note: This bit can be configured only when I ² S is disabled. This bit is not used in SPI mode.
Bit 7	PCMSYNCSEL: PCM frame synchronization 0: Short frame synchronization 1: Long frame synchronization Note: This bit is meaningful only when I2SAP = 11 (PCM standard is used). This bit is not used in SPI mode.
Bit 6	Reserved. Forced to be '0' by hardware.
Bit 5:4	I2SAP[1:0]: I ² S standard selection 00: I ² S Philips standard 01: MSB-justified standard (Left-justified) 10: LSB-justified standard (Right-justified) 11: PCM standard For more details about I ² S standard, please refer to Section 17.3.2.2 . Note: For correct operation, this bit can be configured only when I ² S is disabled. This bit is not used in SPI mode.
Bit 3	CPOL: Steady state clock polarity 0: I ² S clock steady state is low level. 1: I ² S clock steady state is high level. Note: For correct operation, this bit can be configured only when I ² S is disabled. This bit is not used in SPI mode.
Bit 2:1	DLEN[1:0]: Data length to be transferred 00: 16-bit data length 01: 24-bit data length 10: 32-bit data length 11: Not allowed Note: For correct operation, this bit can be configured only when I ² S is disabled. This bit is not used in SPI mode.
Bit 0	CHLEN: Channel length (Number of bits per audio channel) 0: 16-bit wide 1: 32-bit wide Write operation to this bit is meaningful only if DLEN = 00; otherwise, the channel length is fixed to 32-bit by hardware. Note: For correct operation, this bit can be configured only when I ² S is disabled. This bit is not used in SPI mode.

17.4.9 SPI_I2S Prescaler Register (SPI_I2SCLKP)

Address offset: 0x20

Reset value: 0x0002

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		I2SDIV[9:8]	I2SM CLK OE	I2SO DD	I2SDIV[7:0]										
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 15:12	Reserved. Forced to be '0' by hardware.
Bit 12:11	I2SDIV[9:8]: I ² S linear prescaler Please refer to the description of I2SDIV[7:0].
Bit 9	I2SMCLKOE: Master clock output enable 0: Master clock output is disabled. 1: Master clock output is enabled. Note: For correct operation, this bit can be configured only when I ² S is disabled. It is only used in I ² S master mode. This bit is not used in SPI mode.
Bit 8	I2SODD: Odd factor for the prescaler 0: Actual divider factor = I2SDIV*2 1: Actual divider factor = (I2SDIV*2) + 1 Please refer to Section 17.3.2.3 . Note: For correct operation, this bit can be configured only when I ² S is disabled. It is only used in I ² S master mode. This bit is not used in SPI mode.
Bit 7:0	I2SDIV[7:0]: I ² S linear prescaler I2SDIV[9:8] is configured in Bit 12:11. It is not allowed to configure I2SDIV[9:0] = 0 or I2SDIV[9:0] = 1 Please refer to Section 17.3.2.3 . Note: For correct operation, this bit can be configured only when I ² S is disabled. It is only used in I ² S master mode. This bit is not used in SPI mode.

18 CAN Bus Controller

18.1 Introduction

bxCAN is the abbreviation of Basic Extended CAN. It supports the CAN protocols version 2.0A and 2.0B. It is designed to manage a large quantity of incoming messages efficiently with a minimum CPU load. It also meets the priority requirements for message transmissions (The priority is software configurable). For safety-critical applications, bxCAN provides all hardware functions needed to support Time-Triggered Communication option.

18.2 Main Features

- Supports CAN protocol version 2.0A and 2.0B active mode
- Baud rate up to 1 Mbit/s
- Supports time-triggered communication

Transmission

- Three transmit mailboxes
- Software configurable transmit priority
- Time stamp on SOF transmission

Reception

- 2 receive FIFO with 3-level depth
- Scalable filter banks:
 - 14 filter banks
- Identifier list
- Configurable FIFO overrun handling
- Time stamp on SOF reception

Time-triggered communication mode

- Disable automatic retransmission
- 16-bit free running timer
- Time stamp sent in last two data bytes

Management

- Maskable interrupts
- Software-efficient mailbox mapping at a unique address space

CAN SRAM Memory

- CAN manages the 512-byte SRAM memory.

Note: *USB and CAN share a dedicated 512-byte SRAM memory for data transmission and reception, so they cannot be used concurrently (the shared SRAM is accessed through CAN and USB exclusively). USB and CAN can be used in the same application but not at the same time.*

18.3 Function Overview

18.3.1 CAN Overall Function Description

In current CAN applications, the number of nodes in a CAN network is increasing, and several networks are often linked together via gateways. Therefore, the number of messages (handled by each node) in the system has significantly increased. In addition to the application messages, Network Management and Diagnostic messages are also introduced.

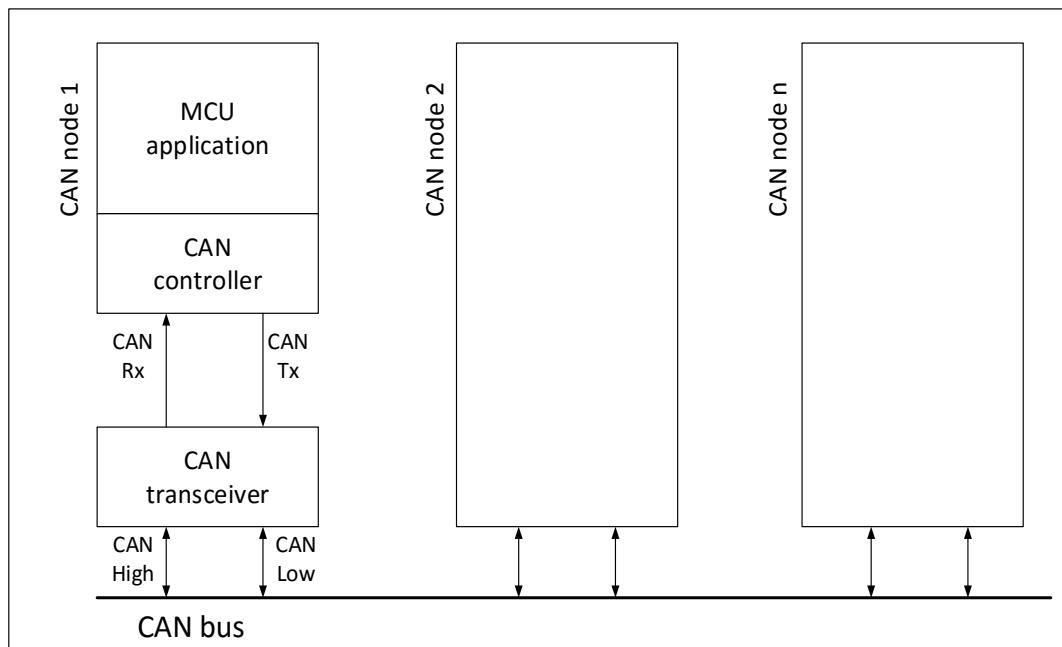
- An enhanced filtering mechanism is required to handle different types of messages.

In addition, since application tasks take more CPU time, real-time constraints caused by message reception should be reduced.

- A receive FIFO scheme allows the CPU to be dedicated to application tasks for a long period of time without losing messages.

The Higher Layer Protocol based on standard CAN drivers requires an efficient interface to the CAN controller.

Figure 18-1 CAN Network Topology



bxCAN module can completely transmit and receive CAN messages automatically; it fully supports standard identifiers (11-bit) and extended identifiers (29-bit).

The application can use these registers to:

- Configure CAN parameters, e.g. baud rate
- Request transmissions
- Handle receptions
- Manage interrupts
- Get diagnostic information

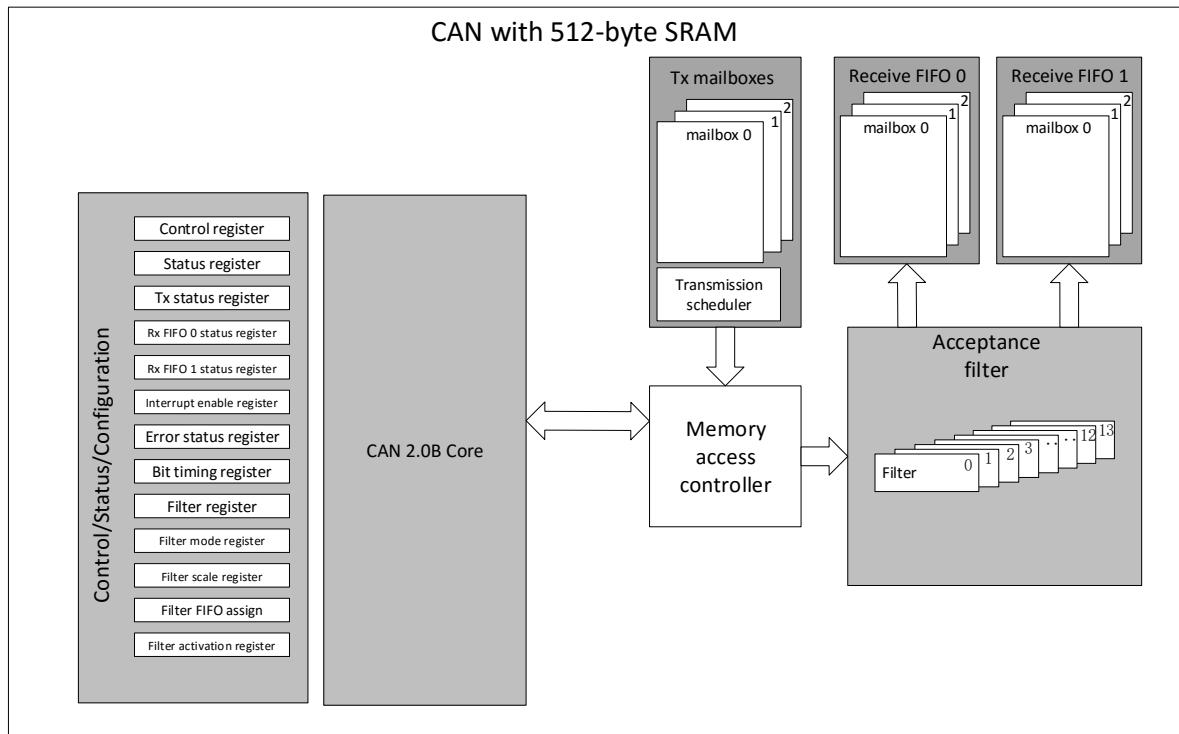
Three transmit mailboxes are provided to the software for sending messages. The transmission scheduler decides messages in which mailbox are to be transmitted first.

bxCAN provides 14 scalable/configurable identifier filter banks, which are programmed by software to select the messages needed and discard the others.

Receive FIFO

There are two receive FIFOs, and each can store three complete messages. The FIFOs are managed completely by hardware.

Figure 18-2 CAN Block Diagram



18.3.2 Operating Mode

bxCAN has three main operating modes: Initialization, Normal, and Sleep. After hardware reset, bxCAN is in Sleep mode to reduce power consumption, and an internal pullup is activated on the CANTX pin. The software requests bxCAN to enter Initialization or Sleep mode by setting the INRQ or SLP bit in the CAN_MCTRL register. Once entering Initialization mode or Sleep mode, bxCAN confirms by setting the IAK or SAK bits in the CAN_MSTS register, and the internal pull-up is disabled. When both IAK and SAK are '0', bxCAN is in Normal mode. Before entering Normal mode, bxCAN always has to be synchronized with the CAN bus. To synchronize, bxCAN waits until the CAN bus is idle, which means that 11 consecutive recessive bits are monitored on CANRX.

18.3.2.1 Initialization Mode

The software initialization should be done while the hardware is in Initialization mode. To enter this mode, set the INRQ bit in the CAN_MCTRL register, and then wait until the hardware confirms the request by setting the IAK bit in the CAN_MSTS register. To leave initialization mode, clear the INRQ bit in the CAN_MCTRL register. bxCAN will leave Initialization mode once the IAK bit is cleared by hardware.

While bxCAN is in Initialization mode, all message transfers are stopped, and the CANTX pin outputs recessive bits (high level). Entering Initialization mode does not change any of the configuration registers.

To initialize bxCAN, software has to include the bit timing (CAN_BTMG) and control (CAN_MCTRL) registers. Before initializing bxCAN filter banks (mode, scale, FIFO assignment, activation, and filter values), software has to set the FINT bit in the CAN_FM register. Filter initialization can be done outside the Initialization mode.

Note: When FINT = 1, message reception is disabled.

The filter value can be modified by clearing the filter activation bits (in the CAN_FA1 register).

If filter banks are not used, they should be kept in deactivation (Keep the FEN bit cleared).

18.3.2.2 Normal Mode

Once the initialization is completed, the software must request the hardware to enter Normal mode to start regular reception and transmission. The request to enter Normal mode is issued by software, clearing the INRQ bit in the CAN_MCTRL register, and then wait until the hardware sets the IAK bit in the CAN_MSTS register to confirm. The bxCAN enters Normal mode and is ready to transmit and receive messages when it is synchronized with the CAN bus, which means a sequence of 11 consecutive recessive bits (Bus Idle state) on the CANRX pin is detected.

There is no need to set the filter values in Initialization mode, but it must be done while the filter is not active (the corresponding FEN bit cleared). The filter scale and mode configuration must be done before entering Normal mode from Initialization mode.

18.3.2.3 Sleep Mode (Low Power)

bxCAN can work in low-power mode, Sleep mode. To enter this mode, software makes request by setting the SLP bit in the CAN_MCTRL register. In this mode, the bxCAN clock is stopped, but software can still access the bxCAN mailboxes.

When bxCAN is in Sleep mode, software can request to enter Initialization mode by setting the INRQ bit in the CAN_MCTRL register, and at the same time clears the SLP bit.

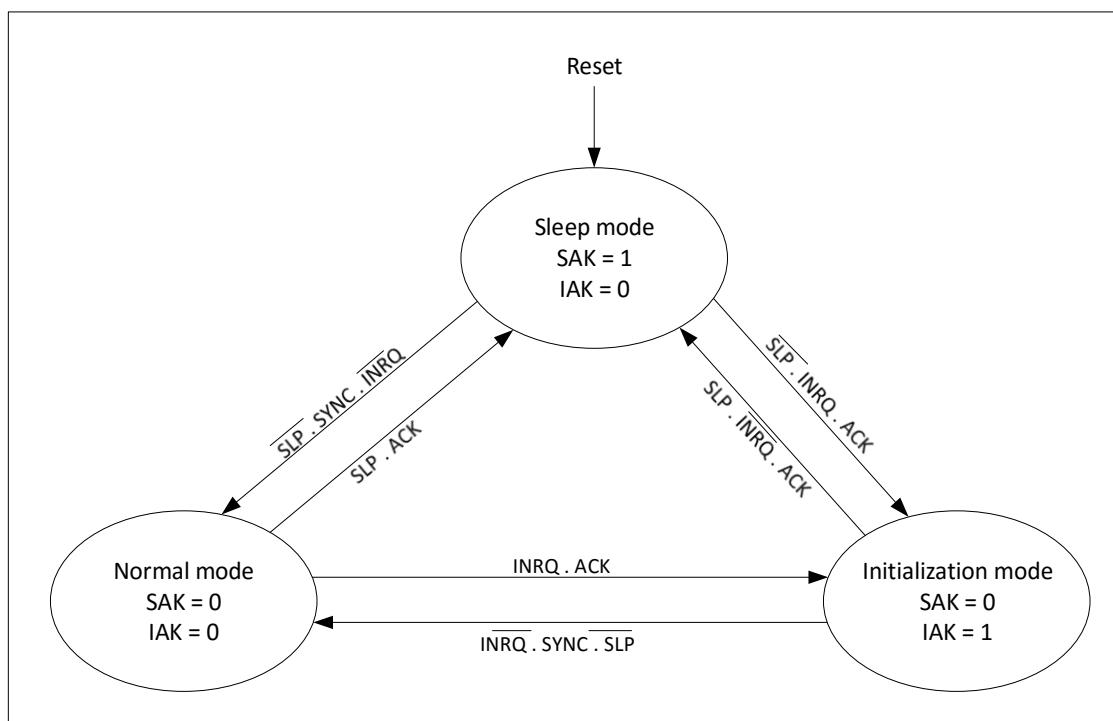
There are two ways to wake up bxCAN (exit Sleep mode): Clearing the SLP bit with software or detection of CAN bus activity through hardware.

On CAN bus activity detection, hardware automatically performs the wakeup sequence by clearing the SLP bit if the AWU bit in the CAN_MCTRL register is set. If the AWU bit is cleared in the CAN_MCTRL register, software has to clear the SLP bit when a wakeup interrupt occurs to exit from Sleep mode.

Note: *If wakeup interrupt is enabled (the WKIE bit set in the CAN_INTEN register), a wakeup interrupt will be generated on detection of CAN bus activity, no matter whether hardware automatically wakes up bxCAN or not.*

After the SLP bit is cleared, the exit from Sleep mode must be synchronized with the CAN bus, please refer to [Figure 18-3](#). The exit from Sleep mode is confirmed once the SAK bit is cleared by hardware.

Figure 18-3 bxCAN Operating Mode



Note:

1. **ACK** = The state during which hardware confirms Sleep or Initialization mode request and sets the IAK or SAK bits in the CAN_MSTS register
2. **SYNC** = The state during which bxCAN waits until the CAN bus is idle, which means that 11 consecutive recessive bits are monitored on CANRX.

18.3.3 Test Mode

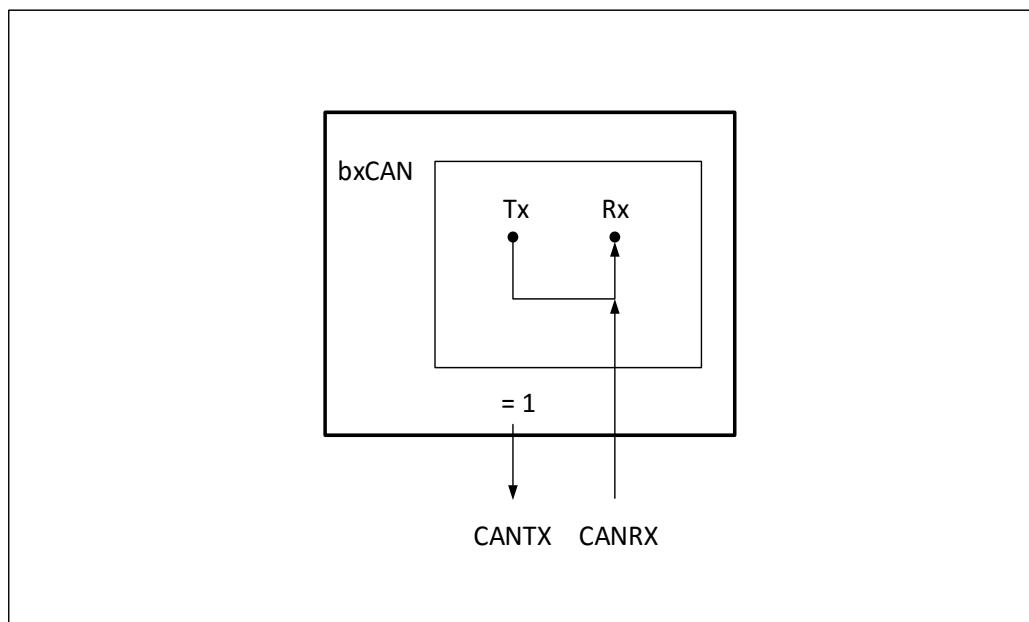
Test modes can be selected by the SIL and/or LBK bits in the CAN_BTMG register. These two bits must be configured in Initialization mode. Once Test mode is selected, the INRQ bit in the CAN_MCTRL register must be reset to successfully enter Test mode.

18.3.3.1 Silent Mode

Silent mode is selected by setting the SIL bit in the CAN_BTMG register. In Silent mode, bxCAN can receive valid data frames and valid remote frames, but it sends only recessive bits and cannot

successfully start message transmission. If the bxCAN has to send a dominant bit (ACK bit, overload flag, and active error flag), this dominant bit is rerouted internally so that the CAN core can monitor it. At the same time, the CAN bus is not affected and still remains in recessive state. Therefore, Silent mode is usually used to analyze the traffic on a CAN bus without affecting it: Dominant bits (acknowledge bits and error frames) will not be transmitted to the bus.

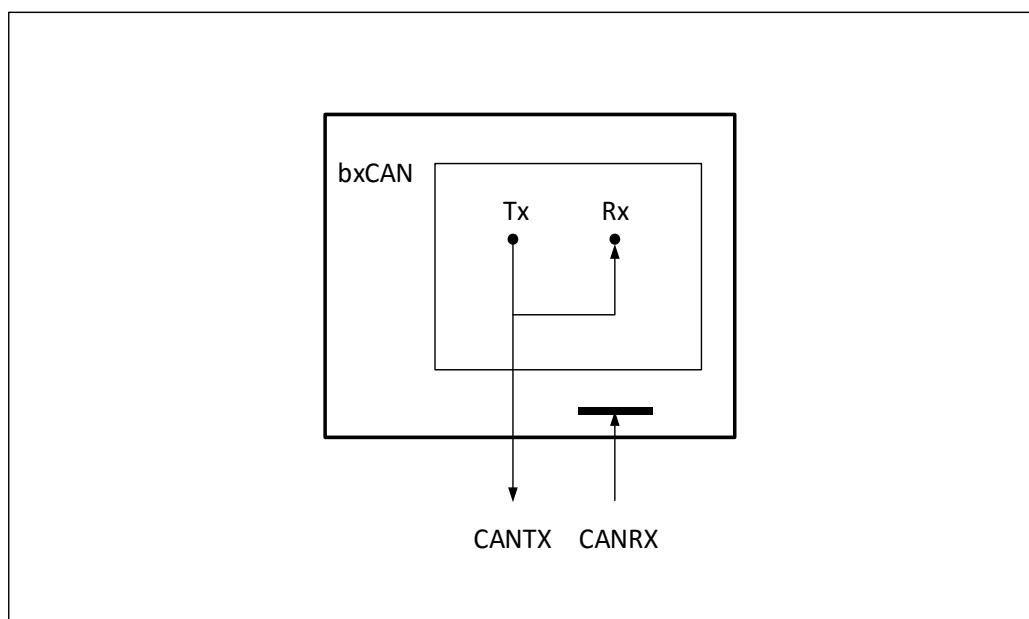
Figure 18-4 bxCAN in Silent Mode



18.3.3.2 Loopback Mode

Loopback mode can be selected by setting the LBK bit in the CAN_BTMG register. In Loopback mode, bxCAN stores its own transmitted messages as received messages (if they pass acceptance filtering) in a receive mailbox.

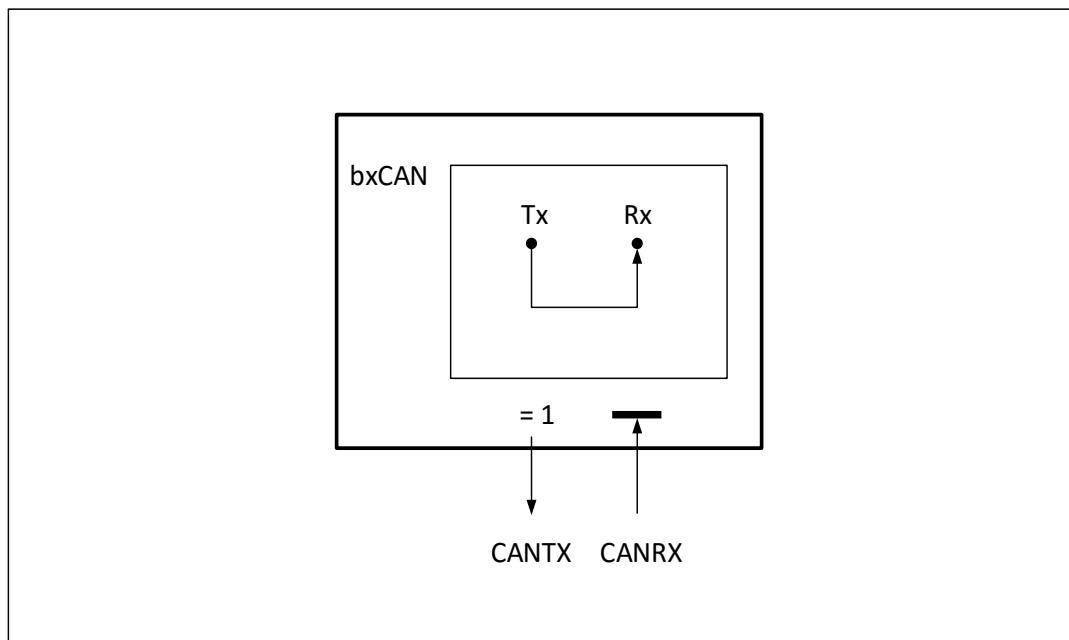
Figure 18-5 bxCAN in Loopback Mode



Loopback mode can be used for self-test functions. To be independent of external events, the CAN core ignores acknowledge errors (no dominant bit sampled in the acknowledge slot of a data/remote frame) in Loopback mode. In this mode, bxCAN performs an internal feedback from its Tx output to its Rx input and disregards the actual value of the CANRX pin. The transmitted messages can be monitored on the CANTX pin.

18.3.3.3 Loopback and Silent Mode

Figure 18-6 bxCAN in Loopback and Silent Mode



It is also possible to combine Loopback mode and Silent mode by setting both the LBK and SIL bits in the CAN_BTMG register. This mode can be used for a “Hot Self-test”, which means that the bxCAN can be tested like in Loopback mode but without affecting the whole CAN system connected to the CANTX and CANRX pins. In this mode, the CANRX pin is disconnected from the CAN bus, and the CANTX pin is held recessive.

18.3.4 AT32F403 in Debug Mode

When the microcontroller enters debug mode, Cortex®-M4 core halted, the bxCAN continues to work normally or stops, depending on the configuration of the following bits:

- The DBG_CAN1_STOP bit of CAN1 in debug module (DBG)
- The DBF bit in CAN_MCTRL

18.3.5 Transmission Handling

Message transmission process is as follows: The application selects one empty transmit mailbox; then, it configures the identifier, the data length, and the data to be transmitted; finally, it sets the TRQ bit in the CAN_TMIx register to make transmission request. Once the TRQ bit is set, the mailbox is not empty anymore, and software no longer has write access to the mailbox registers. Immediately after the TRQ bit is set, the mailbox enters pending state and waits to become the highest priority mailbox (Please refer to “Transmit Priority.”) As soon as the mailbox becomes the top priority, it will be scheduled for transmission. The message transmission of the scheduled mailbox will start (enter transmit state) right after the CAN bus becomes idle. Once the messages in the mailbox are successfully transmitted, it becomes empty again. The hardware then indicates a successful transmission by setting the RQC and TOK bits in the CAN_TSTS register accordingly.

If the transmission fails, setting the ALS bit in the CAN_TSTS register indicates an arbitration lost, and setting the TER bit indicates transmission error.

Transmit priority

1. Determined by identifier

When more than one transmit mailbox is pending, the transmission priority is given by the identifier of the message in the mailbox. The message with the lowest identifier value has the highest priority according to the arbitration of the CAN protocol. If the identifier values are equal, mailbox with the lower number will be scheduled first.

2. Determined by transmit request order

The transmit mailboxes can be configured as transmit FIFO by setting the TFP bit in the

CAN_MCTRL register. In this mode, the priority order is given by the transmit request order. This mode is very useful for segmented transmission.

Abort

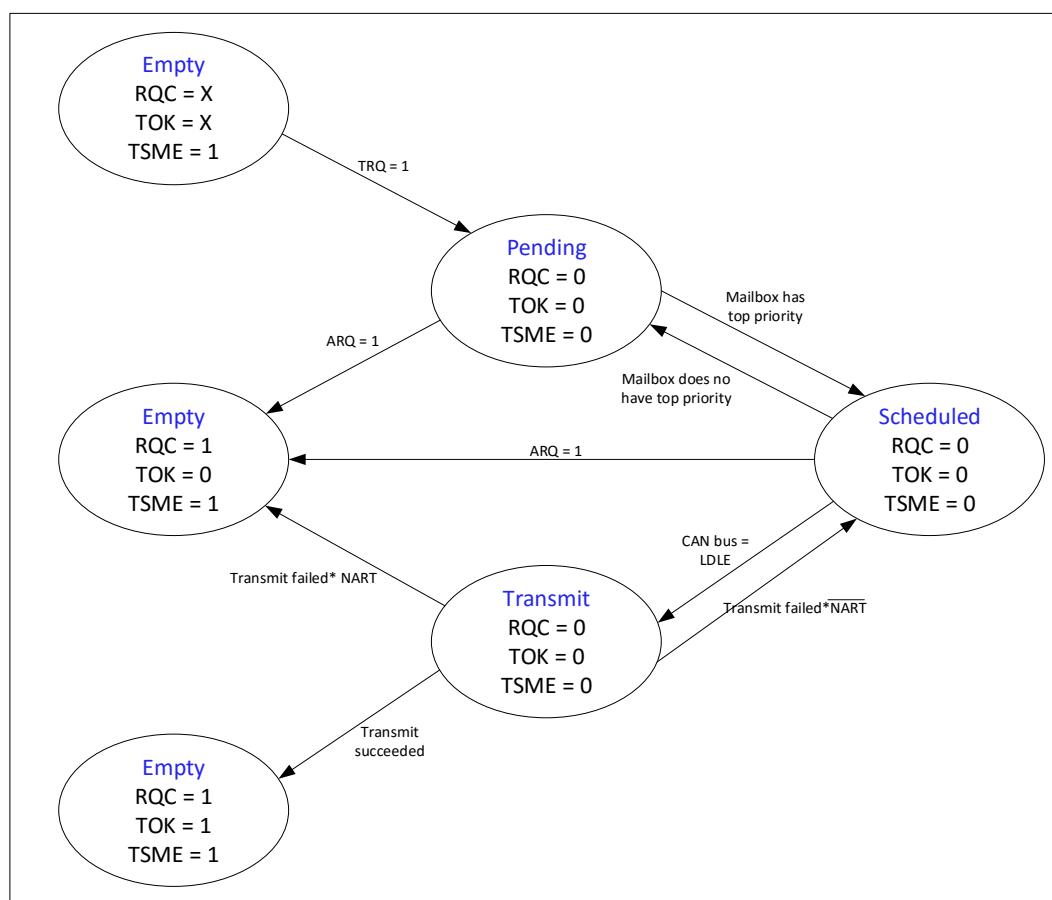
A transmission request can be aborted by setting the ARQ bit in the CAN_TSTS register. If the mailbox is in pending or scheduled state, transmit request is aborted immediately. An abort request while the mailbox is in transmit state can cause two results. If the messages in the mailbox is transmitted successfully, the mailbox becomes empty with the TOK bit set in the CAN_TSTS register. If the transmission fails, the mailbox becomes scheduled, the transmission is aborted, and the mailbox becomes empty with TOK cleared by hardware. Therefore, if the mailbox is in transmission, it will become empty after the current transmission is completed.

Non-automatic retransmission mode

This mode is implemented to fulfil the requirement of time-triggered communication option in the CAN standard. To configure the hardware in this mode, the NART bit in the CAN_MCTRL register must be set. In this mode, each transmission is performed only once. If the first attempt fails due to an arbitration loss or an error, the hardware will not automatically restart the message transmission.

At the end of a transmission, the hardware considers the request completed and sets the RQC bit in the CAN_TSTS register. The result of the transmission is indicated by the TOK, ALS, and TER bits.

Figure 18-7 Transmit Mailbox States



18.3.6 Time-triggered Communication Mode

In this mode, the internal timer of the CAN hardware is activated and used to generate the time stamps (for Tx and Rx mailboxes), which are stored in the CAN_RDTx/CAN_TDTx registers, respectively. The internal timer is incremented each CAN bit time. The internal timer is captured on the sample point of the start of frame bit in both reception and transmission, generating time stamps.

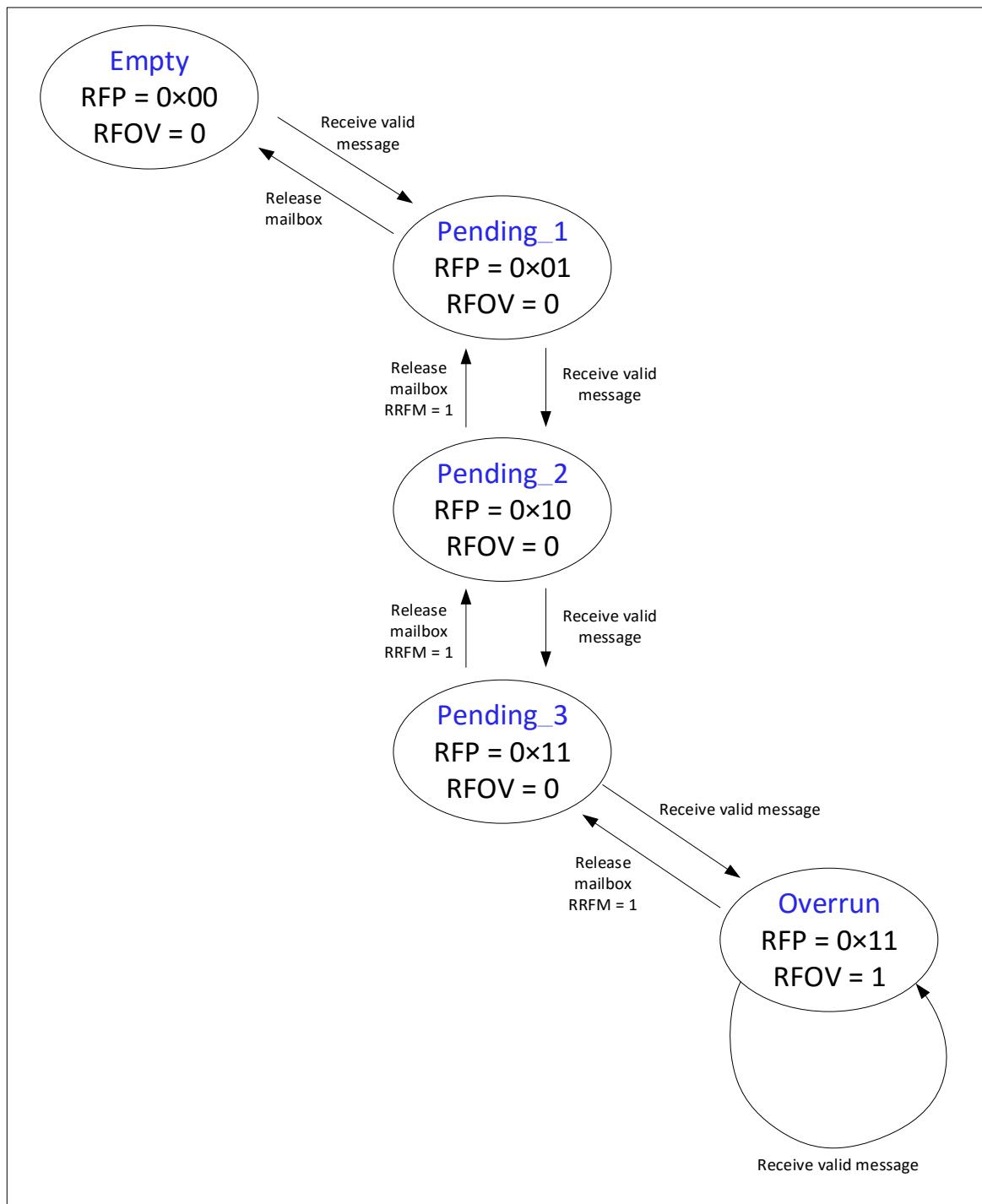
18.3.7 Reception Handling

The received messages are stored in FIFO with three-level mailbox depth. In order to save CPU load, simplify the software, and ensure data consistency, the FIFO is completely managed by hardware. The application accesses the messages received first in the FIFO only through the FIFO output mailbox.

Valid message

A received message is considered valid when it is received correctly according to the CAN protocol (no error until the last bit of the EOF field), and it passes through the identifier filtering successfully.

Figure 18-8 Receive FIFO States



FIFO management

Starting from the empty state, FIFO becomes pending_1 after the first valid message is received. The hardware configures RFP[1:0] in the CAN_RF register as '01'(binary 01b). The software can read the FIFO output mailbox to obtain the messages, and release the mailbox by setting the RRFM bit in the CAN_RF register. Then, the FIFO becomes empty again. If a new valid message is received in the meantime, the FIFO stays in pending_1 state, and software can read the FIFO output mailbox to obtain new messages.

If the application does not release the mailbox, FIFO enters pending_2 state after receiving the next valid message. Hardware configures RFP[1:0] as '10' (binary 10b). The above storage process is

repeated for the third valid message, which puts the FIFO into pending_3 state ($RFP[1:0] = 11b$). At this point, the software must release the mailbox by setting the RRFM bit, so that FIFO has enough room to store the next valid message. Otherwise, the next valid message received will cause a loss of message.

Overrun

When the FIFO is in pending_3 state (i.e. the three mailboxes are full), the next valid message reception will lead to an overrun, and a message will be lost. In this case, the hardware sets the RFOV bit in the CAN_RF register to indicate the overrun. Which message is lost depends on the configuration of the FIFO:

- If the FIFO lock function is disabled (the RFL bit in the CAN_MCTRL register is cleared), the last message received in the FIFO will be overwritten by the new incoming message. In this way, the latest messages will not be lost.
- If the FIFO lock function is enabled (the RFL bit in the CAN_MCTRL register is set), the latest message will be discarded, and the software will read three oldest messages in the FIFO.

Reception-related interrupts

Once a message is stored in the FIFO, the $RFP[1:0]$ bits are updated by hardware, and an interrupt request is generated if the RFPIE bit in the CAN_INTEN register is set.

When the FIFO becomes full (i.e. the third message is stored), the RFFU bit in the CAN_RF register is set, and a FULL interrupt is generated if the RFFUIE bit in the CAN_INTEN register is set. In addition, an overrun interrupt is generated if the RFOVIE bit in the CAN_INTEN register is set.

18.3.8 Identifier Filtering

In the CAN protocol, the identifier of a message is not associated with the address of a node, but related to the content of the message. Consequently, a transmitter broadcasts its message to all receivers. On message reception, a receiver node decides (depending on the identifier value) whether the software needs the message or not. If the message is needed, it is copied into the SRAM; if not, the message is discarded without intervention by the software.

To fulfill this requirement, the bxCAN controller provides 14 configurable and scalable filter banks (13 ~ 0) to receive only the messages that the software needs. This hardware filtering saves CPU resources which would be otherwise taken to perform filtering by software. Each filter bank x consists of two 32-bit registers, CAN_FBXR1 and CAN_FBXR2.

Scalable width

The width of each filter bank can be scaled independently to fulfill the different needs of applications. According to the different filter scales, each filter bank can provide:

- One 32-bit filter, including: the SID[10:0], EID[17:0], IDT, and RTR bits
- Two 16-bit filters, including: the SID[10:0], IDT, RTR, and EID[17:15] bits

Furthermore, the filters can be configured in mask mode or in identifier list mode.

Mask mode

In mask mode, the identifier registers, together with mask registers, specifies which bits of the identifier are handled as "must match" or as "don't care".

Identifier list mode

In identifier list mode, the mask registers are used as identifier registers. Thus, instead of defining an identifier register and a mask register, two identifier registers are adopted. Each bit of the Rx message identifier should match the filter identifier.

Filter bank scale and mode configuration

The filter banks are configured through the corresponding CAN_FM register. Before configuring a filter bank, it must be deactivated by clearing the FEN bit in the CAN_FA1 register. The filter scale is configured through the corresponding FBSx bit in the CAN_FS1 register. The identifier list or identifier mask mode for the corresponding mask/identifier registers is configured through the FMSx bit in the CAN_FM1 register.

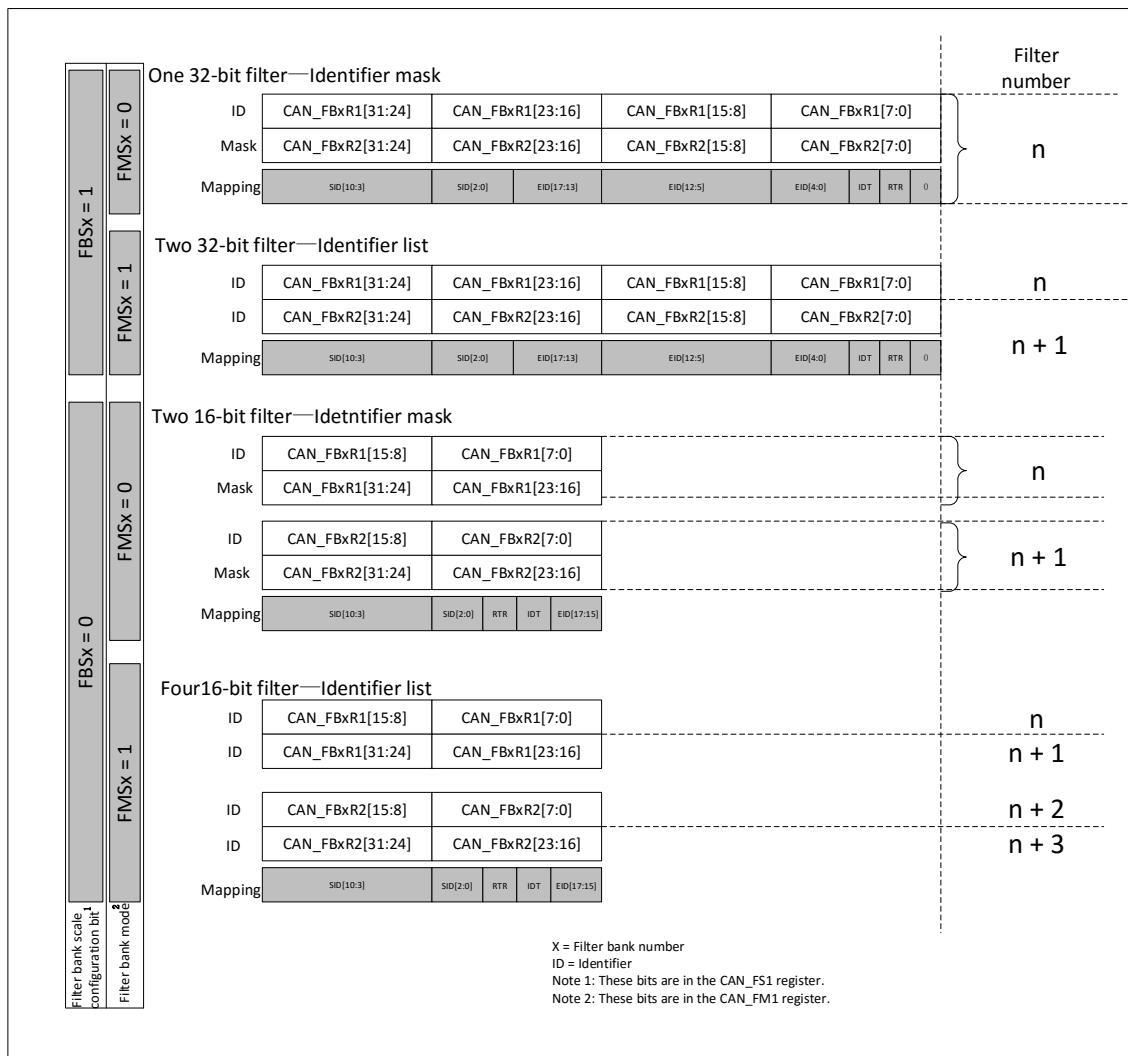
To filter a group of identifiers, configure the filter bank to work in mask mode. To select a single identifier, configure the filter bank to work in identifier list mode.

Filter banks not used by the application should be left deactivated.

Each filter within a filter bank is numbered (the Filter Number), from 0 to a maximum value, which depends on the mode and the scale configured in each filter bank.

For the filter configuration, please refer to Figure 18-9.

Figure 18-9 Filter Bank Scale Configuration - Register Organization



Filter match index

Once a message is received in the FIFO, it can be accessed by the application. Normally, message data is copied into SRAM; to copy the data to the right location, the application has to identify the data according to the identifier. bxCAN provides a filter match index to simplify this identifying process.

This index is stored in the mailbox, together with the message, according to the filter priority rules. Thus, each received message has its associated filter match index.

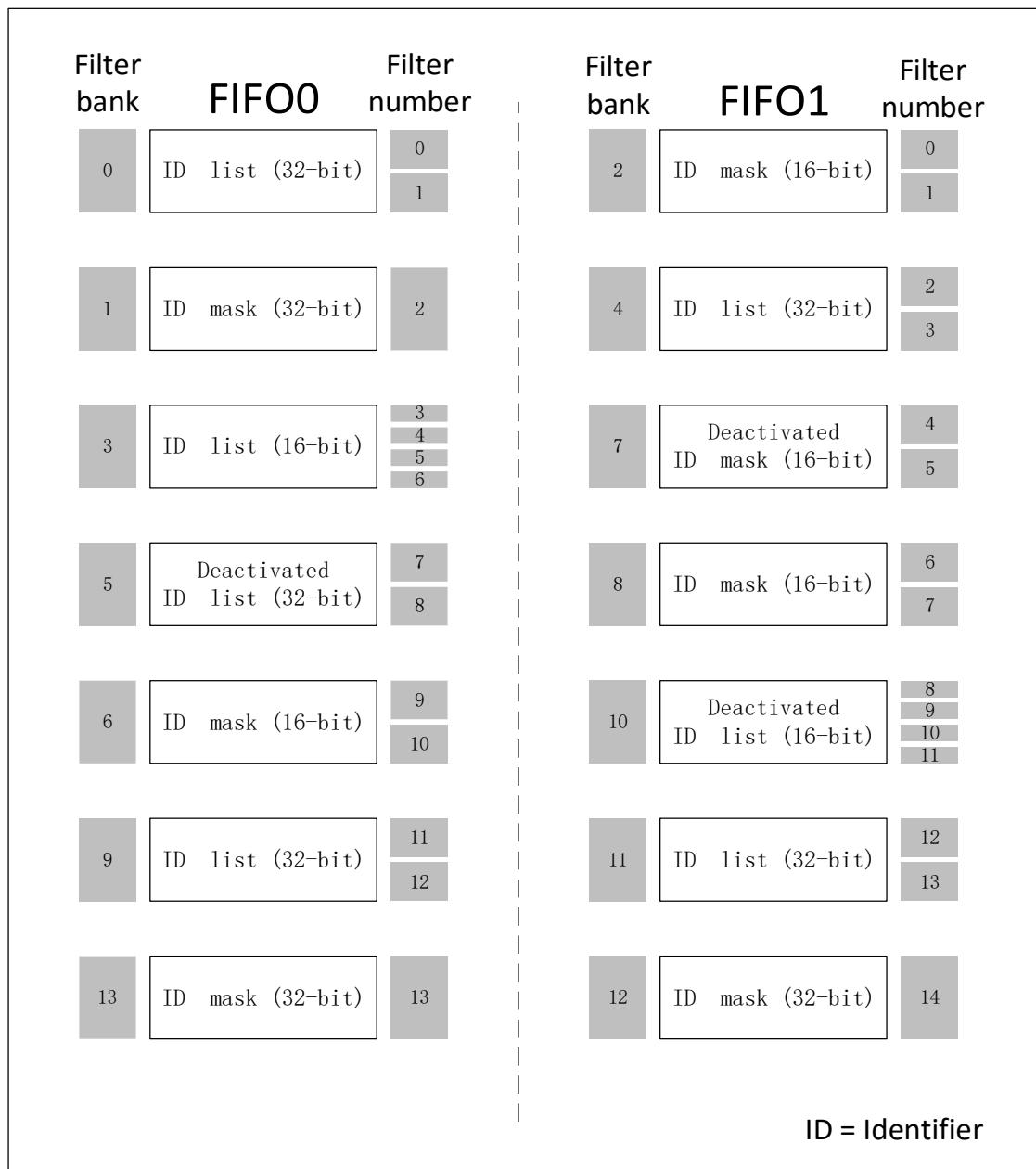
The filter match index can be used in the following two ways:

- Compare the filter match index with a list of expected values
- Use the filter match index as an index to access the data destination location

For filters in identifier list mode (nonmasked filters), the software no longer has to compare the identifier. For masked filters in mask mode, software only compares those masked bits needed (must match bits).

When numbering filters, activation state of the filter banks is not taken into account. In addition, each FIFO numbers its own associated filter independently. Please refer to the example shown in Figure 18-10.

Figure 18-10 Example of Filter Numbering

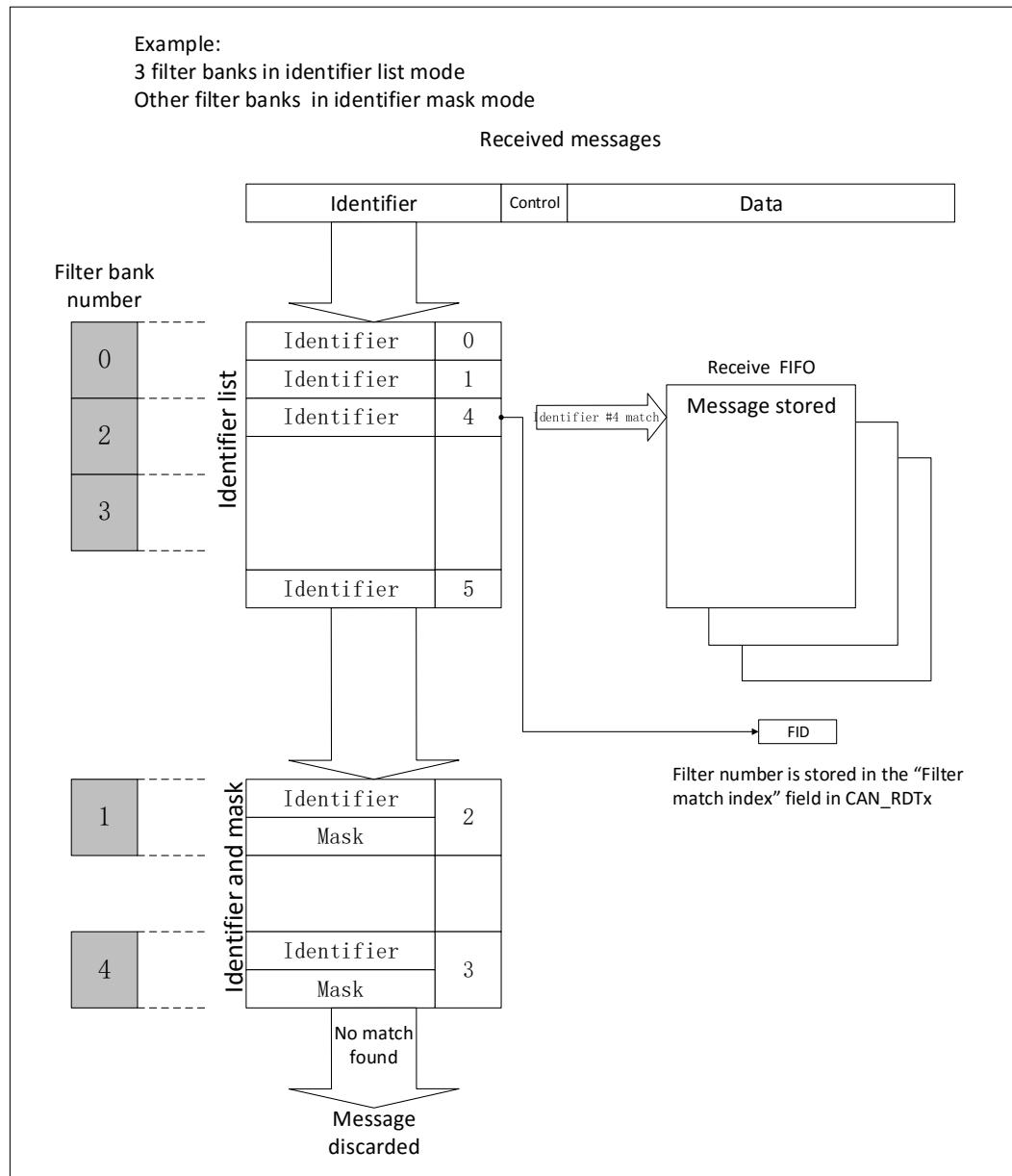


Filter priority rules

According to the different filter combinations, it is possible that a single identifier can pass through several filters successfully. In this case, the filter match value stored in the receive mailbox is determined by the following priority rules:

- A 32-bit filter takes priority over a 16-bit filter.
- For filters with equal scale, priority is given to the identifier list mode over the identifier mask mode.
- For filters with equal scale and in the same mode, priority is given according to the filter number (the lower the number, the higher the priority).

Figure 18-11 Example of Filtering Mechanism



The example above shows the filtering principle of the bxCAN: On reception of a message, its identifier is first compared with the filters configured in identifier list mode. If there is a match, the message is stored in the associated FIFO, and the index of the matched filter is stored in the filter match Index. As shown in the example, the identifier matches with Identifier #4, so the message content and FID4 are stored in the FIFO.

If there is no match, the identifier is then compared with the filters configured in mask mode. If the identifier does not match any of the identifiers configured in the filters, the message is discarded by hardware without disturbing the software.

18.3.9 Message Storage

Mailboxes are the interface between the software and the hardware for messages transfers. A mailbox contains all information related to a message: identifier, data, control, status, and time stamp information.

Transmit mailbox

The software should set up the message to be transmitted in an empty transmit mailbox (and then sends transmit requests). The status of the transmission is indicated in the CAN_TSTS register.

Table 18-1 Transmit Mailbox Mapping

Offset to Transmit Mailbox Base Address	Register Name
0	CAN_TMIx
4	CAN_TDTx
8	CAN_TDLx
12	CAN_TDlx

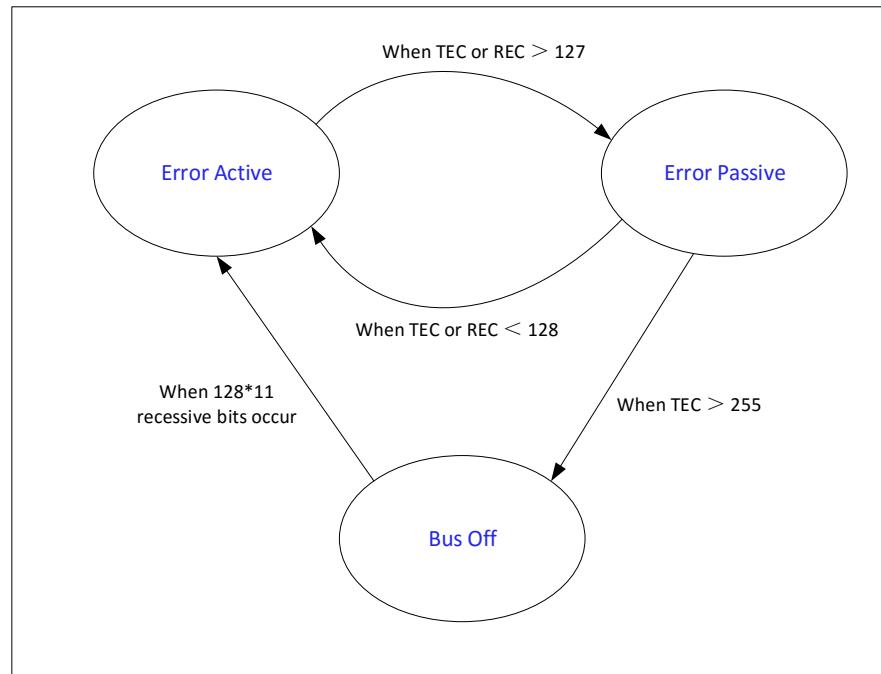
Receive mailbox (FIFO)

When a message is received, the software can access the FIFO output mailbox to read it. Once the software handles the message (e.g. read it), the software must release the message by setting the RRFM bit in the CAN_RFx register to make room to store the following received message. The filter match index is stored in the FID field of the CAN_RDTx register. The 16-bit time stamp value is stored in the TS[15:0] field of CAN_RDTx.

Table 18-2 Receive Mailbox Mapping

Offset to Receive Mailbox Base Address	Register Name
0	CAN_RFIx
4	CAN_RDTx
8	CAN_RDLx
12	CAN_RDlx

Figure 18-12 CAN Error State Diagram



18.3.10 Error Management

The error management as described in the CAN protocol is handled entirely by hardware with a Transmit Error Counter (the TEC field in the CAN_ESTS register) and a Receive Error Counter (the REC field in the CAN_ESR register). Its value is incremented or decremented according to the error. For detailed information about TEC and REC management, please refer to the CAN standard.

The value of both of the counter may be read by software to determine the stability of the CAN network.

Furthermore, the CAN_ESTS register provides detailed information of the current error status. By configuring the CAN_INTEN register (e.g. the ERIE bit.), the software can control the interrupt generation

on error detection in a flexible way.

Bus-off recovery

bxCAN enters bus-off state when TEC is greater than 255, and at the same time the BFF bit is set in the CAN_ESTS register. In bus-off state, bxCAN is unable to transmit and receive messages.

According to the configuration of ABO bit in the CAN_MCTRL register, bxCAN will recover from bus-off (become error active again) either automatically or on software request. But in both cases, the bxCAN has to wait for a recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on the CANRX pin).

If ABO is set, bxCAN will start the recovering sequence automatically after it enters bus-off state.

If ABO is cleared, the software must first request bxCAN to enter and then to leave Initialization mode before initiating the recovering sequence.

Note: *In Initialization mode, bxCAN does not monitor the CANRX signal, so it cannot complete the recovery sequence. To recover, bxCAN must be working in Normal mode.*

18.3.11 Bit Timing

The bit timing logic monitors the serial CAN bus through sampling, and adjust its sampling point by synchronizing with the start-bit edge and resynchronizing with the following edges.

Its operation can be briefly introduced as splitting every nominal bit time into three segments as follows:

- Synchronization segment (SYNC_SEG): Usually a bit change is expected to occur within this time segment. It has a fixed length of one time quantum ($1 \times t_{CAN}$).
- Bit segment 1 (BS1): This segment defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta but may be automatically lengthened to compensate for positive phase drifts caused by different frequencies of various nodes in the network.
- Bit segment 2 (BS2): This segment defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard. Its duration is programmable between 1 and 8 time quanta but may also be automatically shortened to compensate for negative phase drifts.

The re-Synchronization Jump Width (SJW) defines an upper bound to the amount of lengthening or shortening of the bit segments. It is programmable between 1 and 4 time quanta.

A valid edge is defined during the first transition from a dominant bit to a recessive bit when bxCAN itself does not send a recessive bit. If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended up to SJW, so the sample point will be delayed.

On the contrary, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened up to SJW, so the transmit point is moved earlier.

To avoid programming errors, the configuration of the bit timing register (CAN_BTMG) is only possible when bxCAN is in Initialization state.

Note: *For the detailed descriptions of the CAN bit timing and resynchronization mechanism, please refer to the ISO 11898 standard.*

Figure 18-13 Bit Timing

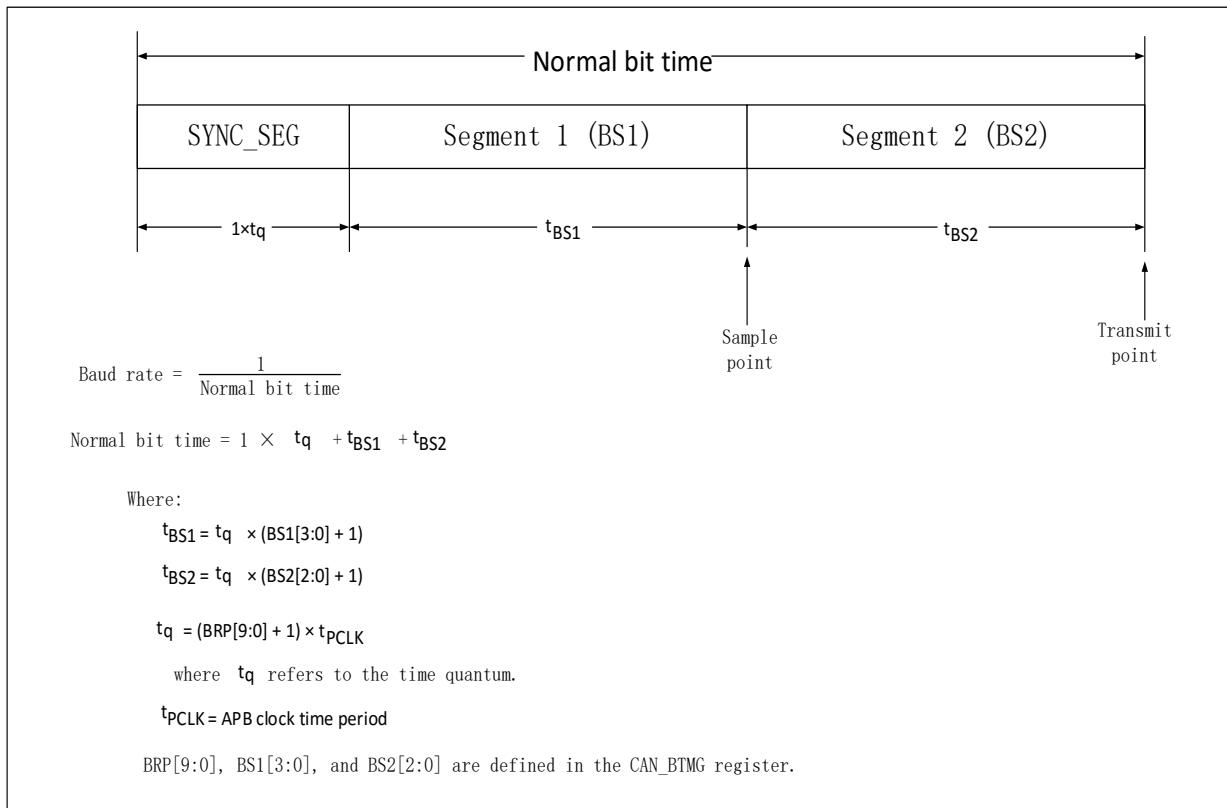
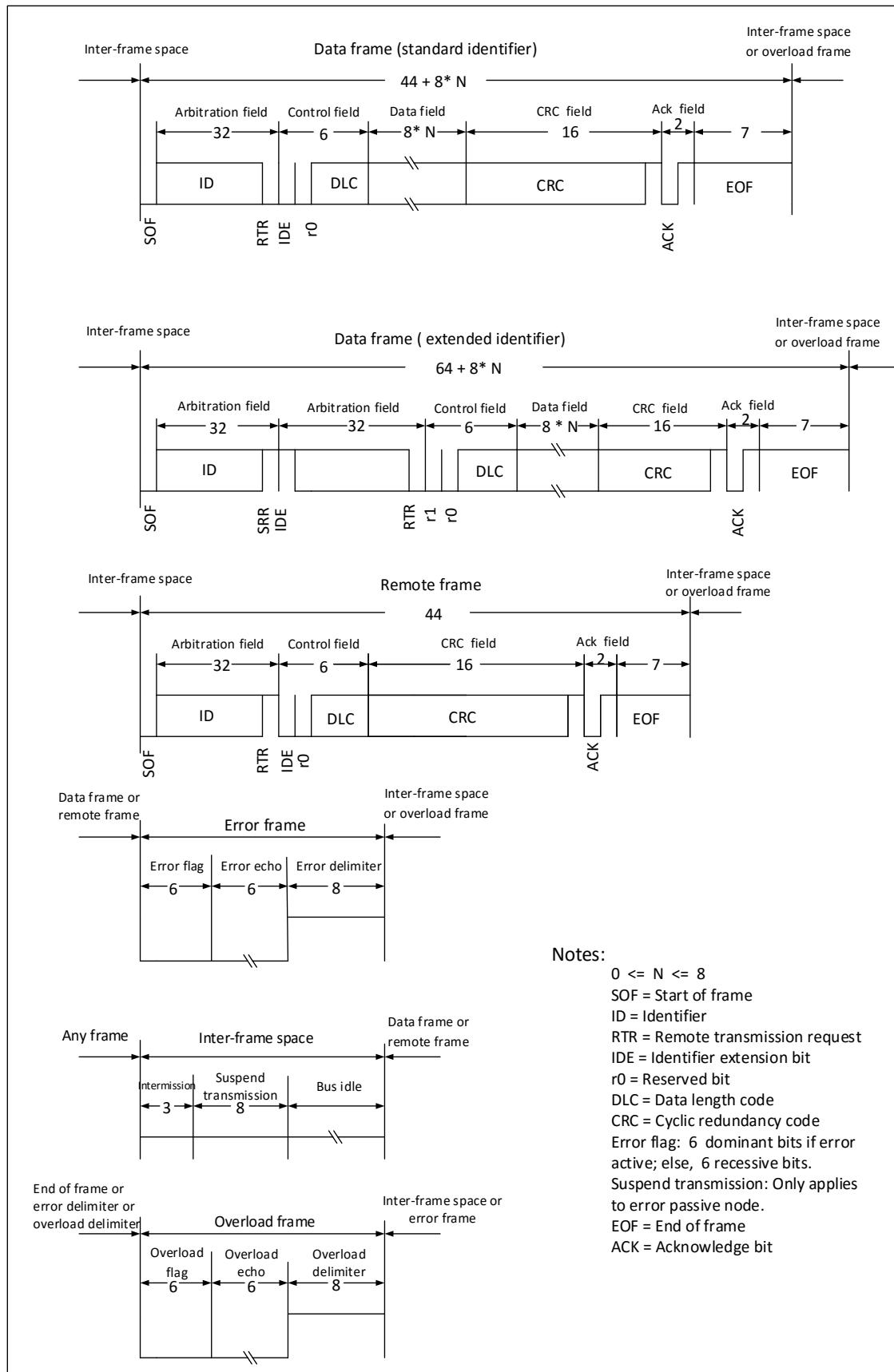


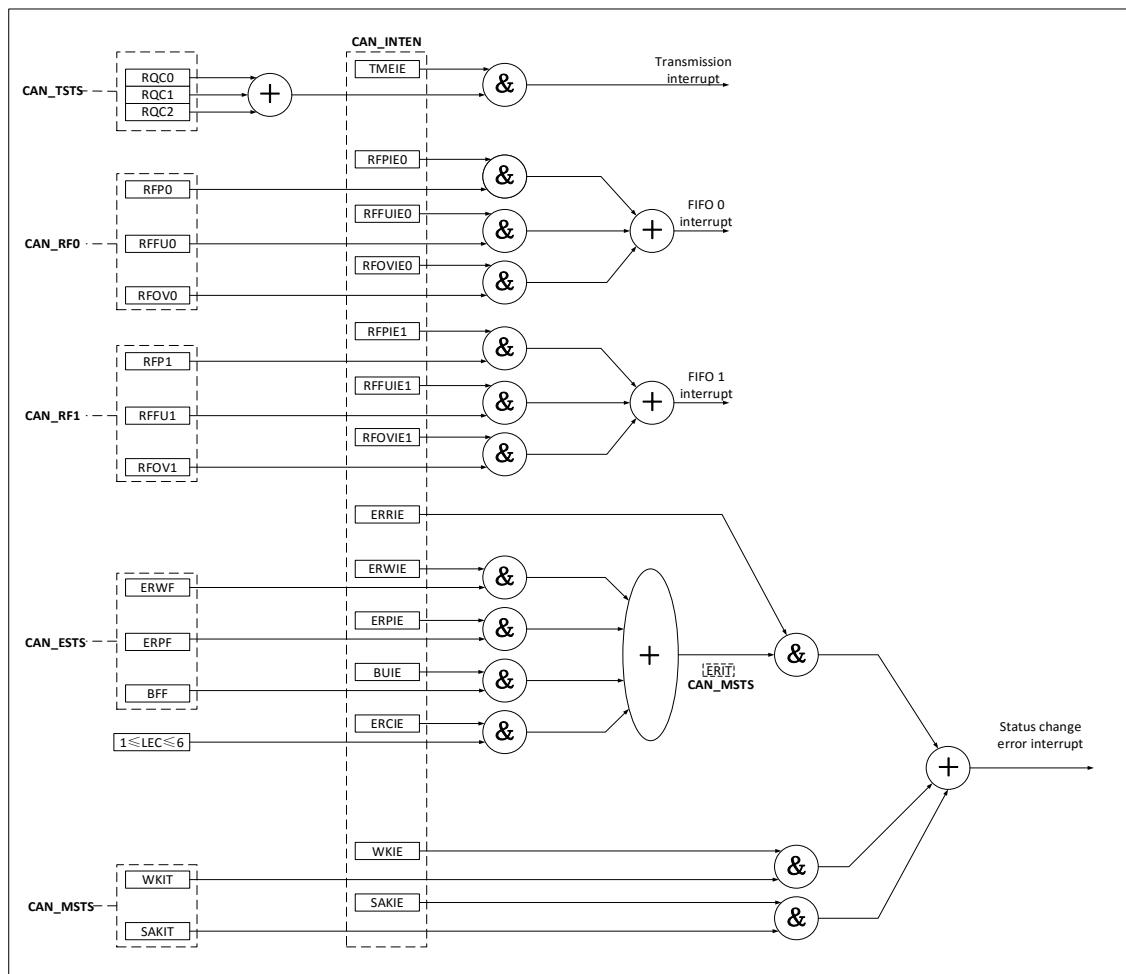
Figure 18-14 Various CAN Frames



18.3.12 bxCAN Interrupt

Four interrupt vectors are dedicated to bxCAN. Each interrupt source can be independently enabled or disabled by the CAN interrupt enable register (CAN_INTEN).

Figure 18-15 Event Flag and Interrupt Generation



- Transmit interrupts can be generated by the following events:
 - Transmit mailbox 0 becomes empty, and the RQC0 bit in the CAN_TSTS register is set.
 - Transmit mailbox 1 becomes empty, and the RQC1 bit in the CAN_TSTS register is set.
 - Transmit mailbox 2 becomes empty, and the RQC2 bit in the CAN_TSTS register is set.
- FIFO0 interrupts can be generated by the following events:
 - FIFO0 receives a new message, the RFP0 bit in the CAN_RF0 register is not '00' anymore.
 - FIFO0 full condition, in which the RFFU0 bit in the CAN_RF0 register is set.
 - FIFO0 overrun condition, in which the RFOV0 bit in the CAN_RF0 register is set
- FIFO1 interrupts can be generated by the following events:
 - FIFO1 receives a new message, the RFP1 bit in the CAN_RF1 register is not '00' anymore.
 - FIFO1 full condition, in which the RFFU1 bit in the CAN_RF1 register is set.
 - FIFO1 overrun condition, in which the RFOV1 bit in the CAN_RF1 register is set
- Error and status change interrupts can be generated by the following events:
 - Error condition; Please refer to "CAN Error Status Register (CAN_ESTS)" for more details about error conditions.
 - Wakeup condition; the start of frame bit (SOF) is monitored on the CAN Rx pin.
 - CAN enters Sleep mode.

18.4 CAN Registers

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

Table 18-3 CAN Register Map and Reset Values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	
000h	MCTRL																									
	Reset Value																									
004h	MSTS																									
	Reset Value																									
008h	TSTS		LPM		TSME		NTM		ARQ2		Reserve d		TER2		ALS2		TOK2		RQC2		ARQ1		Reserve d		TX	
	Reset Value	0	0	0	1	1	1	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
00Ch	RF0																									
	Reset Value																									
010h	FR1																									
	Reset Value																									
014h	INTEN																									
	Reset Value																									
Offset	register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	0	SAKIE	15	0	ERIE	14	0	ERC1E	11	0
018h	ESTS					REC[7:0]																				
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01Ch	BTMG	SIL	LBK		Reserved	SJW		Reserved			BS2		BS1													
	Reset Value	0	0			0	1	0	1	0	0	0	0	1	1											
020h~17Fh																										
180h	TMI0						SID/EID[28:18]																			
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
184h	TDT0							TS																		
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
188h	TDL0					D3			D2				D1											D0		
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	
18Ch	TDH0					D7			D6			D5												D4		
	Reset Value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	

18.4.1 Register Access Protection

Erroneous access to certain registers can cause a CAN note's temporarily disturbance to the whole CAN network. Therefore, the CAN_BTMG register can be modified by software only while the CAN is in Initialization mode.

Although the transmission of incorrect data will not cause problems at the CAN network level, it can severely disturb the application. Hence, a transmit mailbox can be modified by software only while it is in empty state.

The filter values can be modified only when the associated filter banks are deactivated or after setting the FINIT bit. Moreover, the modification of the filter configuration (that is, to modify the CAN_FM1, CAN_FS1, and CAN_FFA1 registers) can only be done when the filter initialization mode is set (FINT = 1).

18.4.2 CAN Control and Status Register

18.4.2.1 CAN Main Control Register (CAN_MCTRL)

Address offset: 0x00

Reset value: 0x0001_0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															DBF
res															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST	Reserved					TTC	ABO	AWU	NART	RFL	TFP	SLP	INRQ		

Bit 31:17	Reserved. Forced to be '0' by hardware.
Bit 16	DBF: Debug freeze 0: CAN works normally during debug. 1: CAN reception/transmission is frozen during debug. Reception FIFOs can still be accessed/controlled normally.
Bit 15	RST: bxCAN software reset 0: This peripheral is in normal operation. 1: bxCAN is forced to reset. After reset, it enters Sleep mode (the FMP bit and the CAN_MCTRL register are initialized to their reset values). Then, this bit is cleared by hardware automatically.
Bit 14:8	Reserved. Forced to be '0' by hardware.

Bit 7	TTC: Time-triggered communication mode 0: Time-triggered communication mode is disabled. 1: Time-triggered communication mode is enabled.
Bit 6	ABO: Automatic bus-off management This bit determines the conditions under which the CAN hardware exits the bus-off state. 0: Software makes requests to leave bus-off state. First, the software sets and clears the INRQ bit of the CAN_MCTRL register. Once 128 occurrences of 11 recessive bits are monitored, it leaves bus-off state. 1: Hardware automatically leaves bus-off state once 128 occurrences of 11 recessive bits are monitored.
Bit 5	AWU: Automatic wakeup mode This bit determines whether hardware or software wakes up CAN from Sleep mode. 0: Software wakeup. Leaving Sleep mode after the software cleans the SLP bit in the CAN_MCTRL register. 1: Hardware automatic wakeup. Leaving Sleep mode once the hardware detects CAN message. During wakeup, hardware automatically clears the SLP and SAK bits in the CAN_MSTS register.
Bit 4	NART: No automatic retransmission 0: According to the CAN standard, when CAN hardware fails to transmit messages, it will automatically retransmit the message until it succeeds. 1: CAN message will be transmitted only once, regardless of the transmission result (success, error, or arbitration lost).
Bit 3	RFL: Receive FIFO locked mode 0: Receive FIFO is not locked on overrun. Once a received FIFO message is not read, the next incoming message will overwrite the previous one. 1: Receive FIFO is locked against overrun. Once a received FIFO message is not read, the next incoming message will be discarded.
Bit 2	TFP: Transmit FIFO priority This bit controls the transmission priority when several mailboxes are pending at the same time. 0: Priority is determined by the identifier of the message. 1: Priority is determined by the request order.
Bit 1	SLP: Sleep mode request This bit is set by software to request CAN to enter Sleep mode. As soon as the current CAN activity (transmission or reception of messages) is completed, CAN enters Sleep mode. This bit is cleared by software to make CAN exit Sleep mode. This bit is cleared by hardware when the AWU bit is set, and a SOF bit is detected on the CANRx signal. This bit is set after reset; that is, CAN is in Sleep mode after reset.
Bit 0	INRQ: Initialization request The software clears this bit to switch CAN from Initialization mode into Normal mode: Once CAN monitors 11 consecutive recessive bits on the Rx signal, the CAN hardware is synchronized and ready for transmission and reception. Hence, hardware signals this event by clearing the IAK bit in the CAN_MSTS register. The software sets this bit to request CAN to enter Initialization mode from Normal mode: Once the current CAN activity (transmission or reception) is completed, CAN enters Initialization mode. Hardware signals this event by setting the IAK bit in the CAN_MSTS register.

18.4.2.2 CAN Main Status Register (CAN_MSTS)

Address offset: 0x04

Reset value: 0x0000 0C02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
res																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved	RXS	LSAP	RX	TX	Reserved	SAKIT	WKIT	ERIT	SAK	IAK	res	rc w1	rc w1	rc w1	r	r

Bit 31:12	Reserved. Forced to be '0' by hardware.
Bit 11	RXS: CANRx signal This bit indicates the actual level of the CAN Rx pin (CAN_RX).
Bit 10	LSAP: Last sample point Last sample value on the CAN Rx pin (corresponding to current Rx bit value).
Bit 9	RX: Receive mode When this bit is set, CAN is the current receiver.
Bit 8	TX: Transmit mode When this bit is set, CAN is the current transmitter.
Bit 7:5	Reserved. Forced to be '0' by hardware.
Bit 4	SAKIT: Sleep acknowledge interrupt When SAKIE = 1, this bit is set by hardware to signal that CAN enters Sleep mode, and then the corresponding interrupts will be generated. When set, this bit generates a status change interrupt if the SAKIE bit in the CAN_INTEN register is set. This bit is cleared by software, or by hardware when SAK is cleared. <i>Note: When SAKIE = 0, no polling on this bit is possible; instead, the SAK bit is polled to learn the Sleep status.</i>
Bit 3	WKIT: Wakeup interrupt When CAN is in Sleep mode, this bit is set by hardware to signal that the start of frame bit (SOF) is detected; a status change interrupt is generated if the WKIE bit in the CAN_INTEN register is set. This bit is cleared by software.
Bit 2	ERIT: Error interrupt On error detection, certain bits in the CAN_ESTS register are set; if the corresponding interrupt enable bit in the CAN_INTEN register is also set, then hardware sets this bit; a status change interrupt is generated if the ERRIE bit in the CAN_INTEN register is set. This bit is cleared by software.
Bit 1	SAK: Sleep mode acknowledge This bit is set by hardware to indicate that CAN hardware is now in Sleep mode. This bit acknowledges the Sleep mode request from the software (set the SLP bit in CAN_MCTRL register). This bit is cleared by hardware when the CAN hardware exits Sleep mode (should be synchronized with CAN bus). The synchronization with CAN bus means that the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal. <i>Note: The process of leaving Sleep mode is triggered when the SLP bit in the CAN_MCTRL register is cleared. For detailed information about clearing the SLP bit, please refer to the description of the AWU bit in the CAN_MCTRL register.</i>
Bit 0	IAK: Initialization acknowledge This bit is set by hardware to indicate that the CAN hardware is now in Initialization mode. This bit acknowledges the initialization request from the software (should be synchronized with CAN bus). The synchronization with CAN bus means that the hardware has to monitor a sequence of 11 consecutive recessive bits on the CAN RX signal.

18.4.2.3 CAN Tx Status Register (CAN_TSTS)

Address offset: 0x08

Reset value: 0x1C00 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LPM2	LPM1	LPM0	TSME2	TSME1	TSME0	NTM[1:0]	ARQ2	Reserved	TER2	ALS2	TOK2	RQC2			
r	r	r	r	r	r	r	r	rs	res	rc w1					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARQ1	Reserved	TER1	ALS1	TOK1	RQC1	ARQ0	Reserved	TER0	ALS0	TOK0	RQC0				
rs	res	rc w1	rc w1	rc w1	rc w1	rs	res	rc w1	rc w1	rc w1	rc w1				

Bit 31	LPM2: Lowest priority flag for mailbox 2 This bit is set by hardware when more than one mailbox are pending for transmission, and mailbox 2 has the lowest priority.
Bit 30	LPM1: Lowest priority flag for mailbox 1 This bit is set by hardware when more than one mailbox are pending for transmission, and mailbox 1 has the lowest priority.
Bit 29	LPM0: Lowest priority flag for mailbox 0 This bit is set by hardware when more than one mailbox are pending for transmission, and mailbox 0 has the lowest priority. <i>Note: The LPM[2:0] bits are cleared when only one mailbox is pending.</i>
Bit 28	TSME2: Transmit mailbox 2 empty This bit is set by hardware when there is no pending transmit request in mailbox 2.
Bit 27	TSME1: Transmit mailbox 1 empty This bit is set by hardware when there is no pending transmit request in mailbox 1.
Bit 26	TSME0: Transmit mailbox 0 empty This bit is set by hardware when there is no pending transmit request in mailbox 0.
Bit 25:24	NTM[1:0]: Mailbox code When at least one transmit mailbox is empty, the two bits indicates the number of the next empty transmit mailbox When all transmit mailbox are empty, the two bits indicates the number of the transmit mailbox with the lowest priority.
Bit 23	ARQ2: Abort request for mailbox 2 Set by software to abort the transmission request of mailbox 2. Cleared by hardware when the mailbox 2 becomes empty. Setting this bit has no effect when the mailbox 2 is not pending for transmission.
Bit 22:20	Reserved. Forced to be '0' by hardware.
Bit 19	TER2: Transmission error of mailbox 2 This bit is set when transmission failure is caused by an error in mailbox 2.
Bit 18	ALS2: Arbitration lost for mailbox 2 This bit is set when transmission failure is caused by an arbitration lost in mailbox 2.
Bit 17	TOK2: Transmission OK of mailbox 2 Hardware updates this bit after each transmission attempt in mailbox 2: 0: The previous transmission fails. 1: The previous transmission succeeds. This bit is set by hardware when the transmission request of mailbox 2 is completed successfully.

Bit 16	RQC2: Request completed mailbox 2 Set by hardware when the last request (transmit or abort) to mailbox 2 is completed. Cleared by software writing '1' or by hardware on transmission request (TRQ set in the CAN_TMI2 register). Clearing this bit also clears other status bits (TOK2, ALS2, and TER2) of mailbox 2.
Bit 15	ARQ1: Abort request for mailbox 1 Set by software to abort the transmission request of mailbox 1. Cleared by hardware when the mailbox 1 becomes empty. Setting this bit has no effect when the mailbox 1 is not pending for transmission.
Bit 14:12	Reserved. Forced to be '0' by hardware.
Bit 11	TER1: Transmission error of mailbox 1 This bit is set when transmission failure is caused by an error in mailbox 1.
Bit 10	ALS1: Arbitration lost for mailbox 1 This bit is set when transmission failure is caused by an arbitration lost in mailbox 1.
Bit 9	TOK1: Transmission OK of mailbox 1 Hardware updates this bit after each transmission attempt in mailbox 1: 0: The previous transmission fails. 1: The previous transmission succeeds. This bit is set by hardware when the transmission request of mailbox 1 is completed successfully.
Bit 8	RQC1: Request completed mailbox 1 Set by hardware when the last request (transmit or abort) to mailbox 1 is completed. Cleared by software writing '1' or by hardware on transmission request (TRQ set in the CAN_TMI1 register). Clearing this bit also clears other status bits (TOK1, ALS1, and TER1) of mailbox 1.
Bit 7	ARQ0: Abort request for mailbox 0 Set by software to abort the transmission request of mailbox 0. Cleared by hardware when the mailbox 0 becomes empty. Setting this bit has no effect when the mailbox 0 is not pending for transmission.
Bit 6:4	Reserved. Forced to be '0' by hardware.
Bit 3	TER0: Transmission error of mailbox 0 This bit is set when transmission failure is caused by an error in mailbox 0.
Bit 2	ALS0: Arbitration lost for mailbox 0 This bit is set when transmission failure is caused by an arbitration lost in mailbox 0.
Bit 1	TOK0: Transmission OK of mailbox 0 Hardware updates this bit after each transmission attempt in mailbox 0: 0: The previous transmission fails. 0: The previous transmission succeeds. This bit is set by hardware when the transmission request of mailbox 0 is completed successfully.
Bit 0	RQC0: Request completed mailbox 0 Set by hardware when the last request (transmit or abort) to mailbox 0 is completed. Cleared by software writing '1' or by hardware on transmission request (TRQ set in the CAN_TMI0 register). Clearing this bit also clears other status bits (TOK0, ALS0, and TER0) of mailbox 0.

18.4.2.4 CAN Receive FIFO 0 Register (CAN_RF0)

Address offset: 0x0C

Reset value: 0x00

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Reserved								RRF M0	RFO V0	RF FU0	Reserved	RFP0	res	r	r
	res								rs	rc w1	rc w1	res	res	r	r	
Bit 31:6	Reserved. Forced to be '0' by hardware.															
Bit 5	RRFM0: Release receive FIFO 0 output mailbox Set by software to release the FIFO output mailbox. Setting this bit when the receive FIFO is empty has no effect; it will be meaningful only when there is message pending in the FIFO. If there are two or more messages pending in the FIFO, the software has to release the output mailbox to access the next message because of the FIFO features. Cleared by hardware when the output mailbox is released.															
Bit 4	RFOV0: FIFO 0 overrun This bit is set by hardware when a new message is received and passes the filter while the FIFO 0 is full. This bit is cleared by software.															
Bit 3	RFFU0: FIFO 0 full Set by hardware when three messages are stored in the FIFO 0. This bit is cleared by software.															
Bit 2	Reserved. Forced to be '0' by hardware.															
Bit 1:0	RFP0[1:0]: FIFO 0 message pending These bits indicate the number of messages pending in the receive FIFO 0. Each time when a new message is stored into the FIFO 0, RFP0 is added '1' by hardware. Each time when software writes '1' to the RRFM0 bit to release the output mailbox, RFP0 is deducted '1' until it becomes 0.															

18.4.2.5 CAN Receive FIFO 1 Register (CAN_RF1)

Address offset: 0x10

Reset value: 0x00

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
	Reserved																							
	res								res															
Bit 31:6	Reserved. Forced to be '0' by hardware.																							
Bit 5	RRFM1: Release FIFO 1 output mailbox Set by software to release the FIFO output mailbox. Setting this bit when the receive FIFO is empty has no effect; it will be meaningful only when there is message pending in the FIFO. If there are two or more messages pending in the FIFO, the software has to release the output mailbox to access the next message because of the FIFO features. Cleared by hardware when the output mailbox is released.																							
Bit 4	RFOV1: FIFO 1 overrun This bit is set by hardware when a new message is received and passes the filter while the FIFO 1 is full. This bit is cleared by software.																							
Bit 3	RFFU1: FIFO 1 full Set by hardware when three messages are stored in the FIFO 1. This bit is cleared by software.																							

Bit 2	Reserved. Forced to be '0' by hardware.
Bit 1:0	RFP1[1:0]: FIFO 1 message pending These bits indicate the number of messages pending in the receive FIFO 1. Each time when a new message is stored into the FIFO 1, RFP1 is added '1' by hardware. Each time when software writes '1' to the RRFM1 bit to release the output mailbox, RFP1 is deducted '1' until it becomes 0.

18.4.2.6 CAN Interrupts Enable Register (CAN_INTEN)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														SAK IE	WK IE
res														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERIE	Reserved	ER CIE	BU IE	ER PIE	ER WIE	Reserved	RFO VIE1	RFF UIE1	RFP IE1	RFO VIE0	RFFU IE0	RFP IE0	TSM EIE		
rw	res	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:18	Reserved. Forced to be '0' by hardware.
Bit 17	SAKIE: Sleep interrupt enable 0: No interrupt is generated when the SAKIT bit is set. 1: An interrupt is generated when the SAKIT bit is set.
Bit 16	WKIE: Wakeup interrupt enable 0: No interrupt is generated when the WKIT bit is set. 1: An interrupt is generated when the WKIT bit is set.
Bit 15	ERIE: Error interrupt enable 0: No interrupt is generated when error is pending in the CAN_ESTS register. 1: An interrupt is generated when error is pending in the CAN_ESTS register.
Bit 14:12	Reserved. Forced to be '0' by hardware.
Bit 11	ERCIE: Last error code interrupt enable 0: The ERIT bit is not set when ERC[2:0] is set by hardware on error detection. 1: The ERIT bit is set when ERC[2:0] is set by hardware on error detection.
Bit 10	BUIE: Bus-off interrupt enable 0: The ERIT bit is not set when the BFF bit is set. 1: The ERIT bit is set when the BFF bit is set.
Bit 9	ERPIE: Error passive interrupt enable 0: The ERIT bit is not set when the ERPF bit is set. 1: The ERIT bit is set when the ERPF bit is set.
Bit 8	ERWIE: Error warning interrupt enable 0: The ERIT bit is not set when the ERWF bit is set. 1: The ERIT bit is set when the ERWF bit is set.
Bit 7	Reserved. Forced to be '0' by hardware.
Bit 6	RFOVIE1: FIFO 1 overrun interrupt enable 0: No interrupt is generated when the RFOV bit of FIFO 1 is set. 1: An interrupt is generated when the RFOV bit of FIFO 1 is set.

Bit 5	RFFUIE1: FIFO 1 full interrupt enable 0: No interrupt is generated when the RFFU bit of FIFO 1 is set. 1: An interrupt is generated when the RFFU bit of FIFO 1 is set.
Bit 4	RFPIE1: FIFO message 1 pending interrupt enable 0: No interrupt is generated when the RFP[1:0] bit of FIFO 1 are not 00b. 1: An interrupt is generated when the RFP[1:0] bit of FIFO 1 are not 00b.
Bit 3	RFOVIE0: FIFO 0 overrun interrupt enable 0: No interrupt is generated when the RFOV bit of FIFO 0 is set. 1: An interrupt is generated when the RFOV bit of FIFO 0 is set.
Bit 2	RFFUIE0: FIFO 0 full interrupt enable 0: No interrupt is generated when the RFFU bit of FIFO 0 is set. 1: An interrupt is generated when the RFFU bit of FIFO 0 is set.
Bit 1	RFPIE0: FIFO 0 message pending interrupt enable 0: No interrupt is generated when the RFP[1:0] bit of FIFO 0 are not 00b. 1: An interrupt is generated when the RFP[1:0] bit of FIFO 0 are not 00b.
Bit 0	TSMEIE: Transmit mailbox empty interrupt enable 0: No interrupt is generated when the RQCx bit is set. 1: An interrupt is generated when the RQCx bit is set.

18.4.2.7 CAN Error Status Register (CAN_ESTS)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REC[7:0]								TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								ERC		Reserved		BFF		ER PF	
res								rw	rw	rw	res		r	r	r

Bit 31:24	REC[7:0]: Receive error counter This counter is implemented according to the Rx part of fault mechanism defined in CAN protocol. In case of a reception error, this counter is incremented by 1 or by 8 depending on the error condition specified in the CAN standard. After every successful reception, the counter is decremented by 1, or is configured to 120 if its value is higher than 127. When the counter value exceeds 127, CAN enters error passive state.
Bit 23:16	TEC[7:0]: Least significant 8 bits of the 9-bit transmit error counter Similar to the previous one, this counter is implemented according to the Tx part of fault mechanism defined in CAN protocol.
Bit 15:7	Reserved. Forced to be '0' by hardware.
Bit 6:4	ERC[2:0]: Last error code Set by hardware according to the error condition on detection of CAN bus error. Cleared by hardware after message is correctly transmitted or received. Hardware does not use error code 7; the value of this bit field can be configured by software, so the updates of the code can be monitored. 000: No error 001: Stuff error 010: Form error 011: Acknowledgment (ACK) error 100: Bit recessive error 101: Bit dominant error 110: CRC error 111: Set by software

Bit 3	Reserved. Forced to be '0' by hardware.
Bit 2	BFF: Bus-off flag This bit is set by hardware when entering the bus-off state. When the transmit error counter, TEC, overflows (greater than 255), CAN enters bus-off state.
Bit 1	ERPF: Error passive flag This bit is set by hardware when the number of errors reaches error passive threshold. (The value in the receive error counter or transmit error counter > 127).
Bit 0	ERWF: Error warning flag This bit is set by hardware when number of errors reaches the warning limit. (The value in the receive error counter or transmit error counter ≥ 96).

18.4.2.8 CAN Bit Timing Register (CAN_BTMG)

Address offset: 0x1C

Reset value: 0x0123 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SIL	LBK	Reserved				SJW[1:0]	Reserved	BS2[2:0]				BS1[3:0]			
rw	rw	res				rw	rw	res	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		BRP[11:0]													
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31	SIL: Silent mode (debug) 0: Normal mode 1: Silent mode
Bit 30	LBK: Loopback mode (debug) 0: Loopback mode is disabled. 1: Loopback mode is enabled.
Bit 29:26	Reserved. Forced to be '0' by hardware.
Bit 25:24	SJW[1:0]: Resynchronization jump width To perform the resynchronization, these bits define the maximum number of time quanta that CAN hardware is allowed to lengthen or shorten in each bit. $t_{RJW} = t_{CAN} \times (SJW[1:0] + 1)$
Bit 23	Reserved. Forced to be '0' by hardware.
Bit 22:20	BS2[2:0]: Time segment 2 These bits define the number of time quanta in time segment 2. $t_{BS2} = t_{CAN} \times (BS2[2:0] + 1)$
Bit 19:16	BS1[3:0]: Time segment 1 These bits define the number of time quanta in time segment 1. $t_{BS1} = t_{CAN} \times (BS1[3:0] + 1)$
Bit 15:12	Reserved. Forced to be '0' by hardware.
Bit 11:0	BRP[11:0]: Baud rate prescaler These bits define the length of a time quanta (t_q). $t_q = (BRP[11:0]+1) \times t_{PCLK}$

18.4.3 CAN Mailbox Register

This section describes the registers of the transmit and receive mailboxes. Please refer to [Section 18.3.9](#) for more information about register mapping.

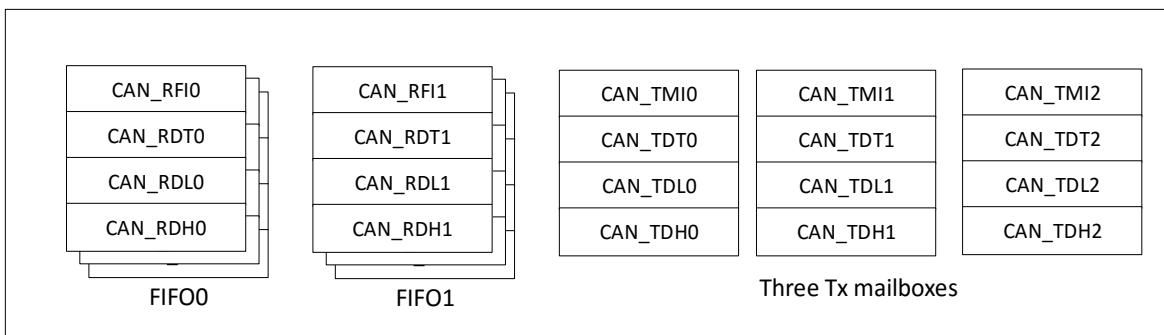
Transmit and receive mailboxes are almost the same except:

- The FID field in the CAN_RDTxR register
- A receive mailbox is read-only.
- A transmit mailbox can be written only when it is empty (the corresponding TSME bit in the CAN_TSTS register is set).

There are three Tx mailboxes and two Rx mailboxes. Each Rx mailbox is a 3-level depth FIFO, and can only access the oldest received message in the FIFO.

Each mailbox consists of four registers.

Figure 18-16 Tx and Rx Mailboxes



18.4.3.1 Tx Mailbox Identifier Register (CAN_TMIx) (x = 0...2)

Address offset: 0x180, 0x190, 0x1A0

Reset value: 0XXXX XXXX, X = Undefined (Except for Bit 0, TRQ = 0 at reset)

Note:

1. This register is write-protected when its mailbox is pending for transmission.
2. This register also implements the Tx request control function (Bit 0) - Reset value is 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
SID[10:0]/EID[28:18]														EID[17:16]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
EID[15:0]														IDT	RTR	TRQ
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
Bit 31:21		SID[10:0]/EID[28:18]: Standard identifier or extended identifier Based on the IDT bit value, these bits are standard identifier or the MSBs of extended identifier.														
Bit 20:3		EID[17:0]: Extended identifier The LSBs of the extended identifier.														
Bit 2		IDT: Identifier extension This bit defines the type of identifier of the mailbox messages. 0: Standard identifier 1: Extended identifier														
Bit 1		RTR: Remote transmission request 0: Data frame 1: Remote frame														
Bit 0		TRQ: Transmit mailbox request Set by software to request the transmission of mailbox data. Cleared by hardware when data transmission is completed, and the mailbox becomes empty.														

18.4.3.2 Mailbox Data Length and Time Stamp Register (CAN_TDTx) (x = 0...2)

All bits of this register are write-protected when the mailbox is not in empty state.

Address offset: 0x184, 0x194, 0x1A4

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16												
TS[15:0]																											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												
Reserved				TMEN		Reserved				DLC																	
res				rw		res				rw		rw		rw													
Bit 31:16		TS[15:0]: Message time stamp This field contains the 16-bit timer value captured at the SOF transmission.																									
Bit 15:9		Reserved																									
Bit 8		TMEN: Transmit global time This bit is active only when CAN is in time-triggered communication mode, i.e. the TTC bit of the CAN_MCTRL register is set. 0: Time stamp TS[15:0] is not sent. 1: Time stamp TS[15:0] is sent. Time stamp, TS[15:0], is the last two data bytes sent in the 8-byte-long message: TS[7:0] is the data byte 7, and TS[15:8] is data byte 8; They replace the data written into CAN_TDhxR[31:16] (D6[7:0] and D7[7:0]). DLC must be programmed as 8 in order to send these two bytes of the time stamp.																									
		Reserved																									
Bit 7:4		Reserved																									
Bit 3:0		DLC[3:0]: Data length code This field defines the length of data bytes or requested by a remote frame. A message can contain from 0 to 8 data bytes, depending on the value in the DLC field.																									

18.4.3.3 Tx Mailbox Data Low Register (CAN_TDLx) (x = 0...2)

All bits of this register are write-protected when the mailbox is not in empty state.

Address offset: 0x188, 0x198, 0x1A8

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D3								D2							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1								D0							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:24		D3[7:0]: Data byte 3 Data byte 3 of the message													
Bit 23:16		D2[7:0]: Data byte 2 Data byte 2 of the message													
Bit 15:8		D1[7:0]: Data byte 1 Data byte 1 of the message													

Bit 7:0	D0[7:0]: Data byte 0 Data byte 0 of the message
---------	--

18.4.3.4 Tx Mailbox Data High Register (CAN_TDhx) (x = 0...2)

All bits of this register are write-protected when the mailbox is not in empty state.

Address offset: 0x18C, 0x19C, 0x1AC

Reset value: Undefined

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D7								D6							
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D5								D4							
rw															

Bit 31:24	D7[7:0] : Data byte 7 Data byte 7 of the message <i>Note: If the TTC bit is set in the CAN_MCTRL register, and its TMEN bit of the mailbox is also set, then D7 and D6 will be replaced by TS, time stamp.</i>
Bit 23:16	D6[7:0] : Data byte 6 Data byte 6 of the message
Bit 15:8	D5[7:0] : Data byte 5 Data byte 5 of the message
Bit 7:0	D4[7:0] : Data byte 4 Data byte 4 of the message

18.4.3.5 Rx FIFO Mailbox Identifier Register (CAN_RFIx) (x = 0...1)

Address offset: 0x1B0, 0x1C0

Reset value: Undefined

Note: All Rx mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SID[10:0]/EID[28:18]												EID[17:16]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EID[15:0]												IDT	RTR	Reserved	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	res

Bit 31:21	SID[10:0]/EID[28:18]: Standard identifier or extended identifier Based on the IDT bit value, these bits are standard identifier or the MSBs of extended identifier.
Bit 20:3	EID[17:0]: Extended identifier The LSBs of the extended identifier.
Bit 2	IDT: Identifier extension This bit defines the type of identifier of the mailbox messages. 0: Standard identifier 1: Extended identifier

Bit 1	RTR: Remote transmission request 0: Data frame 1: Remote frame
Bit 0	Reserved

18.4.3.6 Rx FIFO Mailbox Data Length and Time Stamp Register (CAN_RDTx) (x = 0...1)

Address offset: 0x1B4, 0x1C4

Reset value: Undefined

Note: All Rx mailbox registers are read-only.

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

Bit 31:16	TS[15:0]: Message time stamp This field contains the 16-bit timer value captured at the SOF reception.
Bit 15:8	FID[7:0]: Filter match index This register contains the filter index of message transfer stored in the mailbox. For more details on identifier filtering, please refer to Section 18.3.8.
Bit 7:4	Reserved. Forced to be '0' by hardware.
Bit 3:0	DLC[3:0]: Data length code This field defines the number of data bytes a data frame contains (0 ~ 8). For a remote frame request, DLC is always 0.

18.4.3.7 Rx FIFO Mailbox Data High Register (CAN_RDLx) (x = 0...1)

Address offset: 0x1B8, 0x1C8

Reset value: Undefined

Note: All Rx mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D3								D2							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D1								D0							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31:24	D3[7:0]: Data byte 3 Data byte 3 of the message
Bit 23:16	D2[7:0]: Data byte 2 Data byte 2 of the message
Bit 15:8	D1[7:0]: Data byte 1 Data byte 1 of the message
Bit 7:0	D0[7:0]: Data byte 0 Data byte 0 of the message A message can contain from 0 to 8 data bytes and starts with byte 0.

18.4.3.8 Rx FIFO Mailbox Data High Register (CAN_RDHx) (x = 0...1)

Address offset: 0x1BC, 0x1CC

Reset value: Undefined

Note: All Rx mailbox registers are read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
D7								D6							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
D5								D4							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:24	D7[7:0]: Data byte 7 Data byte 7 of the message														
Bit 23:16	D6[7:0]: Data byte 6 Data byte 6 of the message														
Bit 15:8	D5[7:0]: Data byte 5 Data byte 5 of the message														
Bit 7:0	D4[7:0]: Data byte 4 Data byte 4 of the message														

18.4.4 CAN Filter Register

18.4.4.1 CAN Filter Main Control Register (CAN_FM)

Address offset: 0x200

Reset value: 0x2A1C 0E01

Note: The non-reserved bits in this register are fully controlled by software.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															FINT
Bit 31:1	Reserved. Forced to be reset value.														
Bit 0	FINT : Filter Initialization mode Selecting Initialization mode for all filter banks 0: Filter bank works in Normal mode. 1: Filter bank works in Initialization mode.														

18.4.4.2 CAN Filter Mode Register (CAN_FM1)

Address offset: 0x204

Reset value: 0x0000 0000

Note: This register can only be written when CAN_FM is set (FINT = 1), which puts filters in Initialization mode.

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

Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FMS 13	FMS 12	FMS 11	FMS 10	FM S9	FM S8	FM S7	FM S6	FM S5	FM S4	FM S3	FM S2	FM S1	FM S0	
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:14		Reserved. Forced to be '0' by hardware.													
Bit 13:0		FMSx : Filter mode The operation mode of filter bank x. 0: Two 32-bit registers of filter bank x are in identifier mask mode. 1: Two 32-bit registers of filter bank x are in identifier list mode.													

18.4.4.3 CAN Filter Scale Register (CAN_FS1)

Address offset: 0x20C

Reset value: 0x0000 0000

Note: This register can only be written when CAN_FM is set (FINT = 1), which puts filters in Initialization mode.

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

Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FBS 13	FBS 12	FBS 11	FBS 10	FB S9	FB S8	FB S7	FB S6	FB S5	FB S4	FB S3	FB S2	FB S1	FB S0	
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:14		Reserved. Forced to be '0' by hardware.													
Bit 13:0		FBSx : Filter bit scale configuration Scale configuration of filter banks x (13 ~ 0) 0: Dual 16-bit scale configuration 1: Single 32-bit scale configuration													

18.4.4.4 CAN Filter FIFO Assignment Register (CAN_FFA1)

Address offset: 0x214

Reset value: 0x0000 0000

Note: This register can only be written when CAN_FM is set (FINT = 1), which puts filters in Initialization mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FA F13	FA F12	FA F11	FA F10	FA F9	FA F8	FA F7	FA F6	FA F5	FA F4	FA F3	FA F2	FA F1	FA F0	
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:14	Reserved. Forced to be '0' by hardware.														
Bit 13:0	FAFx: Filter assignment for FIFO configuration The message passing through certain filter will be stored in the assigned FIFO. 0: Filter is assigned to FIFO 0. 1: Filter is assigned to FIFO 1.														

18.4.4.5 CAN Filter Activation Register (CAN_FA1)

Address offset: 0x21C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	FE N13	FE N12	FE N11	FE N10	FE N9	FE N8	FE N7	FE N6	FE N5	FE N4	FE N3	FE N2	FE N1	FE NO	

Bit 31:14	Reserved. Forced to be '0' by hardware.
Bit 13:0	FENx: Filter active Software sets a certain bit to activate the corresponding filter. To modify the corresponding filter registers x (CAN_FBiRx), the FENx bit must be cleared, or the FINT bit of the CAN_FM register must be set. 0: Filter is not activated. 1: Filter is activated.

18.4.4.6 CAN Filter Bank i Register x (CAN_FBiRx) (where i = 0...13; x = 1...2)

Address offset: 0x240h...0x31C

Reset value: Undefined

Note: *The filter register can only be modified when the corresponding FENx bit of the CAN_FA1 register is cleared, or when the FINT bit of the CAN_FM register is set.*

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FD31	FD30	FD29	FD28	FD27	FD26	FD25	FD24	FD23	FD22	FD21	FD20	FD19	FD18	FD17	FD16
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FD15	FD14	FD13	FD12	FD11	FD10	FD9	FD8	FD7	FD6	FD5	FD4	FD3	FD2	FD1	FD0
rw															

In all configurations:

Bit 31:0	FD[31:0]: Filter bits Identifier mode Each bit of the register specifies the level of the corresponding bit of the expected identifier. 0: Dominant bit is expected. 1: Recessive bit is expected. Mask mode Each bit of the register specifies whether the bit of the associated identifier register must match with the corresponding bit of the expected identifier. 0: Don't care. The bit is not used for the comparison. 1: Must match. The bit of the incoming identifier must have the same level specified in the corresponding identifier register of the filter.
----------	--

Note: *According to the different configurations of the scale and mode, the functions of the two registers may differ. Please refer to [Section 18.3.8](#) for more details about filter mapping, function overview, and mask register association.*

19 External Memory Controller (XMC)

19.1 Introduction

The XMC block can interface with synchronous and asynchronous memories and 16-bit PC memory cards. Its main functions are:

- Translating the AHB transactions into the appropriate external device protocol
- Meeting the timing requirements to access the external devices

All external memories share the addresses, data, and control signals with the controller. Each external device is accessed by a unique chip-select signal. XMC performs only one access to an external device at a time.

XMC has the following main functions:

- Devices with static memory-mapped interfaces, including:
 - Static Random Access Memory (SRAM)
 - Read-Only Memory (ROM)
 - NOR Flash memory
 - PSRAM (4 memory banks)
- Two NAND Flash memory banks which support ECC hardware and check up to 8 Kbytes of data
- 16-bit PC Card compatible devices
- Supports burst mode access to synchronous devices, e.g. NOR Flash and PSRAM
- 8-bit or 16-bit wide data bus
- Independent chip-select control for each memory bank
- Independent configuration for each memory bank
- Programmable timings to support a wide range of devices:
 - Programmable wait states (Up to 15 cycles)
 - Programmable bus turnaround cycles (Up to 15 cycles)
 - Programmable output enable and write enable delays (Up to 15 cycles)
 - Independent read and write timings and protocol to support a wide variety of memories and timings
- Write enable and byte lane select outputs used by PSRAM and SRAM devices
- Translation of 32-bit wide AHB transactions into consecutive 16-bit or 8-bit accesses to external 16-bit or 8-bit devices
- A 2-word-long write FIFO, with each word 32-bit wide. It only buffers AHB write burst transactions, which allows to free AHB for other operations when writing to slow memories. Before a new XMC operation, the FIFO should be cleared.

Usually before system reset or power-on, XMC registers that define the external device type and characteristics should be configured, and kept unchanged. Also, users can change the configurations anytime if needed.

19.2 Main Features

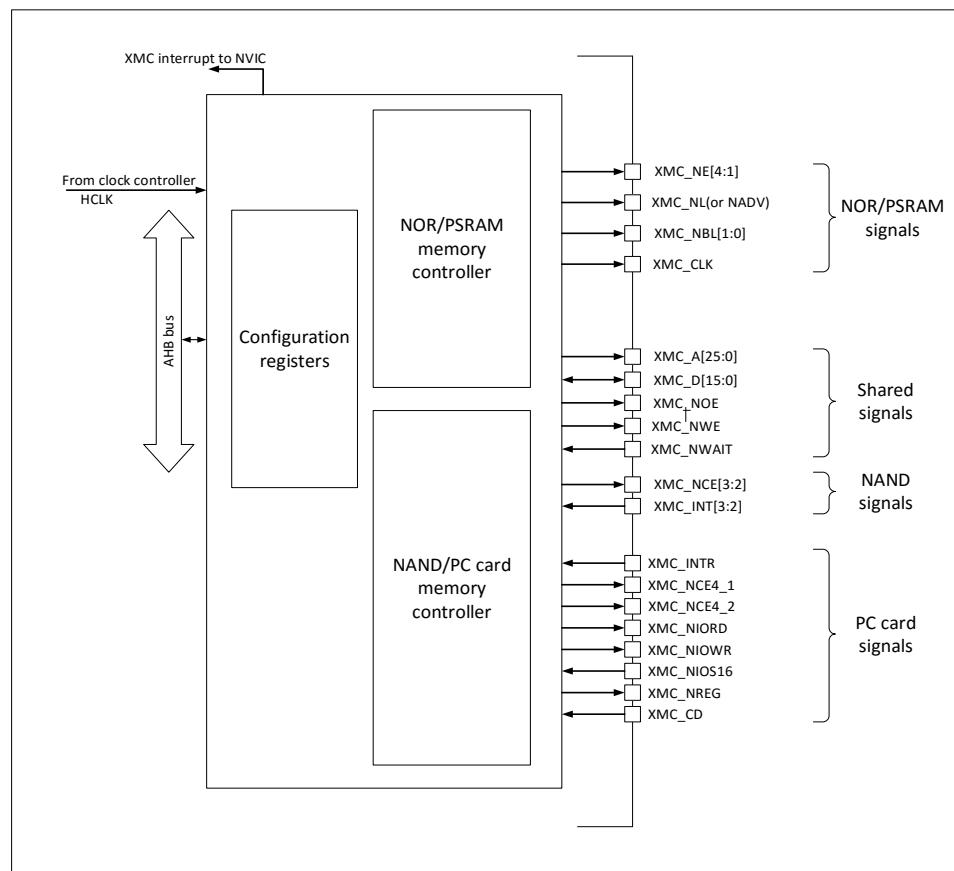
19.2.1 Block Diagram

XMC consists of four main blocks:

- AHB interface (including XMC configuration registers)
- NOR Flash and PSRAM controller
- NAND Flash and PC Card controller
- External device interface

Figure 19-1 is the block diagram of XMC:

Figure 19-1 XMC Block Diagram



19.2.2 AHB Interface

The AHB interface provides channels for internal CPU and other bus peripherals to access the external static memories.

AHB transactions are converted into the external device protocol. If the selected external memory is 16-bit or 8-bit wide, 32-bit wide transactions on the AHB are split into consecutive 16-bit or 8-bit accesses.

AHB clock (HCLK) is the reference clock for XMC.

19.2.3 Supported Memories and Transactions

General transaction rules

Data size of requested AHB transactions can be 8-bit, 16-bit, or 32-bit wide, whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, to ensure the consistency of data transfer, XMC follows the rules below:

- AHB transaction size and memory data size are equal: There is no issue in this case.
- AHB transaction size is greater than the memory size: In this case, XMC splits the AHB transaction into smaller consecutive memory accesses to meet the external data width.
- AHB transaction size is smaller than the memory size:

According to the type of external device, asynchronous transfers may be inconsistent.

- During asynchronous accesses to memories with byte select features (SRAM, ROM, and PSRAM, etc.), XMC performs transactions to access the right data via its byte lanes, NBL[1:0].
- During asynchronous accesses to memories without byte select features (NOR and 16-bit NAND, etc.), a byte access is requested to a 16-bit wide Flash memory; apparently, memories cannot be accessed in byte mode (only 16-bit data transfer is allowed), so:
 - a. Write transactions are not allowed.

- b. Read transactions are allowed. (Controllers read the complete 16-bit memory data and only use bytes needed.)

Configuration registers

XMC can be configured by a register set. [Section 19.4](#) describes the NOR Flash, the PSRAM control register, the NAND Flash, and the PC Card registers in detail.

19.3 Function Overview

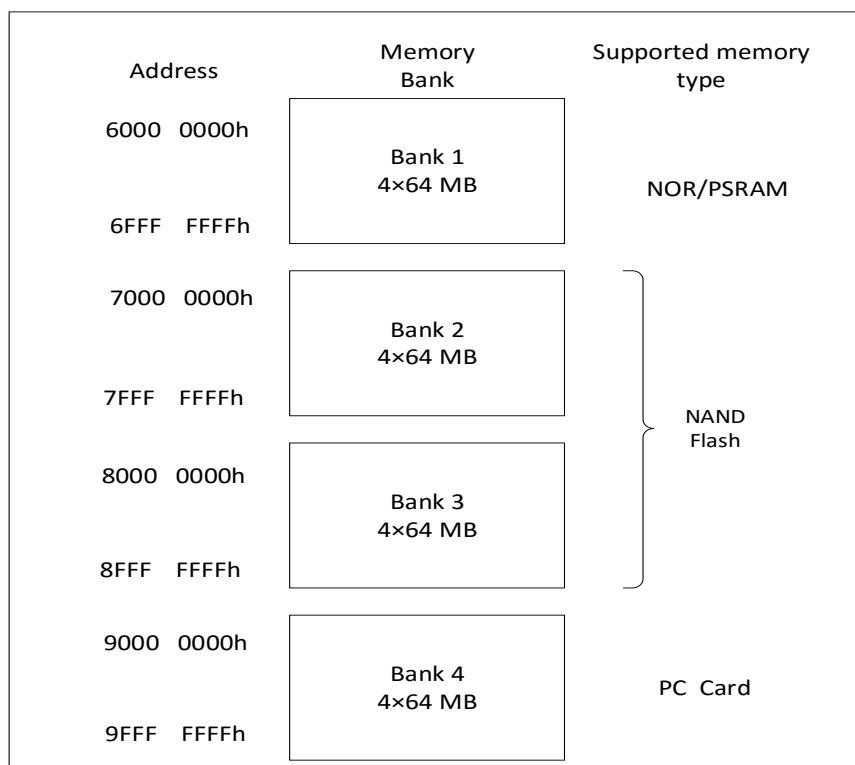
19.3.1 Address Mapping

From XMC point of view, the external memory is divided into 4 fixed-size banks of 256 Mbytes; please refer to Figure 19-2:

- Bank 1 is used to address up to 4 NOR Flash or PSRAM memory devices. This bank is split into 4 NOR/PSRAM sub-banks with 4 dedicated chip selects.
- Banks 2 and Bank 3 are used to address NAND Flash devices (1 NAND Flash device per bank).
- Bank 4 is used to address PC Card device.

The type of memory on each bank is user-defined in the configuration register.

Figure 19-2 XMC Memory Banks



19.3.1.1 NOR and PSRAM Address Mapping

HADDR[27:26] can be used to select one of the four memory banks:

Table 19-1 NOR/PSRAM Memory Bank Selection

HADDR[27:26] (Note)	Selected Bank
00	Bank 1 NOR/PSRAM1
01	Bank 1 NOR/PSRAM2
10	Bank 1 NOR/PSRAM3
11	Bank 1 NOR/PSRAM4

Note: HADDR are internal AHB address lines that need to be converted to external memory.

The HADDR[25:0] bits contain the external memory address. Since HADDR is a byte address, and the memory is addressed in words, the address line connected to the memory varies according to the memory data width, as shown in Table 19-2:

Table 19-2 External Memory Address

Data Width (Note)	Data Address Issued to the Memory	Maximum Memory Capacity (bit)
8-bit	HADDR[25:0] to XMC_A[25:0]	64 Mbyte x 8 = 512 Mbits
16-bit	HADDR[25:1] to XMC_A[24:0], HADDR[0] not issued	64 Mbyte/2 x 16 = 512 Mbits

Note: In case of a 16-bit external memory width, XMC will internally use HADDR[25:1] to generate the address for external memory XMC_A[24:0]. Whatever the external memory width (16-bit or 8-bit) is, XMC_A[0] should always be connected to external memory address A[0].

Wrap support for NOR Flash and PSRAM

Every NOR Flash or PSRAM memory bank can be configured as wrap burst mode.

For memories, the following two situations should be taken into account depending on asynchronous or synchronous access:

- Asynchronous mode: Under this condition, as long as each access has an accurate address, wrap burst mode is fully supported.
- Synchronous mode: Under this condition, XMC only sends one address signal, and then data is transferred in burst mode by CLK order.

Some NOR memories support linear wrap burst mode accesses; data bytes with fixed number can be read from consecutive module N address (Typically, N is 8 or 16, which can be configured through the NOR Flash configuration register.) In this case, wrap burst mode of the memory can be configured as the same mode of AHB.

If wrap burst mode of the memory cannot be configured as the same mode of AHB, the corresponding bits in the XMC configuration register should be set to disable the wrap burst mode, and split the wrap burst accesses into two consecutive accesses.

19.3.1.2 NAND and PC Card Address Mapping

Three memory banks can be used for NAND or PC Card accesses; each bank is separated into the following access space:

Table 19-3 Memory Mapping and Timing Registers

Start Address	End Address	XMC Memory Bank	Memory Space	Timing Register
0x9C000000	0x9FFFFFFF	Bank 4 – PC Card	I/O	XMC_BK4TMGIO (0xB0)
0x98000000	0x9BFFFFFF		Attribute	XMC_BK4TMGATT (0xAC)
0x90000000	0x93FFFFFF		Common	XMC_BK4TMGMEM (0xA8)
0x88000000	0x8BFFFFFF	Bank 3 – NAND Flash	Attribute	XMC_BK3TMGATT (0x8C)
0x80000000	0x83FFFFFF		Common	XMC_BK3TMGMEM (0x88)
0x78000000	0x7BFFFFFF	Bank 2 – NAND Flash	Attribute	XMC_BK2TMGATT (0x6C)
0x70000000	0x73FFFFFF		Common	XMC_BK2TMGMEM (0x68)

For NAND Flash memory, the common and attribute memory spaces are subdivided into three sections (See [Table 19-4](#)) located in the lower 256 Kbytes:

- Data section (The first 64 Kbytes in the common/attribute memory space)
- Command section (The second 64 Kbytes in the common/attribute memory space)

- Address section (The next 128 Kbytes in the common/attribute memory space)

Table 19-4 NAND Bank Selection

Section Name	HADDR[17:16]	Address Range
Address section	1X	0x020000 ~ 0x03FFFF
Command section	01	0x010000 ~ 0x01FFFF
Data section	00	0x000000 ~ 0x00FFFF

The application software uses the 3 sections to access the NAND Flash memory:

- **To send a command to the NAND Flash memory:** The software only needs to write the command value to any address in the command section.
- **To specify the NAND Flash address:** The software only needs to write the address value to any address in the address section. Since a NAND address can be 4-byte or 5-byte long (depending on the actual memory size), several consecutive writes to the address section are needed to specify the full address.
- **To read or write data:** The software only needs to read or write the data value to any address in the data section.

Since the NAND Flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

19.3.2 NOR Flash/PSRAM Controller

XMC can generate appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
 - 8-bit
 - 16-bit
 - 32-bit
- PSRAM (Cellular RAM)
 - Asynchronous mode
 - Burst mode
- NOR Flash
 - Asynchronous mode or Burst mode
 - Multiplexed mode or non-multiplexed mode

XMC outputs a unique chip-select signal, NE[4:1], to each bank. All the other signals (addresses, data, and control) are shared.

For synchronous accesses, XMC issues the clock (CLK) to the selected external device. The frequency of the clock is a submultiple of the HCLK clock. The size of each bank is fixed to 64 Mbytes.

Each bank is configured by dedicated registers (See [Section 19.4](#)).

The programmable memory parameters include access timings (See Table 19-5), whether to support wrap burst data access, and wait management (only for PSRAM and NOR Flash accessed in burst mode).

Table 19-5 Programmable NOR/PSRAM Access Parameters

Parameter	Function	Access Mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	AHB clock cycle (HCLK)	1	16
Address hold	Duration of the address hold phase	Asynchronous and muxed I/Os	AHB clock cycle (HCLK)	2	16
Data setup	Duration of the data setup phase	Asynchronous	AHB clock cycle (HCLK)	2	256
Bus turn	Duration of the bus turnaround phase	Asynchronous or synchronous read	AHB clock cycle (HCLK)	1	16

Clock divide ratio	Ratio between memory access clock cycle (CLK) and AHB clock cycles	Synchronous	AHB clock cycle (HCLK)	2	16
Data latency	Number of clock cycles needed to generate the first data in the burst mode	Synchronous	Memory clock cycle (CLK)	2	17

19.3.2.1 External Memory Interface Signal

[Table 19-6](#), [Table 19-7](#), and [Table 19-8](#) list the signals that are typically used to interface the NOR Flash and PSRAM.

Note: Signals with the prefix “N” are active low.

NOR Flash, non-multiplexed I/Os

Table 19-6 Non-multiplexed I/Os of NOR Flash Interface

XMC Signal Name	Direction	Function
CLK	O	Clock (for synchronous burst mode)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1...4
NOE	O	Output enable
NWE	O	Write enable
NL (= NADV)	O	Latch enable (This signal is called address valid, NADV, by some NOR Flash devices.)
NWAIT	I	NOR Flash signal requesting XMC to wait

NOR Flash memories are addressed by 16-bit words. The maximum capacity is 64 Mbytes (26 address lines).

NOR Flash, multiplexed I/Os

Table 19-7 Multiplexed I/Os of NOR Flash Interface

XMC Signal Name	Direction	Function
CLK	O	Clock (for synchronous burst mode)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NE[x]	O	Chip select, x = 1...4
NOE	O	Output enable
NWE	O	Write enable
NL (= NADV)	O	Latch enable (This signal is called address valid, NADV, by some NOR Flash devices.)
NWAIT	I	NOR Flash signal requesting XMC to wait

NOR Flash memories are addressed by 16-bit words. The maximum capacity is 64 Mbytes (26 address lines).

PSRAM

Table 19-8 Non-multiplexed I/Os of PSRAM Interface

XMC Signal Name	Direction	Function
CLK	O	Clock (for synchronous burst mode)
A[25:0]	O	Address bus
D[15:0]	I/O	Bidirectional data bus
NE[x]	O	Chip select, x = 1..4 (called as NCE by PSRAM (Cellular RAM i.e. CRAM))

NOE	O	Output enable			
NWE	O	Write enable			
NL(= NADV)	O	Address valid (memory signal name: NADV)			
NWAIT	I	PSRAM signal requesting XMC to wait			
NBL[1]	O	Upper byte enable (memory signal name: NUB)			
NBL[0]	O	Lower byte enable (memory signal name: NLB)			

PSRAM memories are addressed by 16-bit words. The maximum capacity is 64 Mbytes (26 address lines).

19.3.2.2 Supported Memories and Transactions

Table 19-9 lists the supported memories, access modes, and transactions; XMC does not support the transactions in gray.

Table 19-9 Supported NOR Flash/PSRAM Memory and Transaction of XMC

Memory	Mode	R/W	AHB Data Size	Memory Data Size	Allowed/Not Allowed	Comment
NOR Flash (Muxed I/Os and Non-muxed I/Os)	Asynchronous	R	8	16	Yes	
	Asynchronous	W	8	16	No	
	Asynchronous	R	16	16	Yes	
	Asynchronous	W	16	16	Yes	
	Asynchronous	R	32	16	Yes	Split into two XMC accesses
	Asynchronous	W	32	16	Yes	Split into two XMC accesses
	Asynchronous page	R	-	16	No	This mode is not supported.
	Synchronous	R	8	16	No	
	Synchronous	R	16	16	Yes	
	Synchronous	R	32	16	Yes	
PSRAM (Muxed I/Os and Non-muxed I/Os)	Asynchronous	R	8	16	Yes	
	Asynchronous	W	8	16	Yes	Use byte signal NBL[1:0]
	Asynchronous	R	16	16	Yes	
	Asynchronous	W	16	16	Yes	
	Asynchronous	R	32	16	Yes	Split into two XMC accesses
	Asynchronous	W	32	16	Yes	Split into two XMC accesses
	Asynchronous page	R	-	16	No	This mode is not supported.
	Synchronous	R	8	16	No	
	Synchronous	R	16	16	Yes	
	Synchronous	R	32	16	Yes	
	Synchronous	W	8	16	Yes	Use byte signal NBL[1:0]
SRAM and ROM	Synchronous	W	16/32	16	Yes	
	Asynchronous	R	8/16/32	16	Yes	Use byte signal NBL[1:0]
	Asynchronous	W	8/16/32	16	Yes	Use byte signal NBL[1:0]

19.3.2.3 Timing Rules

Signals synchronization

- All controller output signals change on the rising edge of the internal clock (HCLK).

- In synchronous mode (PSRAM read or write), all output data changes on the falling edge of memory clock (CLK).

19.3.2.4 NOR Flash and PSRAM Controller Timing Diagram

Asynchronous static memories (NOR Flash, PSRAM, and SRAM)

- All signals are synchronized by the internal clock, HCLK, but this clock is not issued to the memory.
- XMC always samples the data before the NOE signals become invalid. This guarantees that the memory data-hold timing constraint is met (The interval between chip-select invalid and data invalid is usually 0 ns (the min.)).
- If the extended mode is enabled, it is possible to mix mode A, B, C, and D for read and write operations. (For example, read operation can be performed in mode A, and write operation is performed in mode B.)

Mode 1—SRAM/PSRAM (CRAM)

Figure 19-3 Mode 1 Read Access

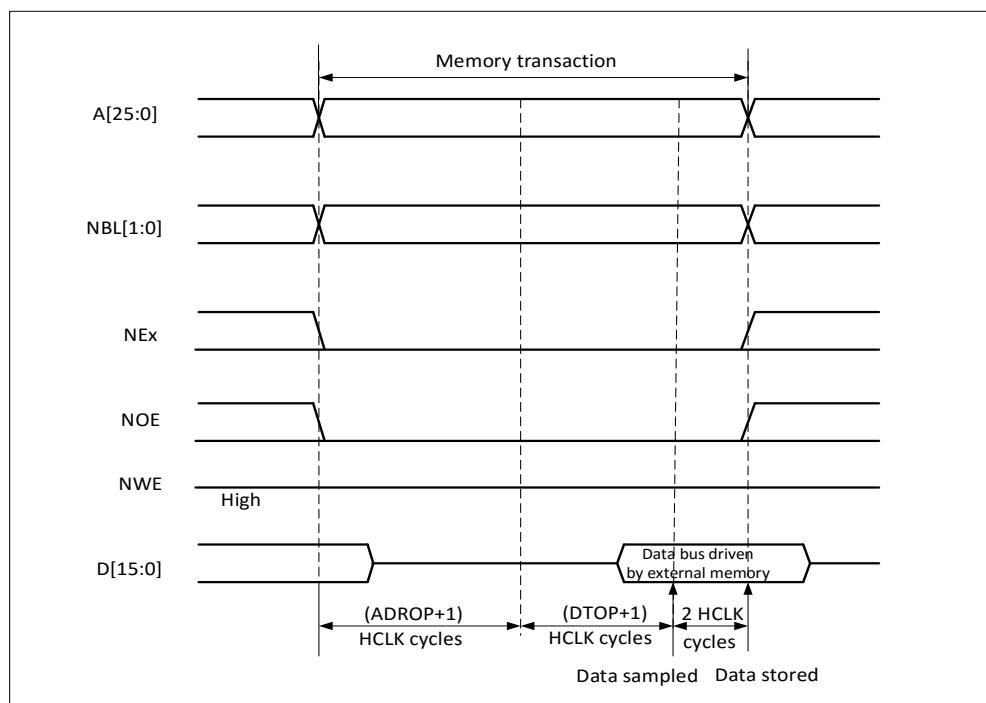
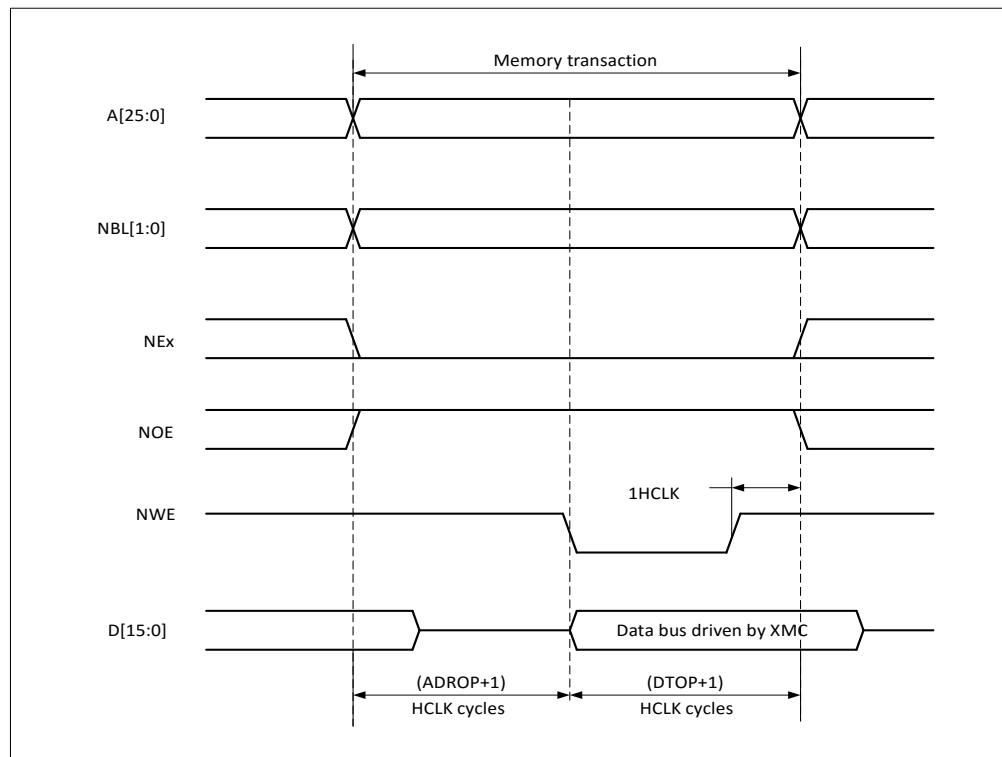


Figure 19-4 Mode 1 Write Access



The last HCLK cycle of write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this HCLK cycle, the DTOP value must be greater than 0 (DTOP > 0).

Table 19-10 XMC_BK1CTRLx Bit Field (Mode 1)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	-
5-4	BUSTYPE	Set as needed
3-2	DEV	Set as needed, excluding 0x2 (NOR Flash)
1	MUXEN	0x0
0	EN	0x1

Table 19-11 XMC_BK1TMGx Bit Field (Mode 1)

Bit Number	Bit Name	Value Set
31-20		0x0000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second access phase, DTOP + 1 HCLK cycle for write accesses, DTOP + 3 HCLK cycles for read accesses This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first access phase (ADROP + 1 HCLK cycle)

Mode A—SRAM/PSRAM (CRAM) OE toggling

Figure 19-5 Mode A Read Access

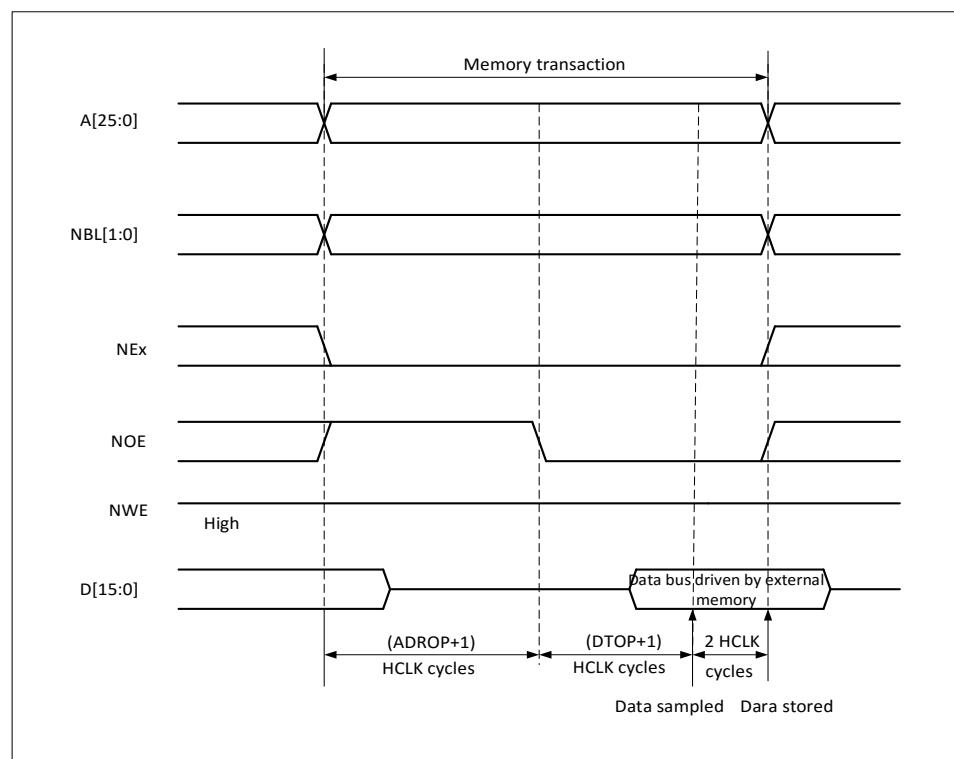
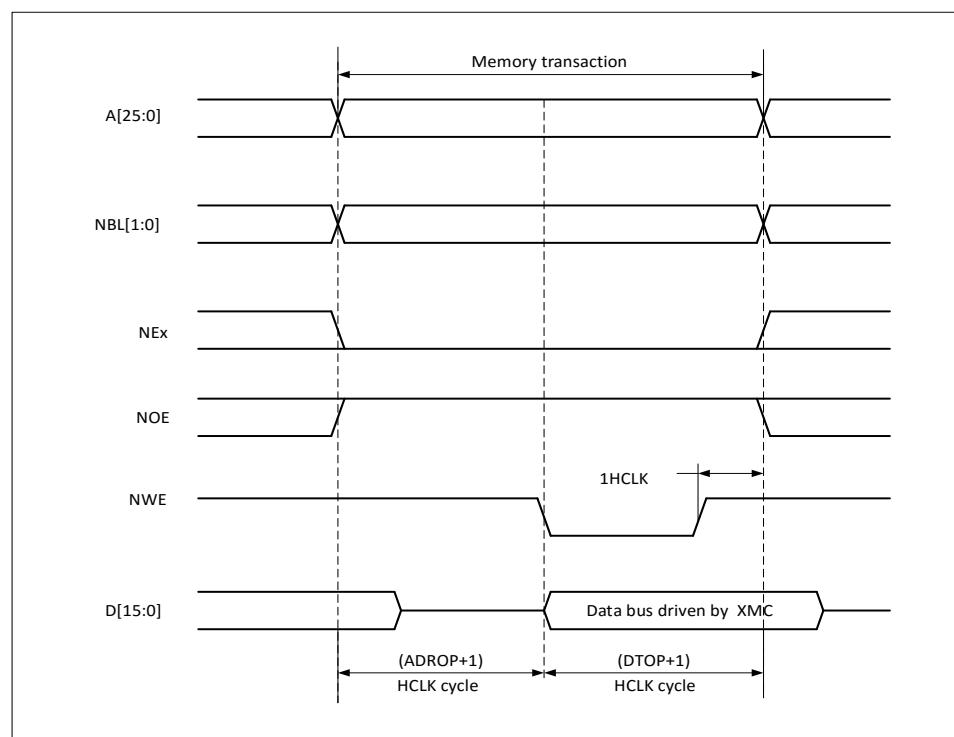


Figure 19-6 Mode A Write Access



The difference between mode A and mode 1 is the toggling of NOE and the independent read/write timings.

Table 19-12 XMC_BK1CTRLx Bit Field (Mode A)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	-
5-4	BUSTYPE	Set as needed
3-2	DEV	Set as needed, excluding 0x2 (NOR Flash)
1	MUXEN	0x0
0	EN	0x1

Table 19-13 XMC_BK1TMGx Bit Field (Mode A)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x0
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second phase of read access (DTOP + 3 HCLK cycles) This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of read access (ADROP + 1 HCLK cycle)

Table 19-14 XMC_BK1TMGWRx Bit Field (Mode A)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x0
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second access phase, DTOP + 1 HCLK cycle for write accesses, DTOP + 3 HCLK cycles for read accesses This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of write access (ADROP + 1 HCLK cycle)

Mode 2/B—NOR Flash

Figure 19-7 Mode 2/B Read Access

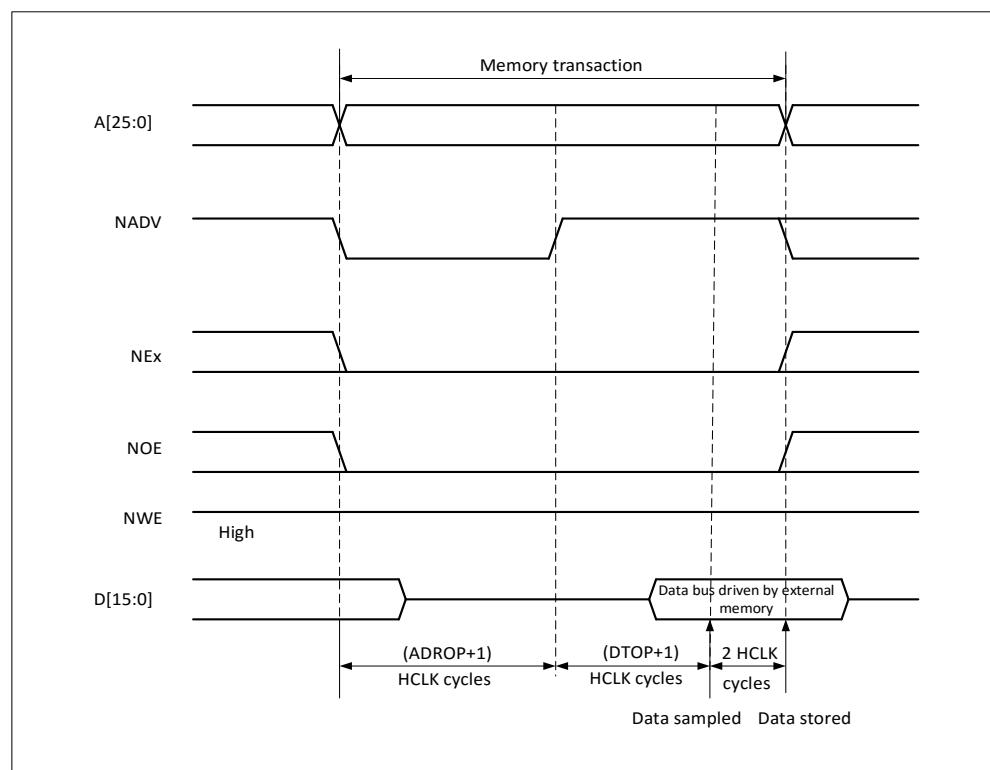


Figure 19-8 Mode 2 Write Access

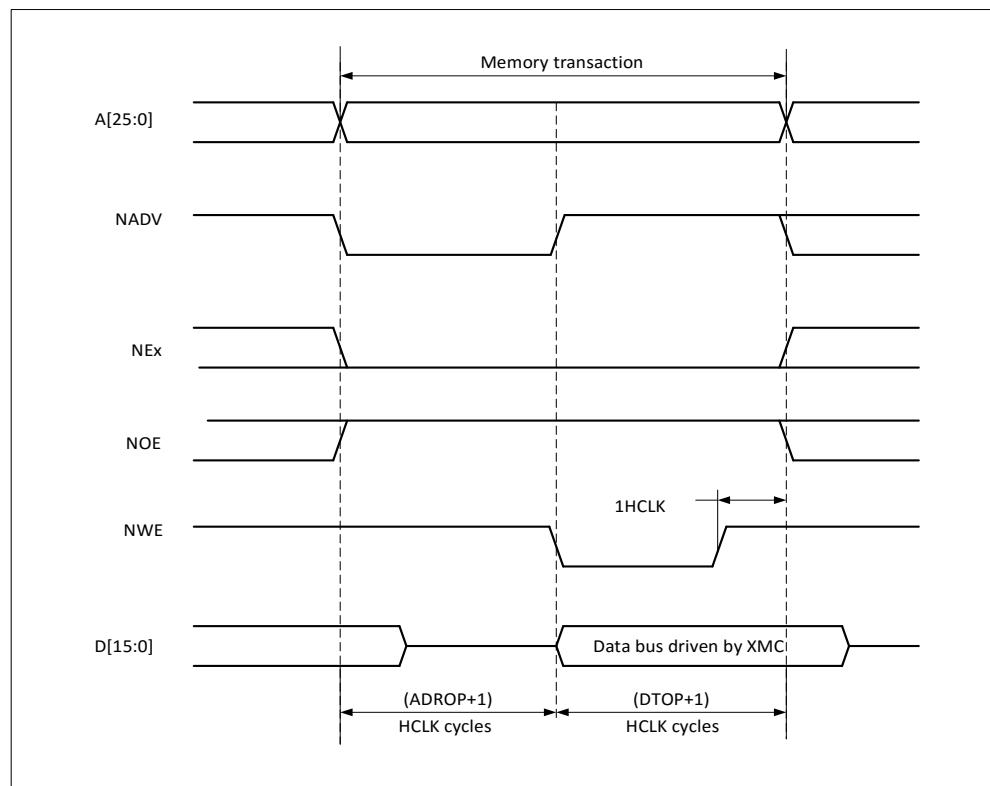
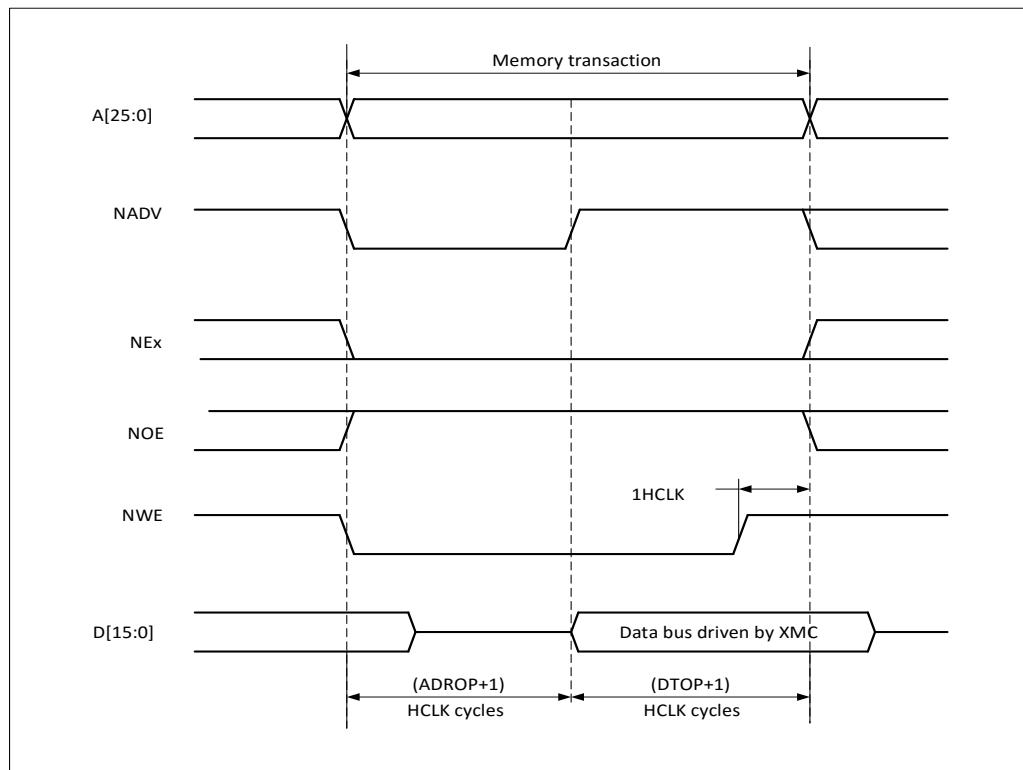


Figure 19-9 Mode B Write Access



The difference between Mode 2/B and Mode A is the NADV toggling, and the independent read/write timings in extended mode (Mode B).

Table 19-15 XMC_BK1CTRLx Bit Field (Mode 2/B)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14	TMGWREN	Mode B: 0x1; Mode 2: 0x0
13-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	Set as needed
3-2	DEV	0x2 (NOR Flash)
1	MUXEN	0x0
0	EN	0x1

Table 19-16 XMC_BK1TMGx Bit Field (Mode 2/B)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x1
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second phase of read access (DTOP + 3 HCLK cycles) This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of read access (ADROP + 1 HCLK cycle)

Table 19-17 XMC_BK1TMGWRx Bit Field (Mode 2/B)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x1
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second access phase, DTOP + 1 HCLK cycle for write accesses, DTOP + 3 HCLK cycles for read accesses This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of write access (ADROP + 1 HCLK cycle)

Note: *XMC_BK1TMGWRx is valid only when extended mode (Mode B) is set; otherwise, this register has no effect.*

Mode C—NOR Flash-OE toggling

Figure 19-10 Mode C Read Access

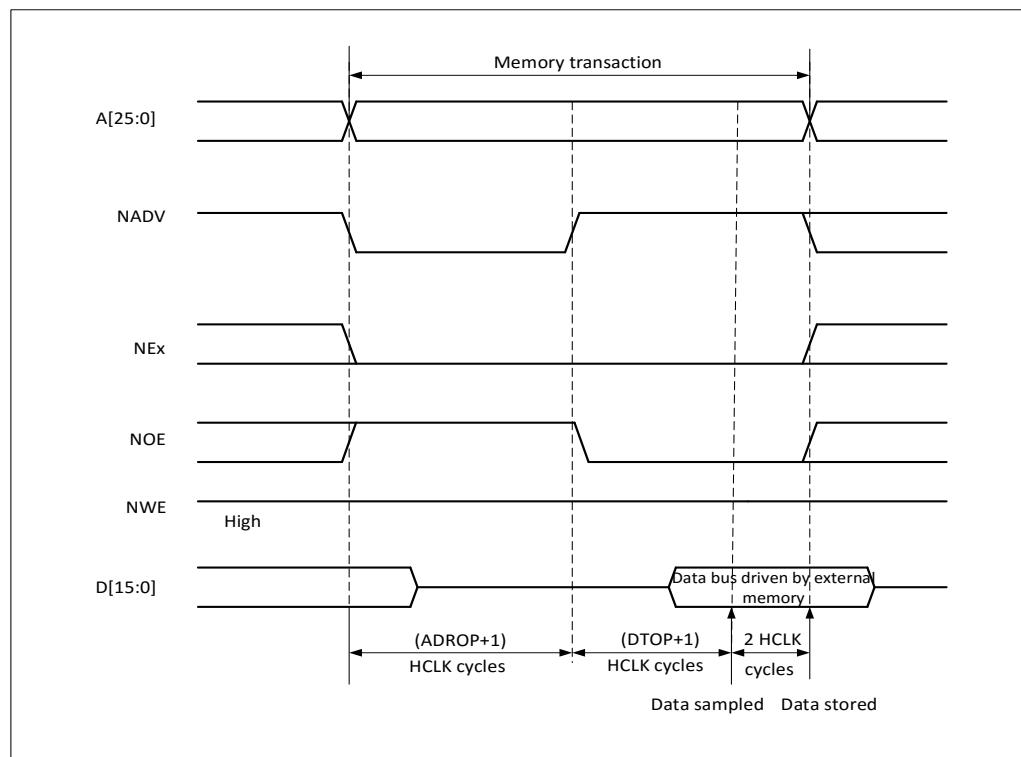
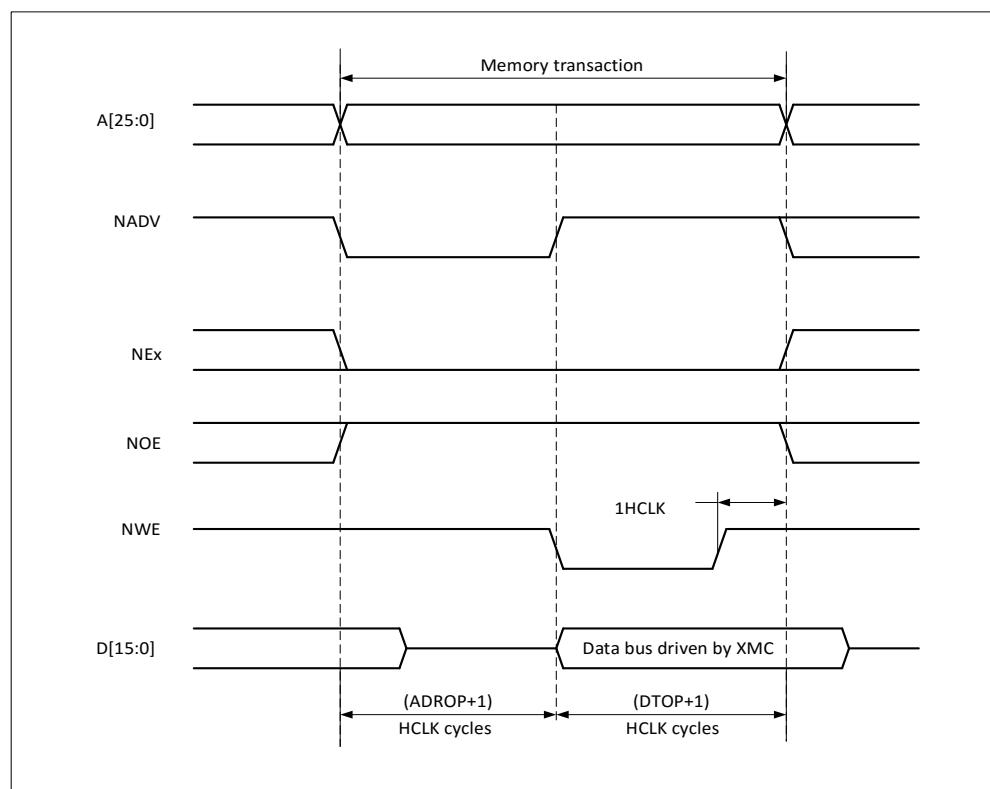


Figure 19-11 Mode C Write Access



The difference between Mode C and Mode1 is the NOE and NADV toggling, and the independent read/write timings.

Table 19-18 XMC_BK1CTRLx Bit Field (Mode C)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	Set as needed
3-2	DEV	0x2 (NOR Flash)
1	MUXEN	0x0
0	EN	0x1

Table 19-19 XMC_BK1TMGx Bit Field (Mode C)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x2
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second phase of read access (DTOP + 3 HCLK cycles) This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of read access (ADROP + 1 HCLK cycle)

Table 19-20 XMC_BK1TMGWRx Bit Field (Mode C)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x2
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second access phase, DTOP + 1 HCLK cycle for write accesses, DTOP + 3 HCLK cycles for read accesses This field cannot be 0, and the minimum is 1.
7-4		0x0
3-0	ADROP	Duration of the first phase of write access (ADROP + 1 HCLK cycle)

Mode D—Asynchronous access with extended address

Figure 19-12 Mode D Read Access

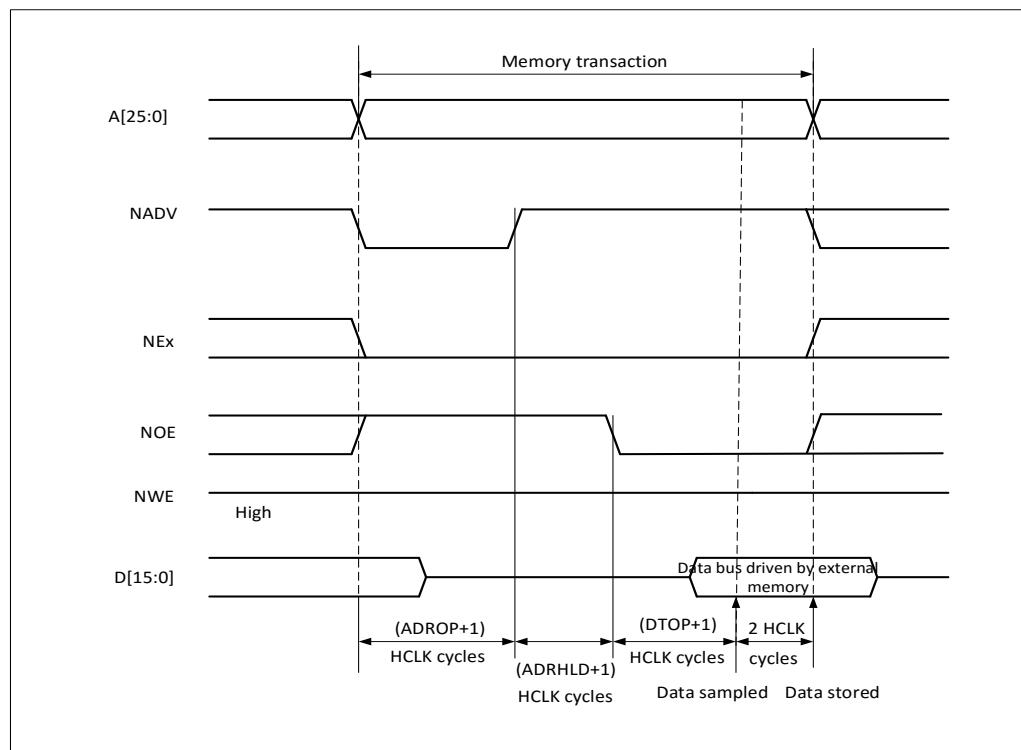
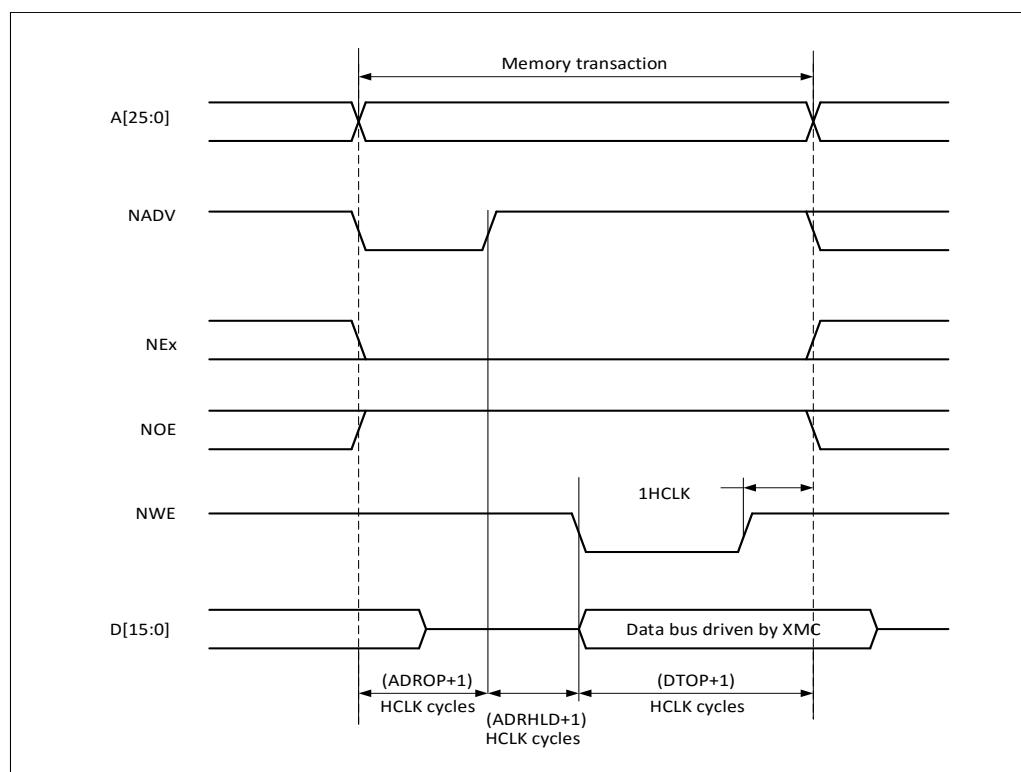


Figure 19-13 Mode D Write Access



The difference between Mode D and Mode 1 is the NADV toggling (the NOE toggling occurs after the NADV toggling) and the independent read/write timings.

Table 19-21 XMC_BK1CTRLx Bit Field (Mode D)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14	TMGWREN	0x1
13-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	Depend on the memory
5-4	BUSTYPE	Set as needed
3-2	DEV	Set as needed
1	MUXEN	0x0
0	EN	0x1

Table 19-22 XMC_BK1TMGx Bit Field (Mode D)

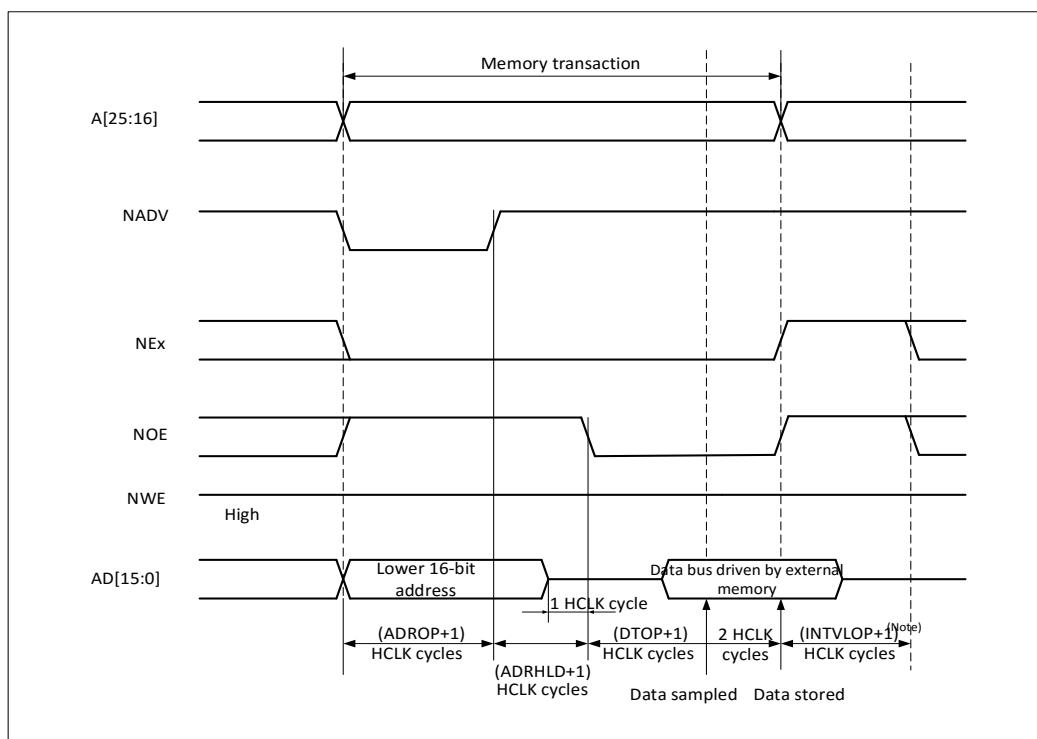
Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x3
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second phase of read access (DTOP + 3 HCLK cycles) This field cannot be 0, and the minimum is 1.
7-4	ADRHL	Duration of the middle phase of read access (ADRHL + 1 HCLK cycle)
3-0	ADROP	Duration of the first phase of read access (ADROP + 1 HCLK cycle)

Table 19-23 XMC_BK1TMGWRx Bit Field (Mode D)

Bit Number	Bit Name	Value Set
31-30		0x0
29-28	MODE	0x3
27-20		0x000
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	Duration of the second phase of write access (DTOP + 1 HCLK cycle) This field cannot be 0, and the minimum is 1.
7-4	ADRHL	Duration of the middle phase of write access (ADRHL + 1 HCLK cycle)
3-0	ADROP	Duration of the first phase of write access (ADROP + 1 HCLK cycle)

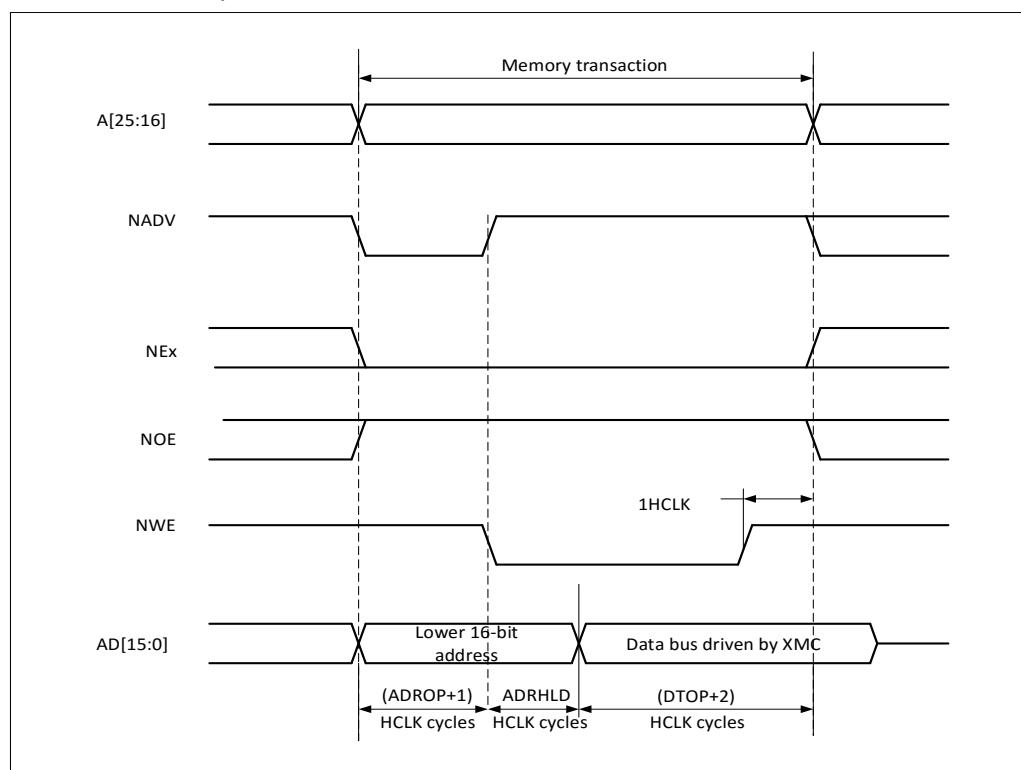
Multiplexed mode—Address/data multiplexed NOR Flash asynchronous access

Figure 19-14 Multiplexed Read Access



Note: The bus turnaround delay ($INTVLOP + 1$) and the internal delay between 2 consecutive read accesses are partially overlapped, so $INTVLOP \leq 5$ will not affect output timing.

Figure 19-15 Multiplexed Write Access



The difference between Multiplexed Mode and Mode D is that the lower 16-bit address occurs on the data bus.

Table 19-24 XMC_BK1CTRLx Bit Field (Multiplexed Mode)

Bit Number	Bit Name	Value Set
31-16		0x0000
15	WAITASYNC	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
14	TMGWREN	0x0
13-10		0x0
9	WAITALV	Meaningful only if the bit 15 is '1'.
8	BURSTEN	0x0
7		-
6	NOREN	0x1
5-4	BUSTYPE	Set as needed
3-2	DEV	0x2 (NOR Flash)
1	MUXEN	0x1
0	EN	0x1

Table 19-25 XMC_BK1TMGx Bit Field (Multiplexed Mode)

Bit Number	Bit Name	Value Set
31-30		0x0
29-20		0x0
19-16	INTVLOP	Duration of the last phase of the access (INTVLOP + 1 HCLK cycle)
15-8	DTOP	Duration of the second access phase, DTOP + 1 HCLK cycle for write accesses, DTOP + 3 HCLK cycles for read accesses This field cannot be 0, and the minimum is 1.
7-4	ADRHL	Duration of the middle phase of the access (ADRHL + 1 HCLK cycle) This field cannot be 0, and the minimum is 1.
3-0	ADROP	Duration of the first access phase (ADROP + 1 HCLK cycle).

19.3.2.5 Synchronous Transactions

According to different values of the parameter, CLKPSC, HCLK cycle is the multiple of memory clock, CLK.

NOR Flash memories specify a minimum time from NADV assertion to CLK high. To meet this constraint, XMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR Flash latency

The data latency is the number of cycles to wait before sampling the data. The DTSTBL value must be consistent with the latency value specified in the NOR Flash configuration register. XMC does not include the clock cycle when NADV is low in the data latency count.

Note: Some NOR Flash memories include the NADV LOW cycle in the data latency count, so the exact relation between the NOR Flash latency and XMC DTSTBL parameter can be either:

- NOR Flash latency = DTSTBL + 2; or
- NOR Flash latency = DTSTBL + 3

Some new memories assert a NWAIT signal at the latency phase. In such cases, DTSTBL can be set to its minimum value. As a result, XMC samples the NWAIT signal and waits long enough until the data are valid. After XMC detects that the memory exits latency phase, correct data are read.

Other memories do not assert NWAIT during latency. In this case, the latency for both XMC and the memory must be set as the same value; otherwise, correct data cannot be obtained, or valid data might be lost at the initial phase of the memory access.

Single-burst transfer

When the selected bank is configured in burst mode for synchronous accesses, if an AHB single-burst transaction is requested with 16-bit data, then XMC performs a burst transaction of length 1; if the AHB transfer is 32-bit, then XMC splits it as two 16-bit transfers, performs a burst transaction of length 2, and then de-assert the chip-select signal when the last data is strobed. When the last data transfer is completed, chip-select signal is withdrawn.

Clearly, such a transfer is not the most efficient (compared to an asynchronous read). Nevertheless, a random asynchronous access will first require to reprogram the memory access mode, which also takes longer time.

Wait management

For synchronous NOR Flash memories, NWAIT should be evaluated after the programmed latency period (DTSTBL + 2 CLK clock cycles).

If NWAIT is detected as active (low level when WAITALV = 0, high level when WAITALV = 1), wait states are inserted by XMC until NWAIT is sensed inactive (high level when WAITALV = 0, low level when WAITALV = 1).

When NWAIT is inactive, the data is considered valid either immediately (WAITCFG = 1) or on the next clock edge (WAITCFG = 0).

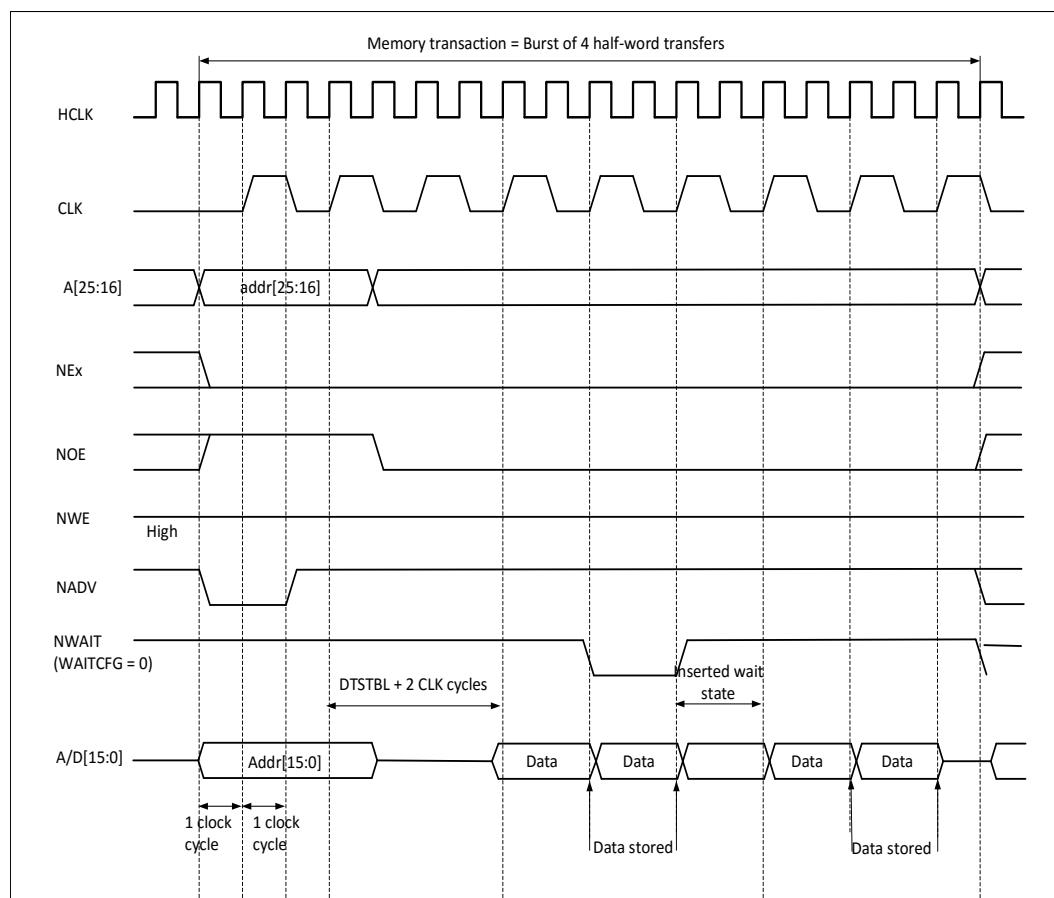
During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the chip-select signal, outputting valid signals, and ignoring invalid data.

There are two timing configurations for the NOR Flash NWAIT signal in burst mode:

- Flash memory asserts the NWAIT signal one data cycle before the wait state (Default after reset).
- Flash memory asserts the NWAIT signal at the wait state.

XMC supports these two NOR Flash wait state configurations on each chip select, by configuring the WAITCFG bit in the XMC_BK1CTRLx registers.

Figure 19-16 Synchronous Multiplexed Read Mode – NOR and PSRAM (CRAM)



Note: The NBL signal is not shown in Figure 19-16; NBL is high for NOR Flash, low for PSRAM (CRAM).

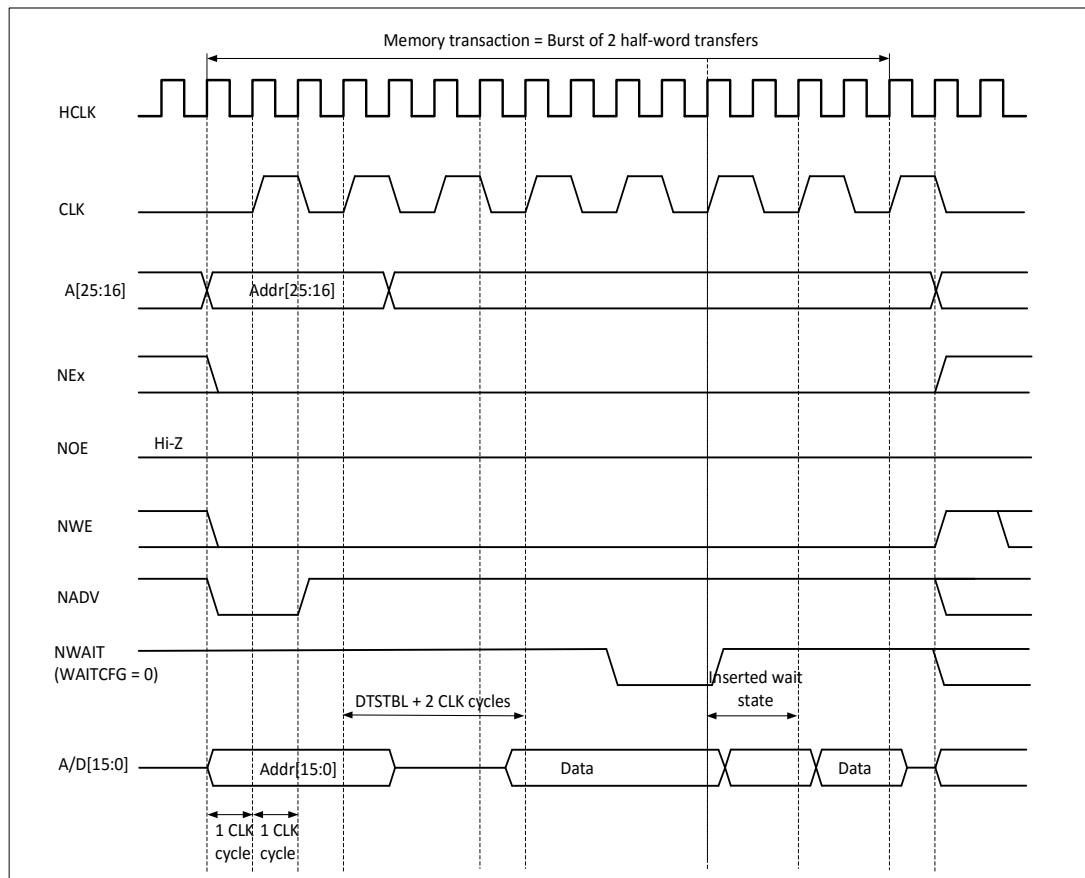
Table 19-26 XMC_BK1CTRLx Bit Field (Synchronous Mode)

Bit Number	Bit Name	Value Set
31-20		0x0000
19	BURSTWRSYN	No effect in synchronous read mode
18-16	PGSIZE	Set as needed (0x1 for CRAM 1.5)
14	TMGWREN	0x0
15		0x0
13	WAITEN	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
12	WREN	No effect in synchronous read mode
11	WAITCFG	Depend on the memory features
10	BRSTSPLTEN	0x0
9	WAITALV	Depend on the memory features
8	BURSTEN	0x1
7		0x1
6	NOREN	Depend on the memory
5-4	BUSTYPE	Set as needed
3-2	DEV	0x1 or 0x2
1	MUXEN	Set as needed
0	EN	0x1

Table 19-27 XMC_BK1TMGx Bit Field (Synchronous Mode)

Bit Number	Bit Name	Value Set
31-28		0x0
27-24	DTSTBL	Data hold time
23-20	CLKPSC	0x0 – Get CLK = HCLK (Not supported) 0x1 – Get CLK = 2xHCLK
19-16	INTVLOP	No effect
15-8	DTOP	No effect
7-4	ADRHL	No effect
3-0	ADROP	No effect

Figure 19-17 Synchronous Multiplexed Write Mode – PSRAM (CRAM)



Note:

1. The memory should issue the NWAIT signal one cycle before, and at the same time WAITCFG should be configured as 0.
2. Byte Lane (NBL) outputs are not shown in Figure 19-17, they are held low while NEx is active.

Table 19-28 XMC_BK1CTRLx Bit Field (Synchronous Mode)

Bit Number	Bit Name	Value Set
31-20		0x0000
19	BURSTWRSYN	0x1
18-16	PGSIZE	Set as needed (0x1 for CRAM 1.5)
15		0x0
14	TMGWREN	0x0
13	WAITEN	Set to 1 if the memory supports this feature. Otherwise, keep at 0.
12	WREN	0x1
11	WAITCFG	0x0
10	BRSTSPLTEN	0x0
9	WAITALV	Depend on the memory features
8	BURSTEN	No effect in synchronous write mode
7		0x1
6	NOREN	Depend on the memory
5-4	BUSTYPE	Set as needed
3-2	DEV	0x1

1	MUXEN	Set as needed
0	EN	0x1

Table 19-29 XMC_BK1TMGx Bit Field (Synchronous Mode)

Bit Number	Bit Name	Value Set
31-28	-	0x0
27-24	DTSTBL	Data hold time
23-20	CLKPSC	0x0 – Get CLK = HCLK (Not supported) 0x1 – Get CLK = 2 x HCLK
19-16	INTVLOP	Time between NEx high to NEx low (INTVLOP HCLK)
15-8	DTOP	No effect
7-4	ADRHL	No effect
3-0	ADROP	No effect

19.3.3 NAND Flash/PC Card Controller

XMC can generate appropriate signal timings to drive the following types of devices:

- NAND Flash
 - 8-bit
 - 16-bit
- 16-bit PC Card compatible devices

The NAND/PC Card controller can control three external banks. Bank 2 and Bank 3 support NAND Flash devices. Bank 4 supports PC Card devices.

Each bank is configured by dedicated registers (Please refer to [Section 19.4](#)). The programmable memory parameters include access timings and ECC configuration.

Table 19-30 Programmable NAND/PC Card Access Parameter

Parameter	Function	Access Mode	Unit	Min.	Max
Memory setup time	Number of clock cycles (HCLK) to set up the address before the command assertion	R/W	AHB clock cycle (HCLK)	1	256
Memory wait time	Minimum duration (HCLK clock cycles) of the command assertion			1	255
Memory hold time	Number of clock cycles (HCLK) to hold the address after the command de-assertion; also the data hold time in case of write access			1	255
Memory databus HiZ time	Number of clock cycles (HCLK) during which the databus is kept in HiZ state after the start of a write access			0	255

19.3.3.1 External Memory Interface Signal

Table 19-31, Table 19-32, and Table 19-33 list the signals that are typically used to interface NAND Flash and PC Card.

Note: When using a PC Card or a CF Card in I/O mode, the NIOS16 input pin must remain at ground level during the whole operation; otherwise, XMC may fail to operate properly. This means that the NIOS16 input pin must NOT be connected to the card, but directly to ground (only 16-bit accesses are allowed).

Prefix “N” indicates that the associated signal is active low.

8-bit NAND Flash

Table 19-31 8-bit NAND Flash

XMC Signal Name	Direction	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal

D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip select, x = 2, 3
NOE (= NRE)	O	Output enable (Memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy signal input to XMC

XMC can generate as many address cycles as needed; theoretically, there is no NAND access capacity limitation.

16-bit NAND Flash

Table 19-32 16-bit NAND Flash

XMC Signal Name	Direction	Function
A[17]	O	NAND Flash address latch enable (ALE) signal
A[16]	O	NAND Flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE[x]	O	Chip select, x = 2, 3
NOE (= NRE)	O	Output enable (Memory signal name: read enable, NRE)
NWE	O	Write enable
NWAIT/INT[3:2]	I	NAND Flash ready/busy signal input to XMC

XMC can generate as many address cycles as needed; theoretically, there is no NAND access capacity limitation.

Table 19-33 16-bit PC Card

XMC Signal Name	Direction	Function
A[10:0]	O	Address bus
NIOS16	I	Data transfer width in I/O space (Only applicable to 16-bit transfer)
NIORD	O	Output enable in I/O space
NIOWR	O	Write enable in I/O space
NREG	O	Register signal indicating if access is in Common or Attribute space
D[15:0]	I/O	Bidirectional databus
NCE4_1	O	Chip select 1
NCE4_2	O	Chip select 2 (indicates if access is 16-bit or 8-bit)
NOE	O	Output enable
NWE	O	Write enable
NWAIT	I	PC Card wait input signal to enter XMC (Memory signal is IORDY)
INTR	I	PC Card interrupt to enter XMC (only for PC Cards that can generate an interrupt)
CD	I	PC Card existing detection signal

19.3.3.2 NAND Flash/PC Card Supported Memories and Transaction

Table 19-34 lists the supported memories, access modes, and transactions; NAND Flash/PC Card Controller do not support the transactions in gray.

Table 19-34 Supported Memories and Transactions

Memory	Mode	R/W	AHB Data Size	Memory Data Size	Allowed/ Not allowed	Comment
8-bit NAND	Asynchronous	R	8	8	Yes	
	Asynchronous	W	8	8	Yes	
	Asynchronous	R	16	8	Yes	Split into two XMC accesses
	Asynchronous	W	16	8	Yes	Split into two XMC accesses
	Asynchronous	R	32	8	Yes	Split into four XMC accesses
	Asynchronous	W	32	8	Yes	Split into four XMC accesses
16-bit NAND	Asynchronous	R	8	16	Yes	
	Asynchronous	W	8	16	No	
	Asynchronous	R	16	16	Yes	
	Asynchronous	W	16	16	Yes	
	Asynchronous	R	32	16	Yes	Split into two XMC accesses
	Asynchronous	W	32	16	Yes	Split into two XMC accesses

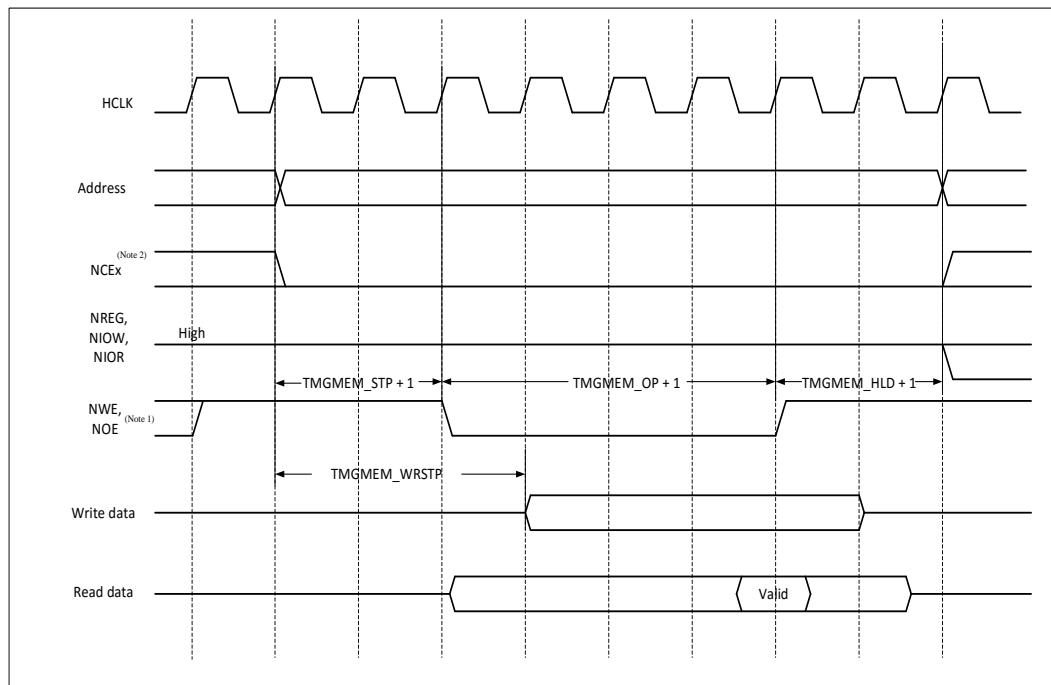
19.3.3.3 NAND Flash, ATA, and PC Card Timing Diagram

Each PC Card/CF Card and NAND Flash memory bank is managed through a set of registers:

- Control register: XMC_BKxCTRL
- Interrupt status register: XMC_BKxSTS
- ECC register: XMC_BKxECC
- Timing register for common memory space: XMC_BKxTMGMEM
- Timing register for attribute memory space: XMC_BKxTMGATT
- Timing register for I/O space: XMC_BK4TMGIO

Each timing configuration register contains three parameters used to define number of HCLK cycles for the three phases of PC Card/CF Card or NAND Flash access, plus one parameter that defines the timing during which XMC starts driving the databus in a write access. Figure 19-18 shows the timing parameter definitions for common memory accesses; in attribute and I/O (only for PC Card) memory space access, timings are similar.

Figure 19-18 NAND/PC Card Controller Timing for Common Memory Access



Note:

1. NOE remains high (*inactive*) during write access. NWE remains high (*inactive*) during read access.
2. NCEx goes *low* as soon as NAND access is requested, and it remains *low* until a different memory bank is accessed.

19.3.3.4 NAND Flash Operation

As mentioned above, the command latch enable (CLE) and address latch enable (ALE) signals of the NAND Flash are driven by address signal lines of XMC. This means that, to send a command or an address to the NAND Flash, CPU has to perform a write access to a certain address in its memory space.

A typical read operation to the NAND Flash device includes the following steps:

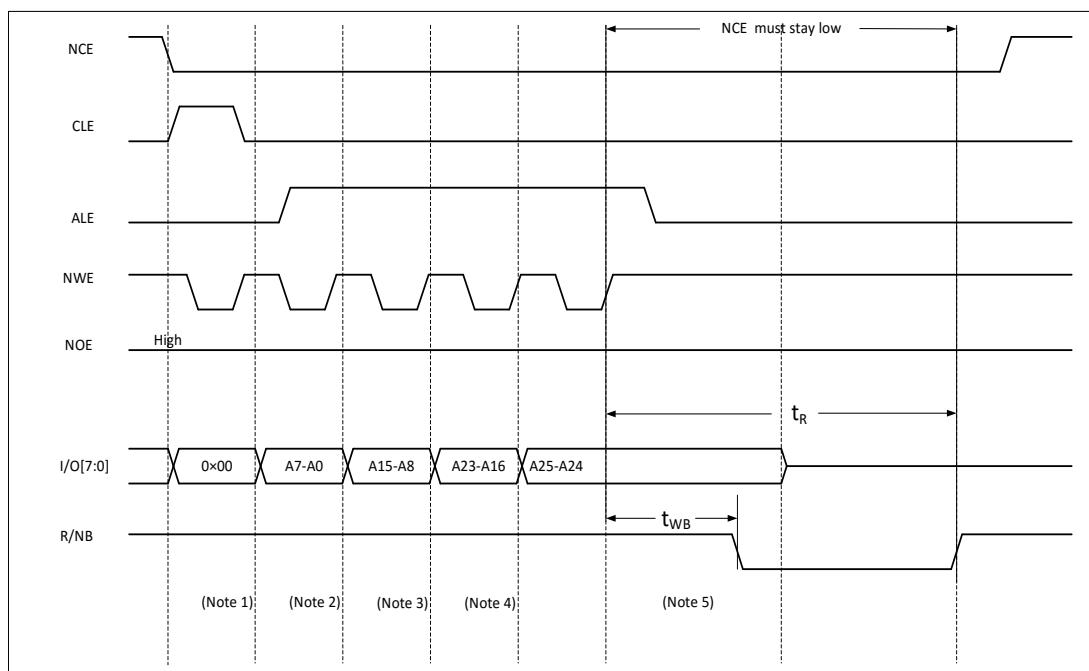
1. According to the NAND Flash characteristics, program and enable the corresponding memory bank by configuring the XMC_BKxCTRL and XMC_BKxTMGMEM registers; for some devices, the XMC_BKxTMGATT register also needs to be programmed (See [Section 19.3.3.5](#)). Bits to be configured include: BUSTYPE for the databus width of the NAND Flash, DEV = 1, and WAITEN = 0 or 1; please refer to the timing configuration in [Section 19.4.2.3](#).
2. CPU performs Flash command byte write in the common memory space (for example, 0x00 for Samsung NAND Flash); during the write strobe (low pulse on NWE), the CLE input of the NAND Flash is active (high level), so the written byte is interpreted as a command by the NAND Flash. Once the command is latched by the NAND Flash, it does not need to be written for the following page read operations.
3. CPU writes four bytes (three for devices with smaller capacity) in the common memory or attribute space as the start address (STARTAD) for a read operation. Take 64 Mb x 8 NAND Flash as an example, the writing sequence is STARTAD[7:0], STARTAD[15:8], STARTAD[23:16], and finally STARTAD[25:24]. During the write strobe (low pulse on NWE), the ALE input of the NAND Flash is active (high level), so the written bytes are interpreted as the start address for read operations. Using the attribute memory space allows XMC to generate different timings, which can be used to implement the prewait function needed by some the NAND Flash (Please refer to [Section 19.3.3.5](#)).
4. Before starting a new access (to the same or another memory bank), the controller waits for the NAND Flash to be ready (R/NB signal becomes high). While waiting, the controller keeps the NCE signal active (low).

5. The CPU can then perform byte read operations in the common memory space to read the NAND Flash page (data field and spare field) byte by byte.
6. Without any CPU command or address write operation, the next NAND Flash page can be read in one of the following three ways:
 - Perform the operation in step 5
 - Restart the operation in step 3 to access a new address
 - Restart the operation in step 2 to access a new command

19.3.3.5 NAND Flash Prewait Functionality

Some NAND Flash devices require that the controller wait for the R/NB signal to go low after writing the last part of the address, as shown in Figure 19-19.

Figure 19-19 Access to Non “CE Don’t Care” NAND Flash



- Note:
1. CPU writes byte 0x00 at address 0x7001 0000.
 2. CPU writes byte A7 ~ A0 at address 0x7002 0000.
 3. CPU writes byte A15 ~ A8 at address 0x7002 0000.
 4. CPU writes byte A23 ~ A16 at address 0x7002 0000.
 5. CPU writes byte A25 ~ A24 at address 0x7802 0000: XMC performs a write access using XMC_BK2TMGATT timing definition, where $TMGATT_HLD \geq 7$ (providing that $(7 + 1) \times HCLK = 112 \text{ ns} > \text{the max. of } t_{WB}$). This guarantees that NCE remains low while R/NB goes from low to high. This requirement only applies for “Non NCE don’t care” NAND Flash.

When this functionality is needed, t_{WB} timing can be guaranteed by programming the TMGMEM_HLD value. However, in any following CPU read/write accesses to the NAND Flash memory, controller will insert a hold delay of $(TMGMEM_HLD + 1)$ HCLK cycles between NWE signal rising edge and the next operation.

To overcome this timing constraint, the attribute memory space can be used by programming TMGATT_HLD value to meet the t_{WB} timing, and leaving the TMGMEM_HLD value at its minimum. Then, CPU must use the common memory space for all NAND Flash read and write accesses, except when writing the last address byte to the NAND Flash device, where CPU must write to the attribute memory space.

19.3.3.6 NAND Flash ECC Calculation

XMC PC Card controller includes two error correction code computation hardware blocks, one for memory bank 2, and the other for memory bank 3. They are used to reduce CPU workload when processing the error correction code by software.

The two identical registers are and associated with bank 2 and bank 3, respectively. No hardware ECC computation is available for memories connected to bank 4.

The error correction code (ECC) algorithm implemented in XMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4096, or 8192 bytes read or written to NAND Flash memory.

The ECC modules monitor the NAND Flash databus and read/write signals (NCE and NWE) each time when the NAND Flash memory card is activated.

The operation of the function is as follows:

- When access to NAND Flash is made in bank 2 or bank 3, the data present on the D[15:0] bus is latched and used for ECC computation.
- When access to NAND Flash occurs at any other address, the ECC logic does not perform any operation. Thus, write operations for defining NAND Flash commands or addresses are not taken into account for ECC computation.

Once the desired number of bytes is read from/written to the NAND Flash, the XMC_BK2/3ECC register must be read by software in order to retrieve the computed ECC value. Once read, the register is cleared by resetting the ECCEN bit to 0. Then, the ECCEN bit must be set to '1' in the XMC_BK2/3CTRL register to enable a new ECC computation.

19.4 XMC Registers

Table 19-35 XMC Registers Address Mapping

Offset	Register Name	Reset Value	Description
000h	XMC_BK1CTRL1	0x000030DB	SRAM/NOR Flash chip-select control register 1
004h	XMC_BK1TMG1	0xFFFFFFFF	SRAM/NOR Flash chip-select timing register 1
008h	XMC_BK1CTRL2	0x000030D2	SRAM/NOR Flash chip-select control register 2
00Ch	XMC_BK1TMG2	0xFFFFFFFF	SRAM/NOR Flash chip-select timing register 2
010h	XMC_BK1CTRL3	0x000030D2	SRAM/NOR Flash chip-select control register 3
014h	XMC_BK1TMG3	0xFFFFFFFF	SRAM/NOR Flash chip-select timing register 3
018h	XMC_BK1CTRL4	0x000030D2	SRAM/NOR Flash chip-select control register 4
01Ch	XMC_BK1TMG4	0xFFFFFFFF	SRAM/NOR Flash chip-select timing register 4
060h	XMC_BK2CTRL	0x00000018	PC Card/NAND Flash control register 2
064h	XMC_BK2STS	0x00000040	FIFO status and interrupt register 2
068h	XMC_BK2TMGMEM	0xFCFCFCFC	Common memory space timing register 2
06Ch	XMC_BK2TMGATT	0xFCFCFCFC	Attribute memory space timing register 2
080h	XMC_BK3CTRL	0x00000018	PC Card/NAND Flash control register 3
084h	XMC_BK3STS	0x00000040	FIFO status and interrupt register 3
088h	XMC_BK3TMGMEM	0xFCFCFCFC	Common memory space timing register 3
08Ch	XMC_BK3TMGATT	0xFCFCFCFC	Attribute memory space timing register 3
0A0h	XMC_BK4CTRL	0x00000018	PC Card/NAND Flash control register 4
0A4h	XMC_BK4STS	0x00000040	FIFO status and interrupt register 4
0A8h	XMC_BK4TMGMEM	0xFCFCFCFC	Common memory space timing register 4
0ACh	XMC_BK4TMGATT	0xFCFCFCFC	Attribute memory space timing register 4
0B0h	XMC_BK4TMGIO	0xFCFCFCFC	I/O memory space timing register 4

104h	XMC_BK1TMGWR1	0xFFFFFFFF	SRAM/NOR Flash write timing register 1
10Ch	XMC_BK1TMGWR2	0xFFFFFFFF	SRAM/NOR Flash write timing register 2
114h	XMC_BK1TMGWR3	0xFFFFFFFF	SRAM/NOR Flash write timing register 3
11Ch	XMC_BK1TMGWR4	0xFFFFFFFF	SRAM/NOR Flash write timing register 4

19.4.1 NOR Flash and PSRAM Controller Register

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

19.4.1.1 SRAM/NOR Flash Chip-select Control Register 1...4 (XMC_BK1CTRL1...4)

Address offset: 0xA0000000 + 8 * (x - 1), x = 1...4

Reset value: 0x000030DB for bank1, and 0x000030D2 from bank2 to bank4

This register contains the control information of each memory bank, which can be used for SRAM, ROM, asynchronous or burst transfer NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														BURSTWRSYN	PGSIZE
res												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAITASYNC	TMGWREN	WAITEN	WREN	WAITCFG	BRSTSPLTEN	WAITALV	BURSTEN	Reserve d	NOREN	BUSTYPE	DEV	MUXEN	EN		
rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

Bit 31:20	Reserved
Bit 19	BURSTWRSYN: Write burst enable For Cellular RAM, this bit enables the synchronous burst protocol during write operations. The enable bit of synchronous burst protocol for read accesses is the BURSTEN bit in the XMC_BK1CTRLx register. 0: Write access is in asynchronous mode. 1: Write access is in synchronous mode.
Bit 18:16	PGSIZE: CRAM page size These are used for Cellular RAM 1.5, which does not allow burst access to cross the address boundaries between pages. When these bits are configured, XMC controller splits to burst access automatically once the memory page size is reached (Please refer to memory datasheet for page size). 000: No burst split when crossing page boundary, which is the default after reset. 001: 128 bytes 010: 256 bytes 011: 512 bytes 100: 1024 bytes Others: Reserved
Bit 15	WAITASYNC: Wait signal during asynchronous transfers This bit enables/disables XMC to use the wait signal, even during an asynchronous protocol. 0: The NWAIT signal is not taken into account when running an asynchronous protocol (default after reset). 1: The NWAIT signal is taken into account when running an asynchronous protocol.
Bit 14	TMGWREN: Extended mode enable This bit enables XMC to program the XMC_BK1TMGWR register, which allows different timings for read and write operations. 0: The XMC_BK1TMGWR register is not used, which is the default state after reset. 1: XMC uses the XMC_BK1TMGWR register.

Bit 13	WAITEN: Wait enable bit This bit enables/disables wait-state insertion via the NWAIT signal in synchronous mode. 0: The NWAIT signal is disabled, and the NWAIT signal wait state insertion will not be detected after the programmed Flash latency period. 1: The NWAIT signal is enabled, and wait states are inserted according to the NWAIT signal after the programmed Flash latency period; this is the default state after reset.
Bit 12	WREN: Write enable bit This bit indicates whether XMC write accesses in the bank are enabled/disabled. 0: XMC write operations in the bank are disabled; otherwise, an AHB error is reported, 1: XMC write operations in the bank are enabled; this is the default state after reset.
Bit 11	WAITCFG: Wait timing configuration When the Flash memory in burst mode, the NWAIT signal indicates whether the data from the memory are valid, or whether a wait state should be inserted. This configuration bit determines if NWAIT is asserted one clock cycle before the wait state or at the wait state: 0: NWAIT signal is active one data cycle before the wait state; the default state after reset. 1: NWAIT signal is active at the wait state (Not applicable for Cellular RAM).
Bit 10	BRSTSPLTEN: Wrapped burst mode support This bit defines whether the controller will split an AHB burst wrap access into two linear accesses. It is valid only when accessing memories in burst mode. 0: Direct wrapped burst is disabled; this is the default state after reset. 1: Direct wrapped burst is enabled.
Bit 9	WAITALV: Wait signal polarity bit This bit defines the polarity of the wait signal from memory; it is valid only when accessing the memory in burst mode: 0: NWAIT signal is active low; this is the default state after reset. 1: NWAIT signal is active high.
Bit 8	BURSTEN: Burst enable bit This bit enables/disables burst mode accesses to the Flash memory. It is valid only for synchronous Flash memories operating in burst mode. 0: Burst mode is disabled; this is the default state after reset. 1: Burst mode is enabled.
Bit 7	Reserved
Bit 6	NOREN: Flash access enable This bit enables NOR Flash memory access operations. 0: NOR Flash memory access is disabled. 1: NOR Flash memory access is enabled.
Bit 5:4	BUSTYPE: Memory databus width This bit defines the external memory device width, and is valid for all types of memories. 00: 8 bits 01: 16 bits (Default after reset) 10: Reserved; do not use. 11: Reserved; do not use.
Bit 3:2	DEV: Memory type The bits define the type of external memory: 00: SRAM, ROM (Default after reset for Bank 2...4) 01: PSRAM (Cellular RAM: CRAM) 10: NOR Flash (Default after reset for Bank 1) 11: Reserved
Bit 1	MUXEN: Address/data multiplexing enable bit When this bit is set, the lower 16 bits of the address/data values are multiplexed on the databus; this bit is valid only with NOR and PSRAM memories: 0: Address/data is not multiplexed. 1: Address/data is multiplexed on databus; this is the default state after reset.

Bit 0	EN: Memory bank enable bit This bit enables the memory bank. After reset, bank 1 is enabled, and all other memories are disabled. Accessing a disabled bank causes an ERROR on AHB bus. 0: The corresponding memory bank is disabled. 1: The corresponding memory bank is enabled.
-------	--

19.4.1.2 SRAM/NOR Flash Chip-select Timing Register 1...4 (XMC_BK1TMG1...4)

Address offset: 0xA0000000 + 0x04 + 8* (x-1), x = 1...4

Reset value: 0xFFFFFFFF

This register contains the control information of each memory bank used for SRAM, ROM, and NOR Flash memories. If the TMGWREN bit is set in the XMC_BK1CTRLx register, then there will be two timing registers available: one to configure read accesses (this register), and the other to configure write accesses (the XMC_BK1TMGWRx register).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		MODE		DTSTBL				CLKPSC				INTVLOP			
res		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTOP						ADRHL				ADROP					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:30	Reserved
Bit 29:28	MODE: Access mode The 2 bits define asynchronous access mode. They only have effects when the TMGWREN bit is set in the XMC_BK1CTRLx register. 00: Access mode A 01: Access mode B 10: Access mode C 11: Access mode D
Bit 27:24	DTSTBL ^(Note) : Data latency (for synchronous burst NOR Flash) For synchronous NOR Flash memory with burst mode enabled, it is required to define the number of memory clock cycles to wait before accessing the first data. This timing parameter is not expressed in HCLK periods, but in Flash clock periods (CLK). In asynchronous NOR Flash, SRAM, or ROM access, this value has no effect. In case of CRAM, this field must be set to 0. 0000: Data latency of 2 CLK clock cycles for first data access 1111: Data latency of 17 CLK clock cycles for first data access (Default value after reset)
Bit 23:20	CLKPSC: Clock divide ratio (for CLK signal) This bit field defines the period of CLK clock output signal, expressed in the number of HCLK cycles: 0000: Reserved 0001: 1 CLK cycle = 2 HCLK cycles 0010: 1 CLK cycle = 3 HCLK cycles 1111: 1 CLK cycle = 16 HCLK cycles (Default value after reset) In asynchronous NOR Flash, SRAM, or ROM accesses, this value has not effect.

Bit 19:16	<p>INTVLOP: Bus turnaround phase duration These bits define the delay time on the bus after a read access (only used for the muxed mode of NOR Flash access). After a read access, the controller needs to issue the address for the next operation on data bus, and this delay is used to avoid bus conflicts. If the extended memory does not include muxed mode memory of the bus, or the slowest memory can restore data bus to HiZ state within 6 HCLK clock periods, this parameter can be set as its minimum.</p> <p>0000: Bus turnaround phase duration = 1 HCLK clock cycle 1111: Bus turnaround phase duration = 16 HCLK clock cycles (Default value after reset)</p>
Bit 15:8	<p>DTOP: Data-phase duration These bits define the duration of the data phase (Please refer to Figure 19-3 to Figure 19-15) used in SRAM, ROM, and asynchronous muxed mode accesses of NOR Flash: 00000000: Reserved 00000001: DTOP phase duration = 2 HCLK clock cycles 00000010: DTOP phase duration = 3 HCLK clock cycles 11111111: DTOP phase duration = 256 HCLK clock cycles (Default value after reset) For each memory type and access mode data-phase duration, please refer to the corresponding figures (Figure 19-3 to Figure 19-15). For example: Mode 1, read access, DTOP = 1: Data-phase duration = DTOP + 3 = 4 HCLK clock cycles</p>
Bit 7:4	<p>ADRHLHD: Address-hold phase duration These bits define the duration of the address hold phase (Please refer to Figure 19-12 to Figure 19-15) used in SRAM, ROM, and asynchronous muxed mode accesses of NOR Flash: 0000: Reserved 0001: ADRHLHD phase duration = 2 HCLK clock cycles 0010: ADRHLHD phase duration = 3 HCLK clock cycles ... 1111: ADRHLHD phase duration = 16 HCLK clock cycles (Default value after reset) Note: In synchronous accesses, this value has no effect. The address hold phase duration is always 1 memory clock period.</p>
Bit 3:0	<p>ADROP: Address setup phase duration These bits define the duration of the address setup phase (Please refer to Figure 19-3 to Figure 19-15) used in SRAM, ROM, asynchronous muxed mode accesses of NOR Flash: 0000: ADROP setup phase duration = 1 HCLK clock cycle 1111: ADROP setup phase duration = 16 HCLK clock cycles (Default value after reset) For each access mode address setup phase duration, please refer to the corresponding figures (Figure 19-3 to Figure 19-15). For example: Mode 2, read access, ADROP = 1: Address setup phase duration = ADROP + 1 = 2 HCLK clock cycles Note: In synchronous accesses, this value has no effect. The address setup phase duration is always 1 memory clock period.</p>

Note: PSRAM (CRAM) has variable hold latency due to internal refresh. Therefore, these memories will issue the NWAIT signal at the whole latency phase to prolong the latency as needed.

With PSRAM (CRAM) in use, the DTSTBL field must be set to 0, so that XMC can exit its latency phase in time, start sampling NWAIT issued from the memory, and then begin to read or write when the memory is ready.

This method can also be used for the latest synchronous Flash memories, which can issue the NWAIT signal; please refer to the corresponding Flash Memory manual for more detailed information.

19.4.1.3 SRAM/NOR Flash Write Timing Register 1...4 (XMC_BK1TMGWR1...4)

Address offset: 0xA0000000 + 0x104 + 8* (x - 1), x = 1...4

Reset value: 0xFFFFFFFF

This register contains the control information of each memory bank used for SRAM, ROM, and NOR Flash memories. If the TMGWREN bit is set in the XMC_BK1CTRLx register, then this register is used for write accesses.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	MODE	Reserved								INTVLOP					
res	rw	rw				res				rw	rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTOP								ADRHL				ADROP			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:30		Reserved													
Bit 29:28		MODE: Access mode The 2 bits define asynchronous access mode. They only have effects when the TMGWREN bit is set in the XMC_BK1CTRLx register. 00: Access mode A 01: Access mode B 10: Access mode C 11: Access mode D													
Bit 27:20		Reserved													
Bit 19:16		INTVLOP: Bus turnaround phase duration These bits define the delay time on the bus after a read access (only used for the muxed mode of NOR Flash access). After a read access, the controller needs to issue the address for the next operation on data bus, and this delay is used to avoid bus conflicts. If the extended memory does not include muxed mode memory of the bus, or the slowest memory can restore data bus to HiZ state within 6 HCLK clock periods, this parameter can be set as its minimum. 0000: Bus turnaround phase duration = 1 HCLK clock cycle 1111: Bus turnaround phase duration = 16 HCLK clock cycles (Default value after reset)													
Bit 15:8		DTOP: Data-phase duration These bits define the duration of the data phase (Please refer to Figure 19-3 to Figure 19-15) used in SRAM, ROM, asynchronous muxed mode accesses of NOR Flash: 00000000: Reserved 00000001: DTOP phase duration = 2 HCLK clock cycles 00000010: DTOP phase duration = 3 HCLK clock cycles 11111111: DTOP phase duration = 256 HCLK clock cycles (Default value after reset)													
Bit 7:4		ADRHL : Address-hold phase duration These bits define the duration of the address hold phase (Please refer to Figure 19-12 to Figure 19-15) used in SRAM, ROM, asynchronous muxed mode accesses of NOR Flash: 0000: Reserved 0001: ADRHL phase duration = 2 HCLK clock cycles 0010: ADRHL phase duration = 3 HCLK clock cycles ... 1111: ADRHL phase duration = 16 HCLK clock cycles (Default value after reset) Note: In synchronous accesses, this value has no effect. The address hold phase duration is always 1 memory clock period.													

Bit 3:0	ADROP: Address setup phase duration These bits define the duration of the address setup phase with HCLK cycles (Please refer to Figure 19-3 to Figure 19-15) used in SRAM, ROM, asynchronous muxed mode accesses of NOR Flash: 0000: ADROP setup phase duration = 1 HCLK clock cycle 1111: ADROP setup phase duration = 16 HCLK clock cycles (Default value after reset) Note: In synchronous accesses, this value has no effect. The address setup phase duration is always 1 memory clock period.
---------	--

19.4.1.4 SRAM/NOR Additional Timing Register 1...4 (XMC_EXT1...4)

Address offset: 0xA0000000+0x220+4* (x-1) ,x=1...4

Reset value: 0x00000808

This register contains the control information of each memory bank used for SRAM, ROM, and NOR Flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BURSTURNR2R								BURSTURNW2W							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:16	Reserved.
Bit 15: 8	BURSTURNR2R: Bus turnaround phase for consecutive read duration This is used to define bus turnaround phase for asynchronous consecutive read duration. 00000000: consecutive read operation at an interval of 1 HCLK period 00000001: consecutive read operation at an interval of 2 HCLK period 00001000: consecutive read operation at an interval of 9 HCLK period (default) 11111111: consecutive read operation at an interval of 256 HCLK period
Bit 7: 0	BURSTURNW2W: Bus turnaround phase for consecutive write duration This is used to define bus turnaround phase for asynchronous consecutive write duration. 00000000: consecutive write operation at an interval of 1 HCLK period 00000001: consecutive write operation at an interval of 2 HCLK period 00001000: consecutive write operation at an interval of 9 HCLK period (default) 11111111: consecutive write operation at an interval of 256 HCLK period

19.4.2 NAND Flash and PC Card Controller Register

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

19.4.2.1 PC Card/NAND Flash Control Register 2...4 (XMC_BK2...4CTRL)

Address offset: 0xA0000000 + 0x40 + 0x20* (x - 1), x = 2...4

Reset value: 0x00000018

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										ECCPGSIZE			DLYAR		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLYAR		DLYCR		Reserved	ECCEN	BUSTYPE	DEV	EN	WAITEN	Reserve d					
rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw	res

Bit 31:20	Reserved
Bit 19:17	ECCPGSIZE: ECC page size These bits define the extended ECC page size: 000: 256 bytes 001: 512 bytes 010: 1024 bytes 011: 2048 bytes 100: 4096 bytes 101: 8192 bytes
Bit 16:13	DLYAR: ALE to RE delay These bits set the time from ALE low to RE low in number of AHB clock cycles (HCLK). Time: $t_{ar} = (\text{DLYAR} + \text{SET} + 4) \times \text{THCLK}$, where THCLK is the HCLK clock period. 0000: 1 HCLK cycle (Default value) 1111: 16 HCLK cycles Note: According to different address spaces, SET is TMGMEM_STP or TMGATT_STP.
Bit 12:9	DLYCR: CLE to RE delay Time: $t_{clr} = (\text{DLYCR} + \text{SET} + 4) \times \text{THCLK}$, where THCLK is the HCLK clock period. 0000: 1 HCLK cycle (Default value) 1111: 16 HCLK cycles Note: According to different address spaces, SET is TMGMEM_STP or TMGATT_STP.
Bit 8:7	Reserved
Bit 6	ECCEN: ECC computation logic enable bit 0: ECC logic is disabled and reset (Default after reset). 1: ECC logic is enabled.
Bit 5:4	BUSTYPE: Databus width These bits define the external NAND Flash data bus width. 00: 8 bits (Default value after reset) 01: 16 bits (PC Card mandatory) 10: Reserved; do not use. 11: Reserved; do not use.
Bit 3	DEV: Memory type This bit defines the type of device attached to the corresponding memory bank. 0: PC Card, CF Card , CF+ Card, or PCMCIA 1: NAND Flash (Default value after reset)
Bit 2	EN: PC Card/NAND Flash memory bank enable bit This bit enables the memory bank. Accessing a disabled memory bank causes an ERROR on AHB bus 0: The corresponding memory bank is disabled (Default after reset). 1: The corresponding memory bank is enabled.

Bit 1	WAITEN: Wait feature enable bit This bit enables PC Card/NAND Flash memory bank wait function. 0: Wait function is disabled (Default value after reset). 1: Wait function is enabled. Note: For a PC Card, if the wait feature is enabled, the MGMEM_OP/TMGATT_OP/TMGIO_OP bits must be programmed to a value higher than $4 + \text{max_wait_assertion_time}/\text{HCLK}$, where max_wait_assertion_time is the maximum time taken by NWAIT to go low once NOE/NWE or NIORD/NIOWR is low.
Bit 0	Reserved

19.4.2.2 FIFO Status and Interrupt Register 2...4 (XMC_BK2...4STS)

Address offset: 0xA0000000 + 0x44 + 0x20*(x - 1), x = 2...4

Reset value: 0x00000040

This register contains information about FIFO status and interrupt. XMC has a FIFO used to store up to 16 words of data from the AHB when writing to memories.

This function helps to free AHB while XMC is draining its FIFO into the memory.

For ECC purposes, this register has one bit that indicates the FIFO status. The ECC is calculated while the data are written to the memory; hence, in order to read the correct ECC value, the software must wait until the FIFO is empty.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								FIFOE	IFE_N	IHL_EN	IRE_N	IFF	IHL_F	IRF	
res								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:7	Reserved
Bit 6	FIFOE: FIFO empty Read-only bit, indicating FIFO status 0: FIFO is not empty. 1: FIFO is empty.
Bit 5	IFEN: Interrupt falling edge detection enable bit 0: Interrupt falling edge detection request is disabled. 1: Interrupt falling edge detection request is enabled.
Bit 4	IHEN: Interrupt high-level detection enable bit 0: Interrupt high-level detection request is disabled. 1: Interrupt high-level detection request is enabled.
Bit 3	IREN: Interrupt rising edge detection enable bit 0: Interrupt rising edge detection request is disabled. 1: Interrupt rising edge detection request is enabled.
Bit 2	IFF: Interrupt falling edge status This bit is set by hardware and cleared by software. 0: No interrupt falling edge occurs. 1: Interrupt falling edge occurs.
Bit 1	IHLF: Interrupt high-level status This bit is set by hardware and cleared by software. 0: No interrupt high-level occurs. 1: Interrupt high-level occurs.

Bit 0	IRF: Interrupt rising edge status This bit is set by hardware and cleared by software. 0: No interrupt rising edge occurs. 1: Interrupt rising edge occurs.
-------	---

19.4.2.3 Common Memory Space Timing Register 2...4 (XMC_BK2...4TMGMEM)

Address offset: 0xA0000000 + 0x48 + 0x20*(x - 1), x = 2...4

Reset value: 0xFCFCFCFC

Each XMC_BKxTMGMEM (x = 2...4) register contains the timing information for PC Card or NAND Flash memory bank x. The parameters are used for accesses to the common memory space of the 16-bit PC Card/CF Card, or to access the NAND Flash command, address, and data read/write access.

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

Bit 31:24	WRSTP: Common memory x databus HiZ time After the start of a PC Card/NAND Flash write access to common memory space x, databus needs to be kept in HiZ for a period of time. This parameter defines the time during which the databus is kept in HiZ in the number of HCLK clock cycles (+ 1 for NAND types). This parameter is only valid for write transactions: 00000000: (0x00) PC Card is 0 HCLK cycle; NAND Flash is 1 HCLK cycle. 11111111: (0xFF) PC Card is 255 HCLK cycles; NAND Flash is 256 HCLK cycles.
Bit 23:16	HLD: Common memory x hold time For PC Card/NAND Flash read/write accesses to the common memory space x, these bits define the hold time of address signal (data signal for write access) after command is sent (NWE and NOE become high) in HCLK clock cycle number. 00000000: Reserved 00000001: 1 HCLK cycle 11111111: 255 HCLK cycles
Bit 15:8	OP: Common memory x wait time For PC Card/NAND Flash read/write accesses to the common memory space x, these bits define the minimum time of command hold (NWE and NOE are low) in HCLK clock cycle number(+ 1). The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK. 00000000: Reserved 00000001: 1 HCLK cycle (+ wait cycle introduced by deasserting NWAIT) 11111111: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT from Card)
Bit 7:0	STP: Common memory x setup time For PC Card/NAND Flash read/write accesses to the common memory space x, these bits define address signal setup time before command assertion (NWE and NOE go low) in HCLK clock cycle number (+ 1 in PC Card operation, + 2 in NAND Flash operation). 00000000: PC Card is 1 HCLK cycle; NAND Flash is 2 HCLK cycles. 11111111: PC Card is 256 HCLK cycles; NAND Flash is 257 HCLK cycles.

19.4.2.4 Attribute Memory Space Timing Register 2...4 (XMC_BK2...4TMGATT)

Address offset: 0xA0000000 + 0x4C + 0x20* (x - 1), x = 2...4

Reset value: 0xFCFCFCFC

Each XMC_BKxTMGATT (x = 2...4) read/write register contains the timing information for PC Card/CF card or NAND Flash memory bank x. The parameters are used for accesses to the attribute memory space of the 8-bit PC Card/CF Card (Every AHB operation is split to a series of 8-bit operations), or when the timing of last address write access of the NAND Flash is different from other operations (For more information about ready/busy management, please refer to [Section 19.3.3.5](#)).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRSTP								HLD							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OP								STP							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:24		WRSTP: Attribute memory x databus HiZ time After the start of a PC Card/NAND Flash write access to attribute memory space x, databus needs to be kept in HiZ for a period of time. This parameter defines the time during which the databus is kept in HiZ in the number of HCLK clock cycles. This parameter is only valid for write transactions: 00000000: 0 HCLK cycles 11111111: 255 HCLK cycles (Default value after reset)													
Bit 23:16		HLD: Attribute memory x hold time For PC Card/NAND Flash read/write accesses to the attribute memory space x, these bits define the hold time of address signal (data signal for write access) after command is sent (NWE and NOE become high) in HCLK clock cycle number. 00000000: Reserved 00000001: 1 HCLK cycle 11111111: 255 HCLK cycles (Default value after reset)													
Bit 15:8		OP: Attribute memory x wait time For PC Card/NAND Flash read/write accesses to the attribute memory space x, these bits define the minimum time of command hold (NWE and NOE are low) in HCLK clock cycle number (+ 1). The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK. 00000000: Reserved 00000001: 1 HCLK cycle (+ wait cycle introduced by deasserting NWAIT) 11111111: 255 HCLK cycles (+ wait cycle introduced by deasserting NWAIT from Card)													
Bit 7:0		STP: Attribute memory x setup time For PC Card/NAND Flash read/write accesses to the attribute memory space x, these bits define address signal setup time before command assertion (NWE and NOE go low) in HCLK clock cycle number (+ 1). 00000000: 1 HCLK cycle 11111111: 256 HCLK cycles (Default value after reset)													

19.4.2.5 I/O Space Timing Register 4 (XMC_BK4TMGIO)

Address offset: 0xA0000000 + 0xB0

Reset value: 0xFCFCFCFC

The XMC_BK4TMGIO register includes the timing parameters of 16-bit PC Card/CF Card access in I/O space.

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

Bit 31:24	WRSTP: I/O x databus HiZ time After the start of a PC Card write access to I/O space x, databus needs to be kept in HiZ for a period of time. This parameter defines the time during which the databus is kept in HiZ in the number of HCLK clock cycles. This parameter is only valid for write transactions. 00000000: 0 HCLK cycle 11111111: 255 HCLK cycles (Default value after reset)
Bit 23:16	HLD: I/O x hold time For PC Card read/write accesses to the I/O space x, these bits define the hold time of address signal (data signal for write access) after command is sent (NWE and NOE become high) in HCLK clock cycle number. 00000000: Reserved 00000001: 1 HCLK cycle 11111111: 255 HCLK cycles (Default value after reset)
Bit 15:8	OP: I/O x wait time For PC Card/NAND Flash read/write accesses to the I/O space x, these bits define the minimum time of command hold (SMNWE and SMNOE are low) in HCLK clock cycle number (+ 1). The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of HCLK. 00000000: Reserved; do not use this value. 00000001: 2 HCLK cycles (+ wait cycle introduced by deasserting NWAIT) 11111111: 256 HCLK cycles (+ wait cycle introduced by deasserting NWAIT from Card)
Bit 7:0	STP: I/O x setup time For PC Card read/write accesses to the I/O space x, these bits define address signal setup time before command assertion (NWE and NOE go low) in HCLK clock cycle number (+ 1). 00000000: 1 HCLK cycle 11111111: 256 HCLK cycles (Default value after reset)

19.4.2.6 ECC Result Register 2/3 (XMC_BK2/3ECC)

Address offset: 0xA0000000 + 0x54 + 0x20* (x - 1), x = 2 or 3

Reset value: 0x00000000

The two registers contain the current error correction code value computed by the ECC computation modules of XMC controller, one ECC module per NAND Flash memory bank. When CPU reads/writes the data from the NAND Flash at the correct address (See [Section 19.3.3.6](#)), ECC computation module automatically processes the data written or read. According to the ECCPGSIZE field in the XMC_BKxCTRL registers, after the last byte read, CPU must read the computed ECC value from the MC_BKxECC register, and then compare the value with those data recorded in the NAND Flash spare area, to determine whether it is valid and can be used for correction if applicable. After being read, the XMC_BKxECCR register must be cleared by configuring the ECCEN bit as '0'. The ECCEN bit is set to '1' again when computing a new data block.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ECC															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECC															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:0				ECC: ECC result The computed result generated by the ECC circuit. Table 19-36 shows the content of these bits.											

Table 19-36 ECC Result Relevant Bit

ECCPGSIZE[2:0]	Page Size (Byte)	ECC Valid Bit
000	256	ECC[21:0]
001	512	ECC[23:0]
010	1024	ECC[25:0]
011	2048	ECC[27:0]
100	4096	ECC[29:0]
101	8192	ECC[31:0]

20 SDIO Interface

20.1 Introduction

SD/SDIO MMC card host interface (SDIO) provides an interface between the AHB peripheral bus, MultiMediaCard (MMC), SD memory cards, SDIO cards, and CE-ATA devices.

The MultiMediaCard system specifications, published by the MMCA technical committee, are available on the MultiMediaCard Association website (www.mmca.org).

CE-ATA system specifications are available on the CE-ATA workgroup website (www.ce-ata.org).

The SDIO main features include the following:

- Full compliance with MultiMediaCard System Specification Version 4.2; Supports three different data bus modes: 1-bit (Default), 4-bit, and 8-bit
- Full compliance with previous versions of MultiMediaCards (Forward compatibility)
- Full compliance with SD Memory Card Specifications Version 2.0
- Full compliance with SD I/O Card Specification Version 2.0; Supports two different data bus modes: 1-bit (Default) and 4-bit
- Full support of the CE-ATA features (Full compliance with CE-ATA digital protocol Rev1.1)
- Data transfer up to 50 MHz for the 8-bit bus mode
- Data and command output enable signals to control external bidirectional drivers

Note:

1. *SDIO does not have an SPI-compatible communication mode.*
2. *The SD memory card protocol is a superset of the MultiMediaCard protocol, as defined in the MultiMediaCard system specification V2.11. Several commands required for SD memory devices are not supported by SD I/O-only cards or the I/O portion of combo cards. In this case, some of these commands, such as erase commands, have no effect in SD I/O devices, and thus they are not supported in the SDIO. In addition, several commands are different between SD memory cards and SD I/O cards, so they are not supported in the SDIO. For more details, please refer to SD I/O card Specification Version 1.0. CE-ATA is supported over the MMC interface with the existing MMC access primitives. For the interface electrical and signaling definitions of SDIO, please check the MMC reference.*

The MultiMediaCard/SD bus connects all the cards to the controller.

The current version of the SDIO supports only one SD/SDIO/MMC 4.2 card at a time, but it supports a stack of MMC 4.1 or previous versions.

20.2 Main Features

Communication over the bus is based on command and data transfers.

The basic transaction on the MultiMediaCard/SD/SD I/O bus is command/response transaction. These types of bus transaction transfer their information directly within command or response structure. In addition, some operations have a data token.

Data transfers to/from SD/SDIO memory cards are done in data blocks. Data transfers to/from MMC are done in data blocks or streams. Data transfers to/from the CE-ATA devices are also done in data blocks.

Figure 20-1 SDIO “No Response” and “No Data” Operation

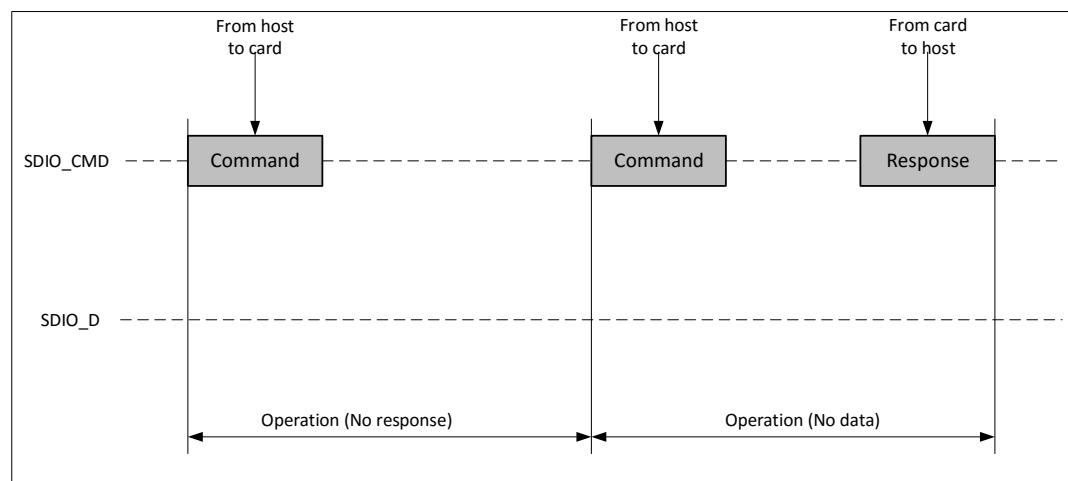


Figure 20-2 SDIO (Multiple) Data Block Read Operation

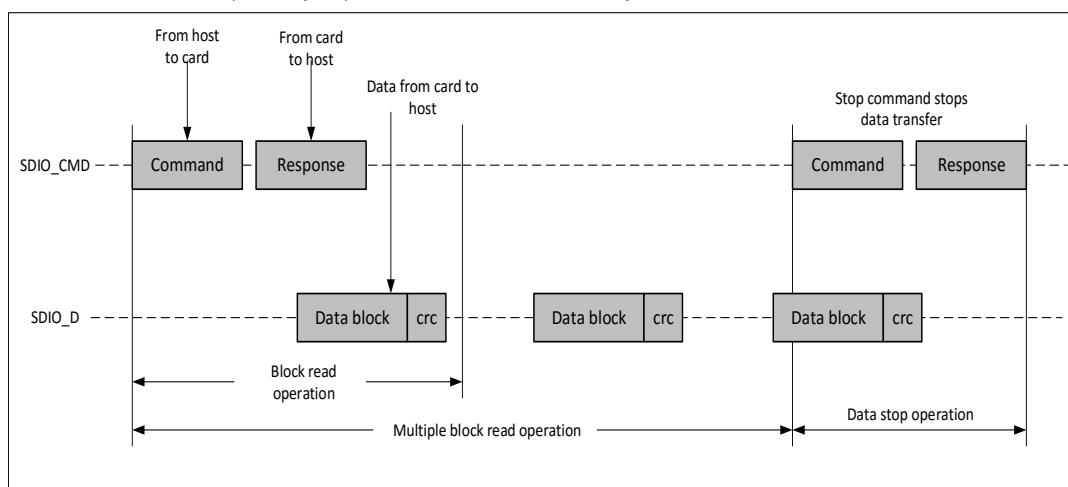
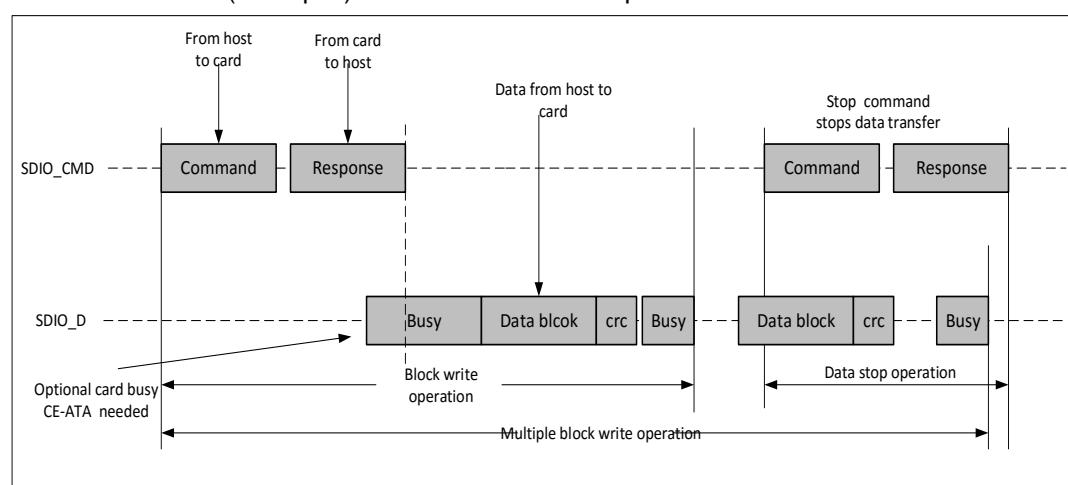


Figure 20-3 SDIO (Multiple) Data Block Read Operation



Note When there is a busy signal, SDIO (SDIO_D0 pulled low) will not send any data.

Figure 20-4 SDIO Sequential Read Operation

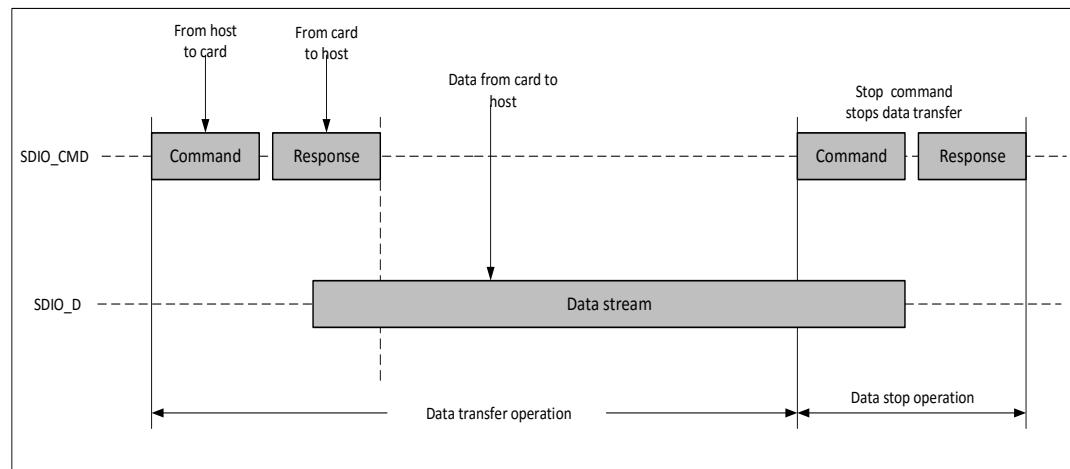
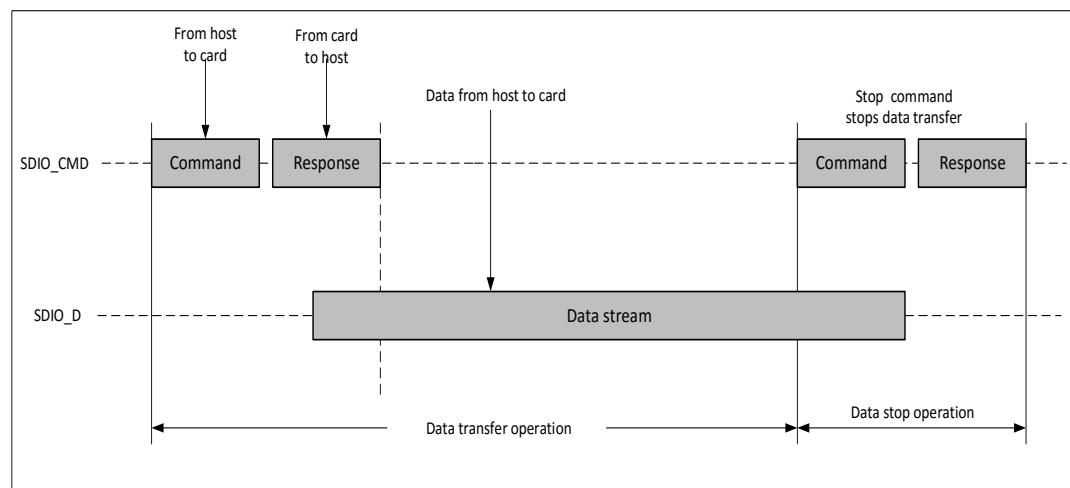


Figure 20-5 SDIO Sequential Write Operation



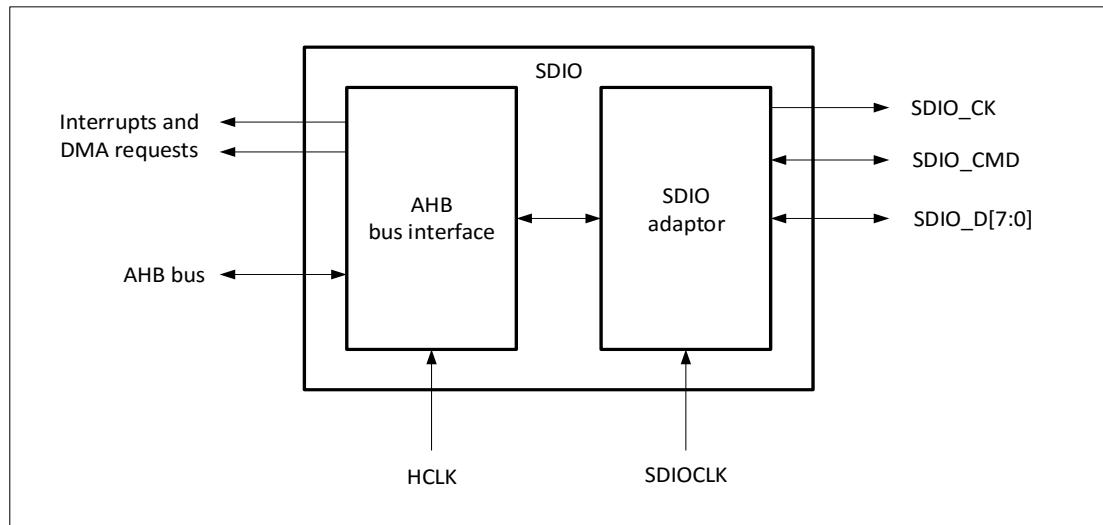
20.3 Function Overview

20.3.1 SDIO Function Overview

SDIO consists of 2 parts:

- SDIO adapter block: Provides all functions specific to the MMC/SD/SD I/O card, such as clock generation, commands, and data transfers.
- AHB interface: Accesses the SDIO adapter registers and generates interrupt and DMA request signals.

Figure 20-6 SDIO Block Diagram



After reset, SDIO_D0 is used for data transfer by default. After initialization, the host can change the data bus width.

If a MultiMediaCard is connected to the bus, SDIO_D0, SDIO_D[3:0], or SDIO_D[7:0] can be used for data transfer. MMC V3.31 or previous versions support only 1-bit data, so only SDIO_D0 can be used.

If an SD or SD I/O card is connected to the bus, data transfer can be configured by the host to use SDIO_D0 or SDIO_D[3:0]. All data lines operate in push-pull mode.

SDIO_CMD has two operational modes:

- Open-drain for initialization (only for MMC V3.31 or previous)
- Push-pull for command transfer (SD/SD I/O card and MMC V4.2 also use push-pull drivers for initialization.)

SDIO_CK is the clock to the card: 1-bit command or data is transferred on command and data lines during each clock cycle. The clock frequency can vary between 0 MHz and 20 MHz for a MultiMediaCard V3.31 protocol, between 0 MHz and 50 MHz for a MultiMediaCard V4.0/4.2 protocol, or between 0 MHz and 50 MHz for an SD/SD I/O card.

SDIO uses two clock signals:

- SDIO adapter clock (SDIOCLK = HCLK)
- AHB bus clock (HCLK)

Table 20-1 shows the signals used on the MultiMediaCard/SD/SD I/O card bus.

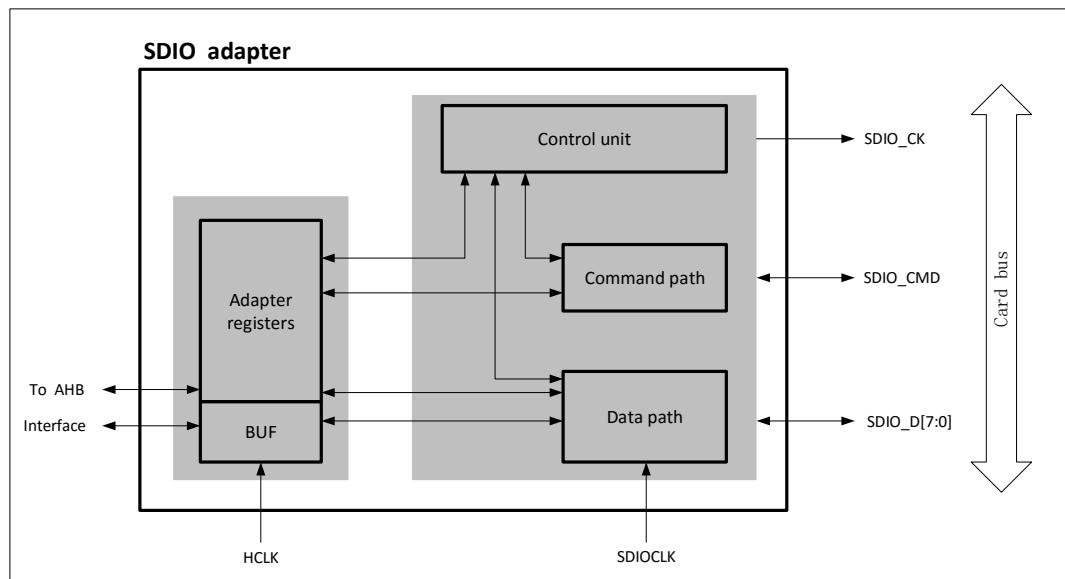
Table 20-1 SDIO Pin Definitions

Pin	Direction	Description
SDIO_CK	Output	MultiMediaCard/SD/SDIO card clock. This pin is the clock from the host to card.
SDIO_CMD	Bidirectional	MultiMediaCard/SD/SDIO card command. This pin is the bidirectional command/response signal.
SDIO_D[7:0]	Bidirectional	MultiMediaCard/SD/SDIO card data. These pins are the bidirectional data bus.

20.3.1.1 SDIO Adapter

Figure 20-7 shows a simplified block diagram of an SDIO adapter.

Figure 20-7 SDIO Adaptor



The SDIO adapter is a multimedia/secure digital memory card bus master that provides an interface to a multimedia card stack or to a secure digital memory card. It consists of five parts:

- Adapter register block
- Control unit
- Command path
- Data path
- Data BUF

Note: The adapter registers and BUF use the AHB bus clock domain (HCLK); the control unit, command path, and data path use the SDIO adapter clock domain (SDIOCLK).

Adapter register block

The adapter register block contains all system registers. This block also generates the signals that clear the static flags in the multimedia card. The clear signals are generated when '1' is written into the corresponding bits in the SDIO clear register.

Control Unit

The control unit includes the power management functions and the clock divider for memory card clock.

There are three power phases:

- Power-off
- Power-up
- Power-on

Figure 20-8 Control Unit

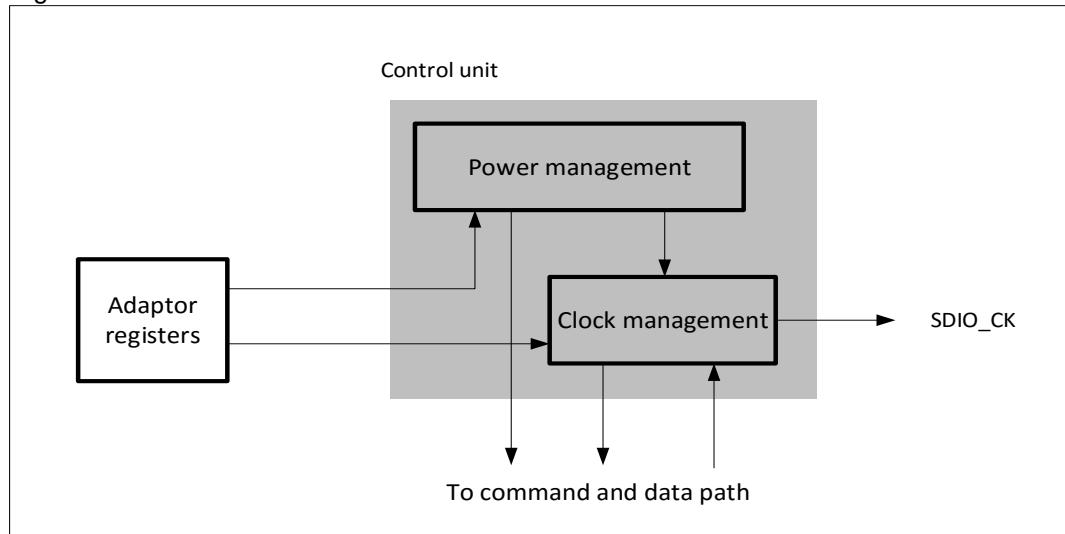


Figure 20-8 is the block diagram of control unit. It consists of a power management sub-unit and a clock management sub-unit.

The power management subunit disables the card bus output signals at the power-off and power-up phases.

The clock management subunit generates and controls the SDIO_CK signal. The SDIO_CK output can use either the clock divide or the clock bypass mode.

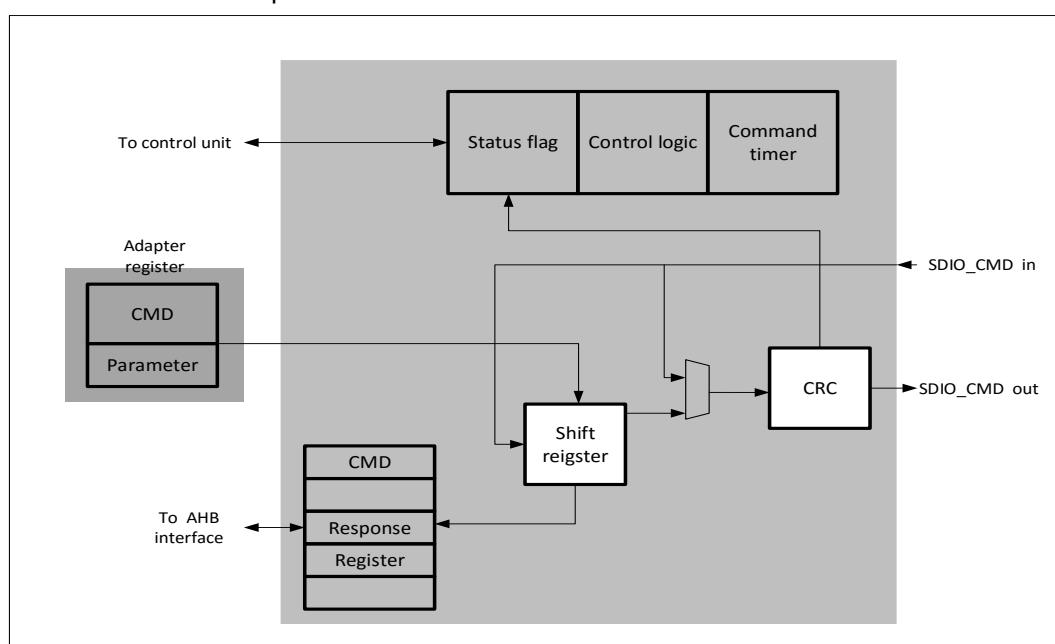
The clock output is inactive under the following conditions:

- After reset
- At power-off and power-up phase
- When the power-saving mode is enabled, and the card bus is at idle state (8 clock periods after both the command and data path subunits enter idle phase).

Command path

Command path unit sends commands to card and receives responses from card.

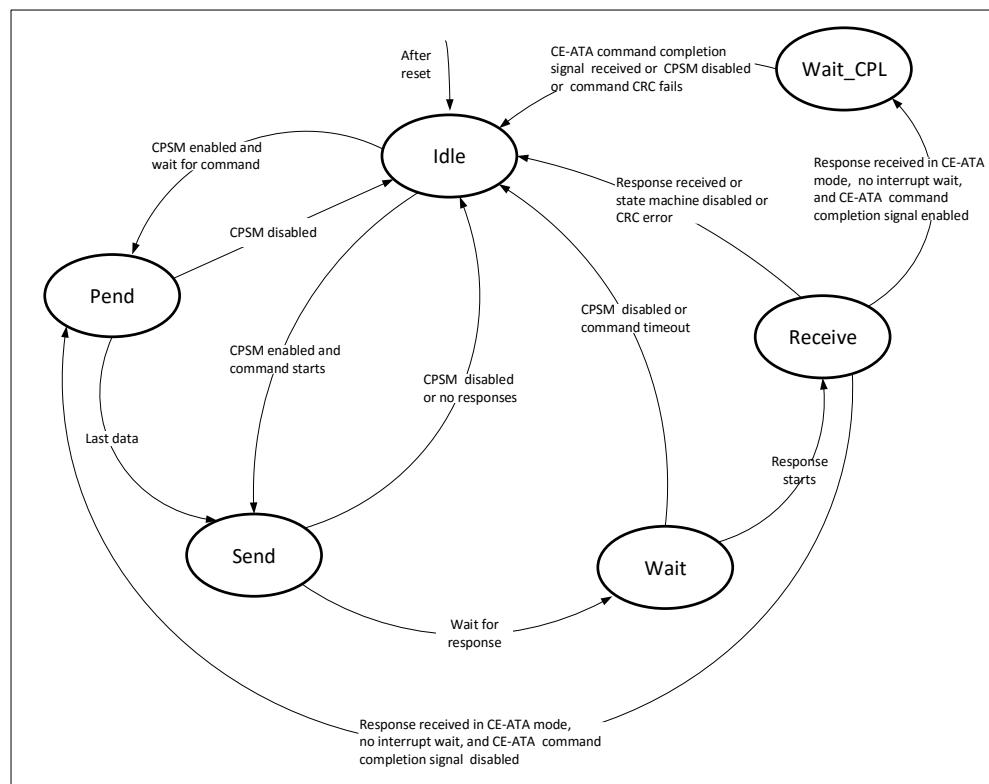
Figure 20-9 SDIO Adapter Command Path



- Command path state machine (CPSM)

- When the command register is written, and the enable bit is set, command transfer starts. When the command transfer is completed, the command path state machine (CPSM) sets the status flags and enters the Idle state if a response is not required (See [Figure 20-10](#)). When the response is received, the received CRC code and the internally-generated code are compared, and the appropriate status flags are set.

Figure 20-10 Command Path State Machine (CPSM)



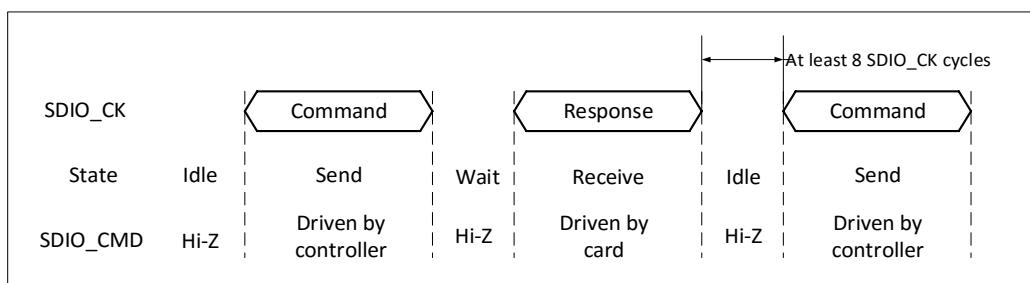
After entering the Wait state, the command timer starts running; if the timeout is reached before CPSM enters the Receive state, the timeout flag is set, and CPSM enters the Idle state.

Note: *The command timeout is fixed to 64 SDIO_CK clock periods.*

If the interrupt bit is set in the command register, the timer is disabled, and CPSM waits for an interrupt request from one of the cards. If a pending bit is set in the command register, CPSM enters the Pend state and waits for a CmdPend signal from the data path subunit. When CmdPend signal is detected, CPSM moves to the Send state, which enables the data counter to transmit stop command.

Note: *CPSM remains at the Idle state for at least 8 SDIO_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.*

Figure 20-11 SDIO Command Transfer



- Command format

- 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 are available only for MMC V3.31 or previous). Commands are transferred serially on the CMD line. All commands have a fixed length of 48 bits. The general format of a command token for MultiMediaCards, SD-Memory cards, and SDIO-Cards is shown in Table 20-2. CE-ATA commands are an extension of MMC V4.2 commands, so they have the same format. The command path operates in a half-duplex mode so that commands and responses can either be sent or received. If CPSM is not at the Send state, the SDIO_CMD output is at the HiZ state, as shown in [Figure 20-11](#). Data on SDIO_CMD are synchronous with the rising edge of SDIO_CK.

Table 20-2 Command Format

Bit	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7
0	1	1	End bit

- Response: A response is a token sent from an addressed card to the host, or synchronously from all connected cards to the host for MMC V3.31 or previous versions. Response is an answer to a previously-received command. Responses are transferred serially on the CMD line.

SDIO supports two response types. Both use CRC error checking:

- 48-bit short response
- 136-bit long response

Note: *If the response does not contain a CRC (e.g. CMD1 response), the device driver must ignore the CRC failed status.*

Table 20-3 Short Response Format

Bit	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	-	Command index
[39:8]	32	-	Argument
[7:1]	7	-	CRC7 (or 1111111)
0	1	1	End bit

Table 20-4 Long Response Format

Bit	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127	-	CID or CSD (including internal CRC7)
0	1	1	End bit

The command register contains the command index (six bits sent to a card) and the command type; they determine whether the command requires a response and whether the response is 48-bit or 136-bit long (See [Section 20.4.4](#)). Status flags in the command path are shown in Table 20-5.

Table 20-5 Command Path Status Flags

Flag	Description
CMDRSPCMPL	Response CRC is correct.
CMDFAIL	Response CRC fails.
CMDCMPL	Command (that does not need a response) is sent.
CMDTIMEOUT	Response timeout
DOCMD	Command transfer is in progress.

The CRC generator calculates the CRC checksum for all bits before the CRC code. This includes the start bit, transmitter bit, command index, and command argument (or card status). The CRC checksum is calculated for the first 120 bits of CID or CSD for the long response format.

Note: *In long response format, the start bit, transmitter bit, and the six reserved bits are not used in the CRC calculation.*

The CRC checksum is a 7-bit value:

$$\text{CRC}[6:0] = \text{Remainder}[(M(x) * x^7)/G(x)]$$

$$G(x) = x^7 + x^3 + 1$$

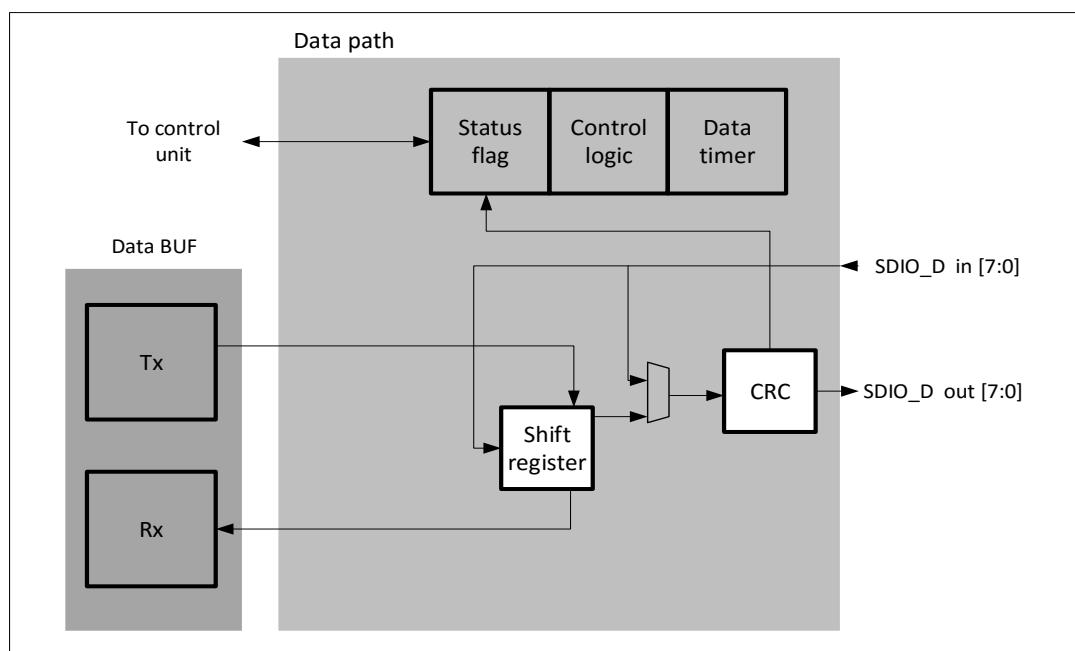
$$M(x) = (\text{Start bit}) * x^{39} + \dots + (\text{Last bit before CRC}) * x^0, \text{ or}$$

$$M(x) = (\text{Start bit}) * x^{119} + \dots + (\text{Last bit before CRC}) * x^0$$

Data path

The data path subunit transfers data to and from cards. Figure 20-12 shows a block diagram of the data path.

Figure 20-12 Data Path



The card data bus width can be programmed through the clock control register. If the 4-bit wide bus mode is enabled, 4-bit data is transferred per clock cycle over all four data signals (SDIO_D[3:0]); if the 8-bit wide bus mode is enabled, 8-bit data is transferred per clock cycle over all eight data signals (SDIO_D[7:0]). If the wide bus mode is not enabled, only 1-bit data is transferred per clock cycle over SDIO_D0.

Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to

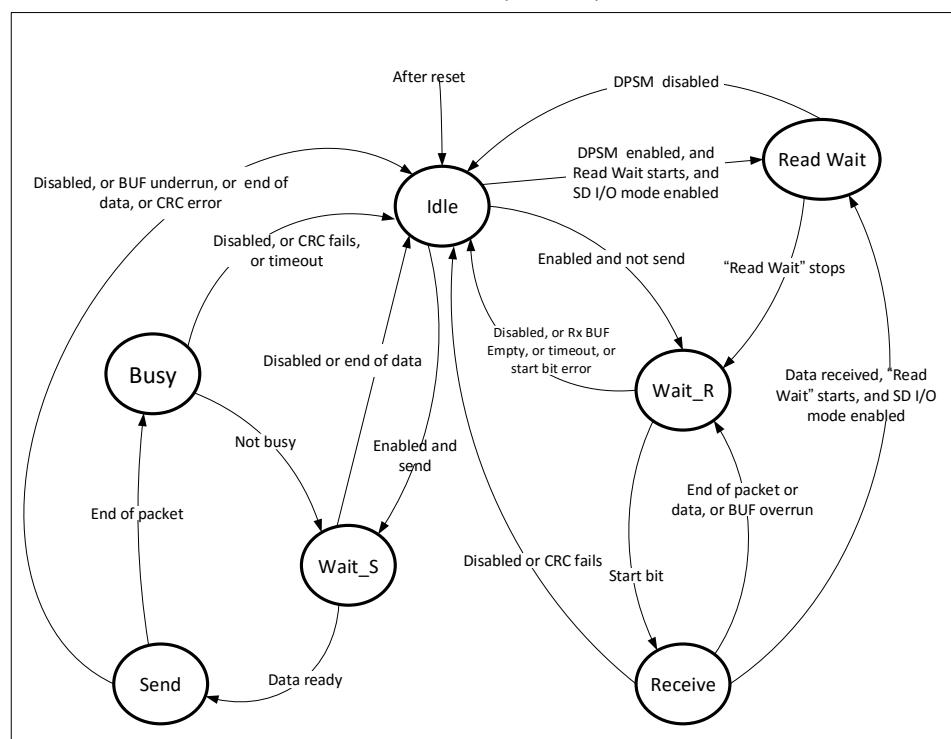
the Wait_S or Wait_R state when it is enabled:

- Send: DPSM moves to the Wait_S state. If there is data in the transmit BUF, DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: DPSM moves to the Wait_R state and waits for a start bit; when it receives a start bit, DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

Data path state machine (DPSM)

DPSM operates at SDIO_CK frequency. Data on the card bus signals is synchronous with the rising edge of SDIO_CK. DPSM has six states, as shown in Figure 20-13:

Figure 20-13 Data Path Status Machine (DPSM)



- Idle: The data path is inactive, and the SDIO_D[7:0] outputs are in HiZ. When the data control register is written and the enable bit is set, DPSM loads the data counter with a new value and enters either the Wait_S or the Wait_R state depending on the data direction bit.
- Wait_R: If the data counter is 0, DPSM enters Idle state when the receive BUF is empty. If the data counter is not 0, DPSM waits for a start bit on SDIO_D. DPSM moves to the Receive state and loads the data block counter if it receives a start bit before a timeout. If DPSM reaches a timeout before it detects a start bit, or a start bit error occurs, it moves to the Idle state and sets the timeout status flag.
- Receive: Serial data received from a card is packed in bytes and written to the data BUF. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
 - In block mode, when the data block counter reaches 0, DPSM waits until it receives the CRC code. If the received code matches with the internally generated CRC code, then DPSM moves to the Wait_R state. If not, the CRC fail status flag is set, and DPSM moves to the Idle state.
 - In stream mode, DPSM receives data when the data counter is not 0. When the counter is 0, the remaining data in the shift register is written to the data BUF, and DPSM moves to the Wait_R state.

If a BUF overrun error occurs, DPSM sets the BUF error flag and moves to the Idle state:

- Wait_S: DPSM moves to the Idle state if the data counter is 0. If not, it waits until the data BUF empty flag is deasserted, and then moves to the Send state.

Note: DPSM remains at the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- Send: DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block or stream:
 - In block mode, when the data block counter reaches 0, DPSM sends an internally generated CRC code and end bit, and then moves to the Busy state.
 - In stream mode, when the enable bit is high and the data counter is not 0, DPSM sends data to a card and then moves to the Idle state.
 If a BUF underrun error occurs, DPSM sets the BUF error flag and moves to the Idle state.
- Busy: DPSM waits for the CRC status flag:
 - If DPSM does not receive a positive CRC status, it moves to the Idle state and sets the CRC fail status flag.
 - If DPSM receives a positive CRC status, it moves to the Wait_S state if SDIO_D0 is not low (the card is not busy).
 If a timeout occurs while DPSM is at the Busy state, it sets the data timeout flag and moves to the Idle state.
- The data timer is enabled when DPSM is at the Wait_R or Busy state, and it generates the data timeout error:
 - When transmitting data, the timeout occurs if DPSM stays at the Busy state for longer than the programmed timeout period.
 - When receiving data, the timeout occurs if the end of the data is not true, and DPSM stays at the Wait_R state for longer than the programmed timeout period.
- Data: Data can be transferred from the card to the host, and vice versa. Data is transferred via the data lines. They are stored in a BUF of 32 words, with each word 32-bit wide.

Table 20-6 Data Token Format

Description	Start Bit	Data	CRC16	End Bit
Block data	0	-	Yes	1
Stream data	0	-	No	1

Data BUF

The data BUF (first-in-first-out) subunit is a data buffer with a transmit and receive unit.

The BUF contains a 32-bit wide, 32-word deep data buffer, and transmit and receive logic. Because the data BUF operates in the AHB clock domain (HCLK), all signals from the subunits in the SDIO clock domain (SDIOCLK) are resynchronized.

Depending on the DOTX and DORX flags, the BUF can be disabled, transmit enabled, or receive enabled. DOTX and DORX are driven by the subunit of data path and are mutually exclusive:

- The transmit BUF refers to the transmit logic and data buffer when DOTX is asserted.
- The receive BUF refers to the receive logic and data buffer when DORX is asserted.
- Transmit BUF: Data can be written to the transmit BUF through the AHB interface when the SDIO is enabled for transmission.
The transmit BUF is accessible via 32 sequential addresses. The transmit BUF contains a data output register that holds the data word pointed to by the read pointer. After the data path subunit loads its shift register, it moves the read pointer to the next data and outputs the data.
If the transmit BUF is disabled, all status flags are deasserted. The subunit of data path asserts DOTX when it transmits data.

Table 20-7 Transmit BUF Status Flag

Flag	Description
TXBUF_F	Set to high when all 32 transmit BUF words contain valid data.
TXBUF_E	Set to high when all the 32 transmit BUF words do not contain valid data.
TXBUF_H	Set to high when 8 or more transmit BUF words are empty. This flag can be used as a DMA request.
TXBUF	Set to high when the transmit BUF contains valid data. This flag is the opposite of the TXBUF_E flag.
TXERRU	Set to high when an underrun error occurs. This flag is cleared by writing to the SDIO clear register.

- Receive BUF: When the subunit of data path receives a word of data, it writes the data into BUF. The write pointer is automatically incremented 1 after the write operation is completed. On the read side, a read pointer always points to the current data in the BUF. If the receive BUF is disabled, all status flags are deasserted, and the read and write pointers are reset. The subunit of data path asserts DORX when it receives data. Table 20-8 lists the receive BUF status flags. The receive BUF is accessible via 32 sequential addresses.

Table 20-8 Receive BUF Status Flag

Flag	Description
RXBUF_F	Set to high when all 32 receive BUF words contain valid data.
RXBUF_E	Set to high when all 32 receive BUF words do not contain valid data.
RXBUF_H	Set to high when 8 or more receive BUF words contain valid data. This flag can be used as a DMA request.
RXBUF	Set to high when the receive BUF is not empty. This flag is the opposite of the RXBUF_E flag.
RXERRO	Set to high when an overrun error occurs. This flag is cleared by writing to the SDIO clear register.

20.3.1.2 SDIO AHB Interface

The AHB interface generates interrupts/DMA requests, and accesses the SDIO registers/data BUF. It consists of a data path, register decoder, and interrupt/DMA logic.

SDIO interrupts

The interrupt logic generates an interrupt request signal when at least one of the selected status flags is high. A mask register is provided to select the conditions that will generate an interrupt. A status flag generates interrupt requests if a corresponding mask flag is set.

SDIO/DMA interface: Procedure for data transfers between SDIO and memory

In the following example, the SDIO host controller transfers 512 bytes to an MMC card, using CMD24 (WRITE_BLOCK). DMA controller can be used to fill the SDIO BUF with data stored in a memory.

1. Perform the card identification process
2. Increase the SDIO_CK frequency
3. Select the card by sending CMD7
4. Configure the DMA2 as follows:
 - a) Enable DMA2 controller and clear all interrupt flag bits.
 - b) Program the DMA2 Channel 4 source address register as the base address of memory buffer, and DMA2_Channel 4 destination address register as the SDIO_BUF register address.
 - c) Program DMA2 Channel 4 control register (memory increment, non-peripheral increment, peripheral and source width is word size).

- d) Enable DMA2 Channel 4
5. Send CMD24 (WRITE_BLOCK) as follows:
 - a) Program the SDIO data length register (SDIO data timer register should be already programmed before the card identification process).
 - b) Program the SDIO argument register with the address location of the card, where data is to be transferred.
 - c) Program the SDIO command register: CmdIndex with '24' (WRITE_BLOCK); WaitRest with '1' (SDIO card host waits for a response); CMDEN with '1' (SDIO card host is enabled to send a command). Other fields are kept at their reset values.
 - d) Wait for SDIO_STS[6] = CMDRSPCMPL interrupt, and then program the SDIO data control register, TFREN, with '1' (SDIO card host is enabled to send data); TFRDIR with '0' (from controller to card); TFRMODE with '0' (block data transfer); DMAEN with '1' (DMA enabled); BLKSIZE with '9' (512 bytes). Other fields are "don't care".
 - e) Wait for SDIO_STS[10] = DTBLKCMPL
6. Confirm that no channels are still enabled by polling the DMA enable channel status register.

20.3.2 Card Function Overview

20.3.2.1 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 bus. All data communications only use the command line (CMD).

20.3.2.2 Card Reset

The GO_IDLE_STATE command (CMD0) is a software reset command, and it puts the MultiMediaCard and SD memory at the Idle state. The IO_RW_DIRECT command (CMD52) resets the SD I/O card. After power-up or CMD0, all bus drivers of cards output are at the high impedance state, and all the cards are initialized with a default relative card address (RCA = 0x0001) and with a default driver register setting (lowest speed and highest driving current capability).

20.3.2.3 Operating Voltage Range Validation

All cards can communicate with the SDIO card host using any operating voltage within the specification range. The supported minimum and maximum VDD values are defined in the operation condition register (OCR) on the card.

Cards that store the card identification number (CID) and card specific data (CSD) in the internal memory can transfer the information only under data-transfer VDD conditions. When the host module of SDIO card and the card are not consistent in the VDD ranges, the card cannot complete the identification cycle and cannot send CSD data. Therefore, SDIO card host can use the following special commands to identify and reject cards that do not match the VDD range: SEND_OP_COND (CMD1), SD_APP_OP_COND (SD memory card ACMD41), and IO_SEND_OP_COND (SDI/O card CMD5). The SDIO card host sends the required VDD voltage window as these commands are being performed. Cards that cannot perform data transfer in the specified range will disconnect from the bus and go to the inactive state.

By using these commands without the voltage range as the operand, the SDIO card host can query each card and determine the common voltage range before placing out-of-range cards at the inactive state. This query is used when the SDIO card host is able to select a common voltage range, or when users need to know the availability of cards.

20.3.2.4 Card Identification Process

Card identification process differs for MultiMediaCards and SD cards. For MultiMediaCards, the identification process starts at clock rate, F_{od} . All SDIO_CMD line output drivers are open-drain and allow parallel card operation during this process. The identification process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SEND_OP_COND (CMD1) and receives operation conditions.

3. The response is the “wired AND” of the operation condition registers from all cards.
4. Incompatible cards are placed at the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. The active cards simultaneously send their CID numbers serially. Cards with outgoing CID bits that do not match the bits on the command line will stop transmitting and must wait for the next identification cycle. Finally, only one card successfully transmits a full CID to the SDIO card host and enters the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to that card. This new address is called the relative card address (RCA); it is shorter than the CID and is used to address the card. At this point, the assigned card switches to the Standby state and does not react to further identification cycles, with its output switching from open-drain to push-pull.
8. The SDIO card host repeats step 5 to step 7 until it receives a timeout condition.

For the SD card, the identification process starts at clock rate, F_{od} , and all SDIO_CMD line output drives are push-pull instead of open-drain. The identification process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host broadcasts SEND_APP_OP_COND (ACMD41) command.
3. The received response is the contents of all cards’ operation condition registers.
4. The incompatible cards are placed at the inactive state.
5. The SDIO card host broadcasts ALL_SEND_CID (CMD2) to all active cards.
6. All active cards send back their unique card identification numbers (CIDs) and enter the Identification state.
7. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and is used to address the card. At this point, the assigned card switches to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.
8. The SDIO card host repeats step 5 to step 7 to all active cards.

For the SD I/O card, the identification process is accomplished as follows:

1. The bus is activated.
2. The SDIO card host sends IO_SEND_OP_COND (CMD5).
3. The received response is the contents of cards operation condition registers.
4. The incompatible cards are set to the inactive state.
5. The SDIO card host issues SET_RELATIVE_ADDR (CMD3) to an active card with an address. This new address is called the relative card address (RCA); it is shorter than the CID and is used to address the card. At this point, the assigned card switches to the Standby state. The SDIO card host can reissue this command to change the RCA. The RCA of the card is the last assigned value.

20.3.2.5 Block Write

During block write (CMD24-27) operation, the host transfers one or more blocks of data from the host to the card with a CRC code appended to the end of each block. A card supporting block write is always able to accept a block of data defined by WRITE_BL_LEN. If the CRC fails, the card indicates the failure on the SDIO_D signal line; the transferred data are discarded and not written, and all further transmitted data blocks (in multiple block write mode) are ignored.

If the host uses partial blocks, and the accumulated data length is not block aligned, then block misalignment is not allowed (the CSD parameter, WRITE_BLK_MISALIGN, is not set); the card will detect the block misalignment error before the first misaligned block (the ADDRESS_ERROR error bit is set in the status register). The write operation will also be aborted if the host tries to write a write-protected area. In this case, the card will set the WP_VIOLATION bit.

Programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents. Some cards may require long and unpredictable time 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 line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) at any time, and the card will respond with its status. The READY_FOR_DATA status bit indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing CMD7 (to select a different card), which will place the card at the Disconnect state to release the SDIO_D line(s) without interrupting the unfinished write operation. When reselecting a card, it will reactivate busy indication by pulling SDIO_D low if programming is still in progress, and the write buffer is unavailable.

20.3.2.6 Block Read

In block read mode, the basic unit of data transfer is a block with its maximum size defined in CSD (READ_BL_LEN). If READ_BL_PARTIAL is set, smaller blocks can also be transmitted; smaller blocks mean that the start and end addresses are entirely contained within one physical block as defined by READ_BL_LEN. A CRC code is appended to the end of each block to ensure data transfer integrity. CMD17 (READ_SINGLE_BLOCK) initiates a block read, and the card returns to the Transfer state after completing the transfer.

CMD18 (READ_MULTIPLE_BLOCK) initiates a read operation of several consecutive blocks.

The host can abort reading at any time within a multiple block operation, regardless of its type. Transaction abort is done by sending the stop transmission command.

If the card detects an error (for example, out of range, address misalignment, or internal error) during a multiple block read operation (both types), it stops the data transmission and remains at the data state. The host must then abort the operation by sending the stop transmission command. The read error is reported in response to the stop transmission command.

If the host sends a stop transmission command after the card transmits the last block of a multiple block operation with a predefined number of blocks, the host is responded with an illegal command, since the card is no longer at the data state. If the host uses partial blocks with which the accumulated length is not block-aligned with physical block, and block misalignment is not allowed, the card detects a block misalignment error condition at the beginning of the first misaligned block, and the ADDRESS_ERROR error bit is set in the status register.

20.3.2.7 Stream Access, Stream Write, and Stream Read (MultiMediaCard Only)

In stream mode, data is transferred in bytes, and no CRC is appended at the end of each block.

Stream Write (MultiMediaCard only)

WRITE_DAT_UNTIL_STOP (CMD20) starts the data transfer from the SDIO card host to the card, beginning at the specified address to continuously transfer until the SDIO card host issues a stop command. When partial blocks are allowed (the CSD parameter, WRITE_BL_PARTIAL, is set), the data stream can start and stop at any address within the card address space; otherwise, it can only start and stop at block boundaries. Because the amount of data to be transferred is not determined in advance, a CRC cannot be used. When the end of the memory range is reached during data transfer, even no stop command is sent by the SD card host, any additional transferred data are discarded.

The maximum clock frequency for a stream write operation is calculated with the following equation:

$$\text{Max Write Frequency} = \text{Min}\left(\text{TRANSPEED}, \frac{(8 \times 2^{\text{writeblen}}) - 100 * \text{NSAC}}{\text{TAAC} \times \text{R2WFACTOR}}\right)$$

- Max Write Frequency = The maximum of write frequency
- TRANSPEED = The maximum of data transfer rate
- writeblen = The maximum of write data block length
- NSAC = Data read access time 2 (expressed in CLK cycles)
- TAAC = Data read access time 1

- R2WFACTOR = Write speed factor

If the host attempts to use a higher frequency, the card may not be able to process the data and stop programming; at the same time, it sets the OVERRUN error bit in the status register, discards all further data transfer, and waits (at the receive data state) for a stop command. The write operation is also aborted if the host tries to write a write-protected area. In this case, the card sets the WP_VIOLATION bit.

Stream Read (MultiMediaCard only)

READ_DAT_UNTIL_STOP (CMD11) controls stream data transfers.

This command instructs the card to send its data from a specified address, until the SDIO card host sends STOP_TRANSMISSION (CMD12). The stop command has an execution delay due to the serial command transmission, and the data transfer stops after the end bit of the stop command. When the maximum of the memory range is reached during data transfer and no stop command is sent by the SDIO card host, any subsequent data sent are considered invalid.

The maximum clock frequency for a stream read operation is calculated with the following equation:

$$\text{Max Read Frequency} = \text{Min}\left(\text{TRANSPEED}, \frac{(8 \times 2^{\text{readblen}}) - 100 * \text{NSAC}}{\text{TAAC} \times \text{R2WFACTOR}}\right)$$

- Max Read Frequency = The maximum of read frequency
- TRANSPEED = The maximum of data transfer rate
- readblen = The maximum of read data block length
- NSAC = Data read access time 2 (expressed in CLK cycles)
- TAAC = Data read access time 1
- R2WFACTOR = Write speed factor

If the host attempts to use a higher frequency, the card may not be able to process the data; at the same time, it sets the UNDERRUN error bit in the status register, stops data transfer, and waits for a stop command.

20.3.2.8 Erase: Group Erase and Sector Erase

The erasable unit of the MultiMediaCard is the erase group. The erase group is measured in write blocks, which are the basic writable units of the card. The size of the erase group is a card-specific parameter and defined in CSD.

The host can erase a contiguous range of erase groups. To start the erase process, there are three steps.

First, the host defines the start address of the contiguous range with the ERASE_GROUP_START (CMD35) command; next, it defines the last address of the range with the ERASE_GROUP_END (CMD36) command, and finally, it starts the erase process by issuing the ERASE (CMD38) command. The address field in the erase commands is an erase group address in byte units. The card discards all parts that are not aligned with the erase group size, effectively rounding the address down to the erase group boundary.

If an erase command is received without following the above steps, the card sets the ERASE_SEQ_ERROR bit in the status register, resets the whole sequence, and waits for the first step.

If other commands (neither SEND_STATUS nor erase command) are received, the card sets the ERASE_RESET bit in the status register, resets the erase sequence, and executes a new command.

If the erase range includes write protected blocks, they are left intact, and only non-protected blocks are erased. At the same time, the WP_ERASE_SKIP status bit in the status register is set.

The card indicates that an erase is in progress by holding SDIO_D low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

20.3.2.9 Wide Bus Selection or Deselection

Wide bus (4-bit bus width) operation mode can be selected or deselected by using SET_BUS_WIDTH (ACMD6). The default bus width after power-up or GO_IDLE_STATE (CMD0) is 1 bit. SET_BUS_WIDTH (ACMD6) is only valid at a transfer state, which means that the bus width can be changed only after a card is selected with SELECT/DESELECT_CARD (CMD7).

20.3.2.10 Protection Management

Three write protection methods for the cards are supported in the SDIO card host module:

1. Internal card write protection (Card responsibility)
2. Mechanical write protection switch (SDIO card host module responsibility only)
3. Password-protected card lock operation

Internal card write protection

Card data can be protected against write and erase. By setting the permanent or temporary write-protect bits in CSD, the entire card can be permanently write-protected by the manufacturer or content provider. For cards that support write protection of groups of sectors by setting the WP_GRP_ENABLE bit in CSD, part of the data can be protected, and the write protection can be changed by the application. The write protection is in units of WP_GRP_SIZE sectors as specified in CSD. The SET_WRITE_PROT and CLR_WRITE_PROT commands control the 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-bit CRC code. The address field of the write protect commands is a group address in byte units.

The card ignores all LSBs below the group size.

Mechanical write protect switch

A mechanical sliding tab on the side of the card allows users to set or clear the write protection on the card. When the sliding tab is positioned with the window open, the card is write-protected; when the window is closed, the contents of the card can be changed. A switch on the socket side matched part indicates whether the card is write-protected to the SDIO card host module. The position of the write protect switch is unknown to the internal circuitry of the card.

Password protect

The password protection feature enables the SDIO card host module 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 non-volatile so that a power cycle does not erase them. Locked cards can respond to and execute certain commands, which means that the SDIO card host module is allowed to reset, initialize, select, and query for status; however, it is not allowed to access data on the card. When the password is set (i.e. the value of PWD_LEN is not 0), the card is locked automatically after power-up. As with the CSD and CID register write commands, the lock/unlock commands are available only at the transfer state. At this state, the command does not include an address argument, but 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 includes all of the required information for the command (the password setting mode, the PWD contents, and card lock/unlock). The command data block size is defined by the SDIO card host module before it sends the card lock/unlock command, and its structure is shown in [Table 20-22](#).

The bit settings are as follows:

- ERASE: Setting this bit forces an erase operation. All other bits must be 0, and only the command byte is sent.
- LOCK_UNLOCK: Setting this bit locks the card. LOCK_UNLOCK can be set simultaneously with SET_PWD, but not with CLR_PWD.
- CLR_PWD: Setting this bit clears the password data.
- SET_PWD: Setting this bit saves the password data to the memory.
- PWD_LEN: This bit defines the length of the password in bytes.
- PWD: The password (new or currently used, depending on the command)

The following sections list the command sequences to set/reset a password, lock/unlock the card, and force an erase.

Setting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7)
2. Define the block length to send in 8-bit card lock/unlock mode (SET_BLOCKLEN, CMD16), as well as the 8-bit PWD_LEN, and the byte number of the new

password. When a password replacement is done, the block size that sent with the command must take both the old and the new passwords into account.

3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line, including the 16-bit CRC. The data block includes the mode (SET_PWD = 1), the length (PWD_LEN), and the password (PWD) itself. When a password replacement is done, the length value (PWD_LEN) includes the length of both old and new passwords, 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 does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

The password length field (PWD_LEN) indicates whether a password is currently set. When this field is not 0, a password is being used, and the card locks itself after power-up. If the password is set, it is possible to lock the card immediately in the current power session or to send an additional command for card locking by setting the LOCK_UNLOCK bit.

Resetting the password

1. Select a card (SELECT/DESELECT_CARD, CMD7)
2. Define the block length to send in 8-bit card lock/unlock mode (SET_BLOCKLEN, CMD16), as well as the 8-bit PWD_LEN, and the byte number of the current password.
3. When the password is matched, the PWD field is cleared, and PWD_LEN is set to 0. When the password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the password is not changed.

Locking a card

1. Select a card (SELECT/DESELECT_CARD, CMD7)
2. Define the block length to send in 8-bit card lock/unlock mode (SET_BLOCKLEN, CMD16) (See byte 0 in [Table 20-22](#)), as well as the 8-bit PWD_LEN, and the byte number of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line, including the 16-bit CRC. The data block includes the mode (LOCK_UNLOCK = 1), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is locked, and the CARD_IS_LOCKED status bit is set in the card status register. When the password sent does not correspond (in size and/or content) to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the lock fails.

It is possible to set the password and to lock the card in the same sequence. In this case, the host module of SDIO card performs all the required steps mentioned above for setting the password, but it is necessary to set the LOCK_UNLOCK bit in Step 3 when the new password command is sent.

When 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 will fail, and the LOCK_UNLOCK_FAILED error bit will be set in the card status register.

Unlocking the card

1. Select a card (SELECT/DESELECT_CARD, CMD7)
2. Define the block length to send in 8-bit card lock/unlock mode (SET_BLOCKLEN, CMD16) (See byte 0 in [Table 20-22](#)), as well as the 8-bit PWD_LEN, and the byte number of the current password.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data block size on the data line, including the 16-bit CRC. The data block includes the mode (LOCK_UNLOCK = 0), the length (PWD_LEN), and the password (PWD) itself.
4. When the password is matched, the card is unlocked, and the CARD_IS_LOCKED

status bit is cleared in the card status register. When the password sent is not correct (in size and/or content) and does not correspond to the expected password, the LOCK_UNLOCK_FAILED error bit is set in the card status register, and the card remains locked.

The unlocking function is only valid for the current power session. If the PWD field is not cleared, the card is locked automatically at the next power-up.

An attempt to unlock an unlocked card will fail, and the LOCK_UNLOCK_FAILED error bit will be set in the card status register.

Forcing erase

If users forget the password (PWD content), it is possible to access the card after clearing all the data on the card. This forced erase operation erases all card data and all password data.

1. Select a card (SELECT/DESELECT_CARD, CMD7)
2. Set the block length (SET_BLOCKLEN, CMD16) to 1 byte. Only the 8-bit card lock/unlock byte (See byte 0 in [Table 20-22](#)) is sent.
3. Send LOCK/UNLOCK (CMD42) with the appropriate data byte on the data line, including the 16-bit CRC. The data block indicates the mode (ERASE = 1). All other bits must be 0.
4. When the ERASE bit is the only bit set in the data field, all card contents are erased, including the PWD and PWD_LEN fields; the card is no longer locked. When any other bits are set, the LOCK_UNLOCK_FAILED error bit will be set in the card status register; the card retains all of its data and remains locked.

An attempt to use a force erase on an unlocked card will fail, and the LOCK_UNLOCK_FAILED error bit will be set in the card status register.

20.3.2.11 Card Status Register

The response format R1 contains a 32-bit card status field. This field is intended to transmit the card status information (which may be stored in a local status register) to the host. If not specified, the status entries are always related to the previously issued command.

[Table 20-9](#) defines the different entries of the status. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: Error bit
- S: Status bit
- R: Detection bit, set according to the actual command response.
- X: Detection bit, set during command execution. The SDIO card host must poll the card status by issuing the status command to read these bits.

Clear condition:

- A: According to the current card state
- B: Always related to the previous command. Reception of a valid command clears it (with a delay of one command).
- C: Cleared by read

Table 20-9 Card Status

Bit	Identifier	Type	Value	Description	Clear Condition
31	ADDRESS_OUT_OF_RANGE	ERX	'0' = No error '1' = Error	The command address argument is out of the allowed range for this card. A multiple block or stream read/write operation (although starts at a valid address) attempts to read or write beyond the card capacity.	C

Bit	Identifier	Type	Value	Description	Clear Condition
30	ADDRESS_MISALIGN		'0' = No error '1' = Error	The command address argument (in accordance with the current block length) positions the first data block misaligned to the card physical blocks. A multiple block read/write operation (although starts at a valid address) attempts to read or write a data block not aligned with the physical blocks of the card.	C
29	BLOCK_LEN_ERROR		'0' = No error '1' = Error	Either the argument of a SET_BLOCKLEN command exceeds the maximum value allowed for the card, or the previously defined block length is illegal for the current command. (E.g. The host issues a write command, the current block length is smaller than the minimum value allowed for the card, and it is not allowed to write partial blocks.)	C
28	ERASE_SEQ_ERROR		'0' = No error '1' = Error	An error in the sequence of erase commands occurs.	C
27	ERASE_PARAM	EX	'0' = No error '1' = Error	An invalid erase group is selected for erase.	C
26	WP_VIOLATION	EX	'0' = No error '1' = Error	Attempt to program a write-protected block	C
25	CARD_IS_LOCKED	SR	'0' = Card unlocked '1' = Card locked	When this bit is set, it indicates that the card is locked.	A
24	LOCK_UNLOCK_FAILED	EX	'0' = No error '1' = Error	A sequence error or a password error is detected in lock/unlock card command.	C
23	COM_CRC_ERROR	ER	'0' = No error '1' = Error	The CRC check fails in the previous command.	B
22	ILLEGAL_COMMAND	ER	'0' = No error '1' = Error	Command is illegal for the current card state.	B
21	CARD_ECC_FAILED	EX	'0' = Success '1' = Failure	Card internal ECC is applied, but it fails when correcting the data.	C
20	CC_ERROR	ER	'0' = No error '1' = Error	(Undefined by the standard) A card error occurs, and is not related to the host command.	C
19	ERROR	EX	'0' = No error '1' = Error	(Undefined by the standard) A card internal error related to the execution of the last host command (e.g. read or write failures) occurs.	C
18	Reserved				
17	Reserved				
16	CID/CSD_OVERWRITE	EX	'0' = No error '1' = Error	It can be either of the following errors: ▪ The CID register is already written and cannot be overwritten. ▪ The read-only section of the CSD does not match the card contents. ▪ An attempt to reverse the copy or permanent write protection is made. i.e. Set as original or remove write protection.	C
15	WP_ERASE_SKIP	EX	'0' = Unprotected '1' = Protected	Set when only partial address space is erased due to existing write-protected data block.	C
14	CARD_ECC_DISABLED	SX	'0' = Allowed '1' = Not allowed	The command is executed without using the internal ECC.	A
13	ERASE_RESET		'0' = Cleared '1' = Set	An erase sequence is stopped before execution because an out of erase sequence command (commands other than CMD35, CMD36, CMD38, or CMD13) is received.	C

Bit	Identifier	Type	Value	Description	Clear Condition
12:9	CURRENT_STATE	SR	0 = Idle 1 = Ready 2 = Identification 3 = Standby 4 = Transmit 5 = Data 6 = Receive 7 = Program 8 = Disconnect 9 = Busy test 10 ~ 15 = Reserved	The state of the card when receiving command If the command execution causes a state change, it will be visible to the host in the response of the next command. The four bits are interpreted as a decimal number between 0 and 15.	B
8	READY_FOR_DATA	SR	'0' = Not ready '1' = Ready	Correspond to buffer empty signal on the bus	
7	SWITCH_ERROR	EX	'0' = No error '1' = Switch error	The card does not switch to the expected mode as requested by the SWITCH command.	B
6	Reserved				
5	APP_CMD	SR	'0' = Allowed '1' = Not allowed	The card expects ACMD, or the indication command is interpreted as ACMD.	C
4	Reserved for SD I/O card				
3	AKE_SEQ_ERROR	ER	'0' = No error '1' = Error	Errors in the sequence of the authentication process occur.	C
2	Reserved for application specific commands				
1,0	Reserved for manufacturer test mode				

20.3.2.12 SD Status Register

The SD status contains status bits that are related to the SD memory card proprietary features and status bits that may be used for future applications. The size of the SD status is one 512-bit data block. The contents of this register are transmitted to the SDIO card host after ACMD13 is received (CMD55 followed with CMD13). ACMD13 can be sent only when a card is at transfer state (card is selected).

[Table 20-10](#) defines the different entries of the SD status register. The type and clear condition fields in the table are abbreviated as follows:

Type:

- E: Error bit
- S: Status bit
- R: Detection bit, set according to the actual command response.
- X: Detection bit, set during command execution. The SDIO card host must poll the card status by issuing the status command to read these bits.

Clear condition:

- A: According to the current card state
- B: Always related to the previous command. Reception of a valid command clears it (with a delay of one command).
- C: Cleared by read

Table 20-10 SD Status

Bit	Identifier	Type	Value	Description	Clear Condition
511:510	DAT_BUS_WIDTH	SR	'00' = 1 (Default) '01' = Reserved '10' = 4-bit wide '11' = Reserved	The current data bus width defined by SET_BUS_WIDTH command	A
509	SECURED_MODE	SR	'0' = Not in secured mode '1' = In secured mode	Card is in secured mode operation (Please refer to the "SD Security Specification").	A
508:496	Reserved				
495:480	SD_CARD_TYPE	SR	'00xxh' = SD Memory Card as defined in Physical Spec Ver. 1.01 ~ 2.00 ('x' means any value) The defined cards include: '0000' = Regular SD R/W Card '0001' = SD ROM Card	The 8 LSBs will be used to define different variations of an SD memory card in the future (Each bit will define different SD types). The 8 MSBs will be used to define SD Cards that do not comply with current SD physical layer specification.	A
479:448	SIZE_OF_PROTECTED_AREA	SR	Size of protected area (Please see the description below.)	(Please see the description below)	A
447:440	SPEED_CLASS	SR	Speed class of the card (Please see the description below.)	(Please see the description below.)	A
439:432	PERFORMANCE_MOVE	SR	Performance of move indicated in 1 MB/s (Please see the description below.)	(Please see the description below.)	A
431:428	AU_SIZE	SR	Size of AU (Please see the description below.)	(Please see the description below.)	A
427:424	Reserved				
423:408	ERASE_SIZE	SR	Number of AUs to be erased at a time	(Please see the description below.)	A
407:402	ERASE_TIMEOUT	SR	Timeout value for erasing areas specified by UNIT_OF_ERASE_AU	(Please see the description below.)	A
401:400	ERASE_OFFSET	SR	Fixed offset value added to erase time	(Please see the description below.)	A
399:312	Reserved				
311:0	Reserved for manufacturer				

SIZE_OF_PROTECTED_AREA

Standard-capacity card and high-capacity card set this field in different ways. In the case of a standard-capacity card, the capacity of protected area is calculated as follows:

$$\text{Protected area} = \text{SIZE_OF_PROTECTED_AREA} * \text{MULT} * \text{BLOCK_LEN}$$

SIZE_OF_PROTECTED_AREA is specified by the unit in MULT*BLOCK_LEN.

In the case of a high-capacity card, the capacity of protected area is calculated as follows:

$$\text{Protected area} = \text{SIZE_OF_PROTECTED_AREA}$$

SIZE_OF_PROTECTED_AREA is specified in bytes.

SPEED_CLASS

This 8-bit field indicates the speed class and the value that can be calculated by $P_w/2$ (where PW is the write performance).

Table 20-11 Speed Class Code Field

SPEED_CLASS	Value Definition
00h	Type 0
01h	Type 2
02h	Type 4
03h	Type 6
04h ~ FFh	Reserved

PERFORMANCE_MOVE

This 8-bit field indicates Pm (Performance move) set in unit of 1 MB/sec. If the card does not move data with RUs (recording units), Pm should be regarded as infinity. Setting the field to FFh indicates infinity.

Table 20-12 Performance Move Field

PERFORMANCE_MOVE	Value Definition
00h	Undefined
01h	1 MB/sec
02h	2 MB/sec
.....
FEh	254 MB/sec
FFh	Infinite

AU_SIZE

This 4-bit field indicates the AU size, and the value can be selected in the power of 2 base from 16 KB.

Table 20-13 AU_SIZE Field

AU_SIZE	Value Definition
00h	Undefined
01h	16 KB
02h	32 KB
03h	64 KB
04h	128 KB
05h	256 KB
06h	512 KB
07h	1 MB
08h	2 MB
09h	4 MB
Ah ~ Fh	Reserved

The maximum AU size, which depends on the card capacity, is defined in Table 20-14. The card can be set to any AU size between the RU size and the maximum of AU size.

Table 20-14 The Maximum of AU Size

Capacity	16 MB ~ 64 MB	128 MB ~ 256 MB	512 MB	1 GB ~ 32 GB
The Max. of AU Size	512 KB	1 MB	2 MB	4 MB

ERASE_SIZE

This 16-bit field indicates N_{ERASE} . When N_{ERASE} numbers of AUs are erased, the timeout value is specified by [ERASE_TIMEOUT](#). The host should determine the proper number of AUs to be erased in one operation so that the host can show the progress of the erase operation. If this field is set to 0, the erase timeout calculation is not supported.

Table 20-15 ERASE_SIZE Field

ERASE_SIZE	Value Definition
0000h	Erase timeout calculation is not supported.
0001h	1 AU
0002h	2 AUs
0003h	3 AUs
.....
FFFFh	65535 AUs

ERASE_TIMOUT

This 6-bit field indicates T_{ERASE} , and the value indicates the erase timeout from offset when multiple AUs are being erased as specified by ERASE_SIZE. The range of ERASE_TIMEOUT can be defined up to 63 seconds, and the card manufacturer can choose appropriate combination of ERASE_SIZE and ERASE_TIMEOUT, depending on the actual implementation. ERASE_TIMEOUT is determined first and then the ERASE_SIZE.

Table 20-16 Erase Timeout Field

ERASE_TIMOUT	Value Definition
00	Erase timeout calculation is not supported.
01	1 second
02	2 seconds
03	3 seconds
.....
63	63 seconds

ERASE_OFFSET

This 2-bit field indicates T_{OFFSET} . This field is meaningless if the ERASE_SIZE and ERASE_TIMEOUT fields are both set to 0.

Table 20-17 Erase Offset Field

ERASE_OFFSET	Value Definition
0	0 second
1	1 second
2	2 seconds
3	3 seconds

20.3.2.13 SD I/O Mode**SD I/O interrupts**

To allow the SD I/O card to interrupt the MultiMediaCard/SD module, an interrupt function is available on a pin of the SD interface: Pin 8. Pin 8 is used as SDIO_D1 when operating in the 4-bit SD mode, and the card signals interrupt to the MultiMediaCard/SD module with this pin. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level sensitive, which means that the interrupt line must be held active (low) before it is recognized and obtains response from the MultiMediaCard/SD module; it is held inactive (high) after the interrupt period is completed. After the MultiMediaCard/SD module serves the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the

SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the MultiMediaCard/SD must provide pull-up resistors on all data lines (SDIO_D[3:0]). The MultiMediaCard/SD module samples the level of pin 8 (SDIO_D/IRQ) and performs interrupt detection during interrupt period; during other time periods, values of the signal line are ignored.

The interrupt period is applicable for both memory and I/O operations. The definition of the interrupt period for single block operation is different from that of multiple block data transfers.

SD I/O suspend and resume

Within a multifunction SD I/O card or a card with both I/O and memory functions, multiple devices (I/O and memory) share the MMC/SD bus. To allow devices in the MMC/SD module to share the bus, SD I/O and combo cards optionally implement the suspend/resume mechanism. If a card supports suspend/resume, the MMC/SD module can temporarily halt a data transfer operation of one function or memory (suspend), to free the bus for a higher-priority transfer of a different function or memory. After this higher-priority transfer is completed, the original suspended transfer is resumed. Support of suspend/resume is optional on a per-card basis. To perform the suspend/resume operation on the MMC/SD bus, please refer to the following steps:

1. Determine the current function of the SDIO_D[3:0] line(s)
2. Request the lower-priority or slower transaction to suspend
3. Wait for the suspension transaction to complete and ensure that the device is suspended
4. Begin the higher-priority transaction
5. Wait for the completion of the higher-priority transaction
6. Resume the suspended transaction

SDI/O Read wait

The optional read wait (RW) operation is applicable only for SD card 1-bit and 4-bit modes. The read wait operation allows the MMC/SD module to request a card to temporarily stall the data transfer when it is reading multiple registers (IO_RW_EXTENDED, CMD53), allowing the MMC/SD module to send commands to other functions within the SD I/O device. To determine whether a card supports the read wait protocol, the MMC/SD module must test the internal registers of the card. The timing for read wait is based on the interrupt period.

20.3.2.14 Command and Response

Application-specific and general commands

The SD card host module system provides a standard interface for a variety of applications types, and it should fulfill specific customer/application requirements at the same time. Therefore, two types of generic commands are defined in the standard: application-specific commands (ACMD) and general commands (GEN_CMD).

When the card receives the APP_CMD (CMD55) command, the card expects the next command to be an application-specific command. ACMDs have the same structure as regular MultiMediaCard commands and can have the same CMD number. The card recognizes it as ACMD because it appears after APP_CMD (CMD55). When the command immediately following the APP_CMD (CMD55) is not a defined application-specific command, it is recognized as a standard command. For example, for a SD_STATUS (ACMD13) application-specific command, if CMD13 is received immediately after APP_CMD (CMD55), then it is interpreted as SD_STATUS (ACMD13); however, when the card receives CMD7 after APP_CMD (CMD55), but the card does not define ACMD7, then it is interpreted as a standard (SELECT/DESELECT_CARD) CMD7.

To use manufacturer-specific ACMDs, the SD card Host must perform the following steps:

1. Send APP_CMD (CMD55) command

The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set, and an ACMD is now expected.

2. Send the required ACMD

The card responds to the MultiMediaCard/SD module, indicating that the APP_CMD bit is set and that the accepted command is interpreted as an ACMD. When a non-ACMD is sent, it is handled by the card as a normal MultiMediaCard command, and the APP_CMD bit in the card status register is cleared.

When an invalid command is sent (neither ACMD nor CMD), it is handled as a standard MultiMediaCard illegal command error.

The bus transaction for a GEN_CMD is the same as the single-block read or write commands (WRITE_BLOCK, CMD24 or READ_SINGLE_BLOCK, CMD17). In this case, the argument denotes the direction of the data transfer rather than the address, and the data block has user-specific format and meaning.

The card must be selected (at transfer state) before sending GEN_CMD (CMD56). The data block size is defined by SET_BLOCKLEN (CMD16). The response to GEN_CMD (CMD56) is in R1b format.

Command types

Both application-specific and general commands include the following four types:

1. Broadcast command (BC): Sent to all cards; no responses returned.
2. Broadcast command with response (BCR): Sent to all cards; responses received from all cards simultaneously.
3. Addressed (point-to-point) command (AC): Sent to the selected card; data transfer is not included on the SDIO_D line(s).
4. Addressed (point-to-point) data transfer command (ADTC): Sent to the selected card; data transfer is included on the SDIO_D line(s).

Command formats

Please refer to [Table 20-2](#) for command formats.

Commands for the MultiMediaCard/SD module

Table 20-18 Block-oriented Write Command

CMD Index	Type	Argument	Response format	Abbreviation	Description
CMD23	ac	[31:16] = 0 [15:0] = Number of blocks	R1	SET_BLOCK_COUNT	Define the number of blocks to be transferred in the following multiple-block read or write command.
CMD24	adtc	[31:0] = Data address	R1	WRITE_BLOCK	Write a block based on the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] = Data address	R1	WRITE_MULTIPLE_BLOCK	Continuously write blocks of data until receiving a STOP_TRANSMISSION or reaching the requested number of blocks.
CMD26	adtc	[31:0] = Stuff bits	R1	PROGRAM_CID	Program the card identification register. This command can be issued only once per card. The card has hardware mechanism to prevent multiple programming operations. Normally, this command is reserved for manufacturer.
CMD27	adtc	[31:0] = Stuff bits	R1	PROGRAM_CSD	Program the programmable bits in the CSD of the card.

Table 20-19 Block-oriented Write Protection Command

CMD Index	Type	Argument	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 addressed group. The write protection properties are set in the card-specific data field (WP_GRP_SIZE).
CMD29	ac	[31:0] = Data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] = Write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				

Table 20-20 Erase Command

CMD Index	Type	Argument	Response format	Abbreviation	Description
CMD32 ... CMD34		Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCard.			
CMD35	ac	[31:0] = Data address	R1	ERASE_GROUP_START	Set the address of the first erase group within a range selected for erase.
CMD36	ac	[31:0] = Data address	R1	ERASE_GROUP_END	Set the address of the last erase group within a continuous range selected for erase.
CMD37		Reserved. These command indexes cannot be used in order to maintain backward compatibility with older versions of the MultiMediaCard.			
CMD38	ac	[31:0] = Stuff bits	R1b	ERASE	Erase all previously selected write blocks.

Table 20-21 I/O Mode Command

CMD Index	Type	Argument	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 addresses a card and a register, providing the data written if the write flag is set. The R4 response contains data read from the addressed register. This command accesses application-related registers not defined in the MultiMediaCard standard.
CMD40	bcr	[31:0] = Stuff bits	R5	GO_IRQ_STATE	Place the system in the interrupt mode.
CMD41		Reserved			

Table 20-22 Lock Command

CMD Index	Type	Argument	Response format	Abbreviation	Description
CMD42	adtc	[31:0] = Stuff bits	R1	LOCK_UNLOCK	Set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43 ... CMD54		Reserved			

Table 20-23 Application-specific Command

CMD Index	Type	Argument	Response format	Abbreviation	Description
CMD55	ac	[31:16] = RCA [15:0] = Stuff bits	R1	APP_CMD	Indicate 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 get a data block from the card for general purpose/application-specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57 ... CMD59		Reserved			
CMD60 ... CMD63		Reserved for manufacturer			

20.3.3 Response Format

All responses are sent via the MCCMD command line, SDIO_CMD. The response transmission always starts with the far left bit of the string corresponding to the response code word. The code length depends on the response type.

A response always starts with a start bit (always 0), followed by the transmission direction bit (card = 0). A value denoted by X in Table 20-24 ~ Table 20-30 indicates a variable entry. All responses, except for the R3 response type, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

There are five types of responses. Their formats are defined as follows:

20.3.3.1 R1 (Normal Response Command)

Code length = 48 bits. The bits [45:40] indicate the index of the command to be responded to; its value falls between 0 and 63. The status of the card is coded in 32 bits.

Table 20-24 R1 Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	X	Command index
[39:8]	32	X	Card status
[7:1]	7	X	CRC7
0	1	1	End bit

20.3.3.2 R1b

This is identical to R1 format, but 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.

20.3.3.3 R2 (The CID and 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 CMD9. Only the bits [127:1] of the CID and CSD registers are transferred, the reserved bit [0] of these receive registers is replaced by the end bit of the response. The card indicates that an erase is in progress by holding MCDAT low. The actual erase time may be quite long, and the host may issue CMD7 to deselect the card.

Table 20-25 R2 Response

Bit	Width (Bit)	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	'111111'	Command index
[127:1]	127	X	Card status
0	1	1	End bit

20.3.3.4 R3 (The OCR Register)

Code length: 48 bits. The contents of the OCR register are sent as a response to CMD1. The level code is defined as: Restricted voltage windows = low, card busy = low.

Table 20-26 R3 Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit

[45:40]	6	'111111'	Reserved
[39:8]	32	X	OCR register
[7:1]	7	'1111111'	Reserved
0	1	1	End bit

20.3.3.5 R4 (Fast I/O)

Code length: 48 bits. The argument field contains the RCA of the addressed card, the register address to be read out or written to, and its content.

Table 20-27 R4 Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'100111'	CMD39
[39:8] Argument field	[31:16]	16	X
	[15:8]	8	X
	[7:0]	8	X
	[7:1]	7	X
0	1	1	End bit

20.3.3.6 R4b

For SD I/O only: An SDIO card receiving the CMD5 will respond with a unique SDIO response R4.

Table 20-28 R4b Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	X	Reserved
[39:8] Argument field	39	1	Card is ready
	[38:36]	3	Number of I/O functions
	35	1	Current memory
	[34:32]	3	Stuff bit
	[31:8]	24	I/O OCR
	[7:1]	7	Reserved
0	1	1	End bit

Once an SD I/O card receives a CMD5, the I/O part of the card is enabled to respond normally to all further commands. This enabled state of the I/O function within the card will remain set until the next reset, power-off, or CMD52 of I/O reset is received. Please note that an SD memory-only card may respond to a CMD5. The proper response for a memory-only card is: Current memory = 1 and Number of I/O functions = 0. A memory-only SD card built to meet the SD Memory Card Specification version 1.0 can detect the CMD5 as an illegal command and do not respond. The I/O-aware host will send CMD5. If the card responds with response R4, the host determines the card's configuration based on the data contained within the R4 response.

20.3.3.7 R5 (Interrupt Request)

Only for MultiMediaCard. Code length: 48 bits. If the response is generated by the host, the RCA field in the argument will be 0x0.

Table 20- 29 R5 Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'101000'	CMD40
[39:8] Argument field	[31:16]	16	X RCA[31:16] of winning card or host
	[15:0]	16	X Undefined; can be used as interrupt data.
[7:1]	7	X	CRC7
0	1	1	End bit

20.3.3.8 R6 (Interrupt Request)

Only for SD I/O. This is the normal response to CMD3 by a memory device.

Table 20-30 R6 Response

Bit	Width (Bit)	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	'000011'	CMD3
[39:8] Argument field	[31:16]	16	X RCA[31:16] of winning card or host
	[15:0]	16	X Card status
[7:1]	7	X	CRC7
0	1	1	End bit

The card status bits [23:8] are changed when 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

20.3.4 SDIO I/O Card-specific Operation

The following features are SD I/O-specific operations:

- SDIO read wait operation achieved by SDIO_D2 signaling
- SDIO read wait operation achieved by stopping the clock
- SDIO suspend/resume operation (write and read suspend)
- SDIO interrupts

SDIO supports these operations only if the SDIO_DCTRL[11] bit is set, except for read suspend, which does not need specific hardware implementation.

20.3.4.1 SDIO I/O Read Wait Operation by SDIO_D2 Signaling

Read wait interval can be started before the first block is received; when the data path is enabled (the SDIO_DTCTRL[0] bit set), the SDIO-specific operation is enabled (the SDIO_DTCTRL[11] bit set), read wait starts (SDIO_DTCTRL[10] =0 and SDIO_DTCTRL[10] =1), and data direction is from card to SDIO (SDIO_DTCTRL[1] = 1), DPSM will directly move from Idle to read wait. At read wait state, DPSM drives SDIO_D2 to 0 after 2 SDIO_CK clock cycles. At this state, if the RDWTSTOP bit (SDIO_DTCTRL [9]) is set, DPSM remains at wait state for two more SDIO_CK clock cycles and drives SDIO_D2 to '1' for one clock cycle (in accordance with SDIO specification). DPSM then starts waiting again until it receives data from the card. DPSM will not start a read wait interval while receiving a block even if read wait start is set; the read wait interval will start after the CRC is received. The RDWTSTOP bit should be cleared to start a new read wait operation. During the read wait interval, the SDIO host can detect SDIO interrupts on SDIO_D1.

20.3.4.2 SDIO Read Wait Operation by Stopping SDIO_CK

If the SDIO card does not support the previous read wait method, SDIO can perform a read wait by stopping SDIO_CK (Set the SDIO_DTCTRL based on the method presented in [Section 20.3.4.1](#), but SDIO_DTCTRL [10] =1). Two SDIO_CK cycles after the end bit of the current block is received, DSPM stops the clock and starts the clock again after the read wait start bit is set.

As SDIO_CK is stopped, no command can be issued to the card. During a read/wait interval, the SDIO host can detect SDIO interrupts on SDIO_D1.

20.3.4.3 SDIO Suspend/Resume Operation

While sending data to the card, SDIO can suspend the write operation. The SDIO_CMD[11] bit is set and indicates to CPSM that the current command is a suspend command. CPSM analyzes the response, and when the ACK is received from the card (suspend accepted), it goes idle after acknowledging the CRC token of the current block received.

The hardware does not save the number of the remaining blocks to be sent after the suspended operation is over.

The write operation can be suspended by software; disable DPSM (SDIO_DTCTRL[0] = 0) when the ACK of suspend command is received from the card. Then, DPSM enters the idle state.

To suspend a read operation: DPSM waits at the Wait_r state; a complete data packet is sent just before stopping the data transaction. The application continues reading RxBUF until the BUF is empty, and DPSM goes idle automatically.

20.3.4.4 SDIO Interrupts

SDIO interrupts are detected on the SDIO_D1 line by the host once the SDIO_DCTRL[11] bit is set.

20.3.5 CE-ATA Specific Operation

The following features are CE-ATA specific operations:

- Sending the command completion signal disables the CE-ATA device.
- Receiving the command completion signal from the CE-ATA device
- Signaling the completion of the CE-ATA command to the CPU with the status bit and/or interrupt

The SDIO host supports these operations only for the CE-ATA CMD61 command, that is, only if SDIO_CMD[14] is set.

20.3.5.1 Command Completion Signal Disable

If the "enable CMD completion" bit, SDIO_CMD[12], is not set, and the "non-interrupt enable" bit, SDIO_CMD[13], is set, then command completion signal disable will be sent after 8-bit cycles after the reception of a short response .

After loading the command shift register with the disable sequence '00001' and the command counter with 43, CPSM enters the Pend state. After eight cycles, a trigger moves CPSM to the Send state. When the command counter reaches 48, CPSM becomes idle as no response is awaited.

20.3.5.2 Command Completion Signal Enable

If the 'enable CMD completion' bit in SDIO_CMD[12] is set, and the 'non-interrupt enable' bit in SDIO_CMD[13] is set, CPSM waits for the command completion signal at the Waitcpl state.

When '0' is received on the CMD line, CPSM enters the Idle state. No new command can be sent within the 7 cycles. Then, for the last 5 cycles (out of the previously-mentioned 7 cycles), the CMD line is driven to '1' in push-pull mode.

20.3.5.3 CE-ATA Interrupt

The command completion is signaled to the CPU by the status bit, SDIO_STA[23]. This static bit can be cleared with the clear bit, SDIO_INTCLR[23].

The SDIO_STS[23] status bit can generate an interrupt on each interrupt line, depending on the configuration of the mask bit, SDIO_INTENx[23].

20.3.5.4 Aborting CMD61

If the "command completion disable" signal is not yet sent, but CMD61 needs to be aborted, the command state machine must be disabled. It then becomes idle, and can send the CMD12 command. No "command completion disable" signal is sent during this operation.

20.3.6 Hardware Flow Control

The HW flow control functionality can avoid BUF underrun (Tx mode) and overrun (Rx mode) errors.

The operation is to stop SDIO_CK and freeze SDIO state machines. The data transfer is stalled when the BUF is unable to transmit or receive data. Only state machines clocked by SDIOCLK are frozen, and the AHB interface is still working. The BUF can still be read or written even if flow control is activated.

To enable HW flow control, the SDIO_CLKCTRL[14] bit must be set to '1'. After reset, HW flow control is disabled.

20.4 SDIO Registers

Devices can communicate with the system via the 32-bit control register operated on AHB.

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

Table 20-31 SDIO Register Map

Offset	register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	SDIO_POWER	Reserved																									PWRCTRL	0	0						
	0x00000000																																		
0x04	SDIO_CLKCTRL	Reserved												CLKPSC[9:8]		FLWCTRLEN		CLKEDG		BUSWIDTH		BYPSS		PWRSVG		CLKEN		CLKPSC[7:0]							
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x08	SDIO_ARG	ARG[31:0]																																	
	0x00000000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x0C	SDIO_CMD	Reserved												ATACMD		INTDIS		CMPLSGNLEN		SDIOSUSP		CMDMEN		PNDWT		INTWT		RSPWT[1:0]		CMDIDX[5:0]					
	0x00000000													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
0x10	SDIO_RSPCMD	Reserved																																	
	0x00000000																																		

20.4.1 SDIO Power Control Register (SDIO_POWER)

Address offset: 0x00

Reset value: 0x00000000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
res															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															
res															
Bit 31:2		Reserved. Always read as 0.													
Bit 1:0		PWRCTRL: Power supply control bits These bits are used to define the current functional state of the card clock: 00: Power-off: The clock to card is stopped. 01: Reserved 10: Reserved power-up 11: Power-on: The card is clocked.													

Note: At least seven HCLK clock periods are needed between two write accesses to this register.

20.4.2 SDIO Clock Control Register (SDIO_CLKCTRL)

Address offset: 0x04

Reset value: 0x00000000

The SDIO_CLKCTRL register controls the SDIO_CK output clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															CLKPSC[9]
res															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLK PSC[8]	FLW CTR LEN	CLK EDG	BUSWIDTH		BYP S	PWR SVG	CLK EN	CLKPSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:17		Reserved. Always read as 0.													
Bit 16:15		CLKPSC[9:8]: Clock divide factor (2 MSBs) This bit field defines the clock divide factor between input clock (SDIOCLK) and output clock (SDIO_CK): SDIO_CK frequency = SDIOCLK/[CLKPSC[9:0] + 2]													
Bit 14		FLWCTRLLEN: Hardware flow control enable 0: Hardware flow control is disabled. 1: Hardware flow control is enabled. When hardware flow control is enabled, please refer to Section 20.4.11 for the definitions of TXBUF_E and RXBUF_F interrupt signal.													
Bit 13		CLKEDG: SDIO_CK dephasing selection bit 0: SDIO_CK is generated on the rising edge of the master clock SDIOCLK. 1: SDIO_CK is generated on the falling edge of the master clock SDIOCLK.													
Bit 12:11		BUSWIDTH: Wide bus mode enable bit 00: Default bus mode; SDIO_D0 is used. 01: 4-bit bus mode; SDIO_D[3:0] is used. 10: 8-bit bus mode; SDIO_D[7:0] is used.													

Bit 10	BYPS: Clock divider bypass enable bit 0: Disable bypass: SDIOCLK is divided according to the CLKPSC value before driving the SDIO_CK output signal. 1: Enable bypass: SDIOCLK directly drives the SDIO_CK output signal.
Bit 9	PWRSVG: Power saving configuration bit To save power, setting the PWRSVG bit can disable SDIO_CK clock output when the bus is idle. 0: Always output SDIO_CK 1: Output SDIO_CK only when the bus is active.
Bit 8	CLKEN: Clock enable bit 0: SDIO_CK is disabled. 1: SDIO_CK is enabled.
Bit 7:0	CLKPSC[7:0]: Clock divide factor (8 LSBs) This bit field defines the clock divide factor between input clock (SDIOCLK) and output clock (SDIO_CK): $\text{SDIO_CK frequency} = \text{SDIOCLK}/[\text{CLKPSC}[9:0] + 2]$

- Note:
1. When the SD/SDIO card or MultiMediaCard is in identification mode, the SDIO_CK frequency must be less than 400 kHz.
 2. The clock frequency can be changed to the maximum card bus frequency allowed when relative card addresses are assigned to all cards.
 3. At least seven HCLK clock periods are needed between two write accesses to this register. SDIO_CK can also be stopped during the read wait interval for SD I/O cards; in this case, the SDIO_CLKCTRL register does not control SDIO_CK.

20.4.3 SDIO Argument Register (SDIO_ARG)

Address offset: 0x08

Reset value: 0x00000000

The SDIO_ARG register contains a 32-bit command argument, which is sent to a card as part of a command message.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARG															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARG															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:0				ARG: Command argument Command argument is sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.											

20.4.4 SDIO Command Register (SDIO_CMD)

Address offset: 0x0C

Reset value: 0x00000000

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 message. The command type bits control the command path state machine (CPSM).

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserve d	ATAC MD	INTDI S	CMPL SGNL EN	SDIO SUSP	CMD MEN	PND WT	INTW T	RSPWT	CMDIDX							
res	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:15		Reserved. Always read as 0.														
Bit 14		ATACMD: CE-ATA command If this bit is set, CPSM transfers CMD61.														
Bit 13		INTDIS: Interrupt disable If this bit is not set, interrupts of CE-ATA devices are enabled.														
Bit 12		CMPLSGNLEN: Enable CMD completion If this bit is set, the command completion signal is enabled.														
Bit 11		SDIOSUSP: SD I/O suspend command If this bit is set, the command to be sent is a suspend command (only used for SDIO card).														
Bit 10		CMDMEN: Command path state machine (CPSM) enable bit If this bit is set, CPSM is enabled.														
Bit 9		PNDWT: CPSM waits for ends of data transfer (CmdPending internal signal) If this bit is set, CPSM waits for the end of data transfer before it starts sending a command.														
Bit 8		INTWT: CPSM waits for interrupt request If this bit is set, CPSM disables command timeout and waits for an interrupt request.														
Bit 7:6		RSPWT: Wait for response bits The 2 bits indicate whether CPSM waits for response; if yes, the type of response is indicated. 00: No response. Expect the CMDCMPL flag. 01: Short response. Expect the CMDRSPCMPL or CMDFAIL flag. 10: No response. Expect the CMDCMPL flag. 11: Long response. Expect the CMDRSPCMPL or CMDFAIL flag.														
Bit 5:0		CMDIDX: Command index The command index is sent to the card as part of a command message.														

- Note:
- At least seven HCLK clock periods are needed between two write accesses to this register.
 - MultiMediaCards can send two kinds of responses: 48-bit short response, or 136-bit long responses. SD card and SD I/O card can send only short responses, and the argument can vary according to the type of response. Software will distinguish the type of response according to the sent command. CE-ATA devices send only short responses.

20.4.5 SDIO Command Response Register (SDIO_RSPCMD)

Address offset: 0x10

Reset value: 0x00000000

The SDIO_RSPCMD register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long response or OCR response), the RSPCMD field is unknown, although it should contain 111111b (the value of the reserved field from the response).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								RSPCMD							
res								r	r	r	r	r	r	r	r
Bit 31:6		Reserved. Always read as 0.													
Bit 5:0		RSPCMD: Response command index Read-only bit field, which contains the command index of the last command response received.													

20.4.6 SDIO Response 1...4 Register (SDIO_RSPx)

Address offset: $0x14 + 4 * (x - 1)$, where $x = 1 \dots 4$

Reset value: 0x00000000

The SDIO_RSP1/2/3/4 register contains the card state; that is, part of the response information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CARDSTSx															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CARDSTSx															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:0		CARDSTSx: Please refer to Table 20-32.													

Based on response status, the card status size can be 32 bits or 127 bits.

Table 20-32 Response Types and the 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 LSB of the SDIO_RSP4 register is always 0.

20.4.7 SDIO Data Timer Register (SDIO_DTTMR)

Address offset: 0x24

Reset value: 0x00000000

The SDIO_DTTMR register contains the data timeout period, in card bus clock periods.

A counter loads the value from the SDIO_DTTMR register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while DPSM is at either of these states, the timeout status flag is set.

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

rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
Bit 31:0	TIMEOUT: Data timeout period Data timeout period expressed in card bus clock periods																							

Note: Before writing to data control register to perform data transfer, the data timer register and the data length register must be written first.

20.4.8 SDIO Data Length Register (SDIO_DTLEN)

Address offset: 0x28

Reset value: 0x00000000

The SDIO_DTLEN register contains the number of data bytes to be transferred. This value is loaded into the data counter when data transfer starts.

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

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (See [SDIO_DTCTRL](#)). Before writing to data control register to perform data transfer, the data timer register and the data length register must be written first.

20.4.9 SDIO Data Control Register (SDIO_DTCTRL)

Address offset: 0x2C

Reset value: 0x00000000

The SDIO_DTCTRL register controls data path state machine (DPSM).

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

Bit 9	RDWTSTOP: Read wait stop 0: Read wait is in progress if RDWTSTART is set. 1: Read wait stops if RDWTSTART is set.
Bit 8	RDWTSRART: Read wait start Setting this bit starts read wait operation.
Bit 7:4	BLKSIZE: Data block size This field defines the data block size in block transfer mode: 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: (Decimal 4) Block length = 2^4 = 16 bytes 0101: (Decimal 5) Block length = 2^5 = 32 bytes 0110: (Decimal 6) 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: DMA enable bit 0: DMA is disabled. 1: DMA is enabled.
Bit 2	TFRMODE: Data transfer mode selection 0: Block data transfer 1: Stream data transfer
Bit 1	TFRDIR: Data transfer direction selection 0: From controller to card 1: From card to controller
Bit 0	TFREN: Data transfer enabled bit Data transfer starts when '1' is written to this bit. Depending on the direction bit, TFRDIR, DPSM moves to the Wait_S or Wait_R state. If the RDWTSTART is set at the beginning of the transfer, DPSM enters read wait state. It is not necessary to clear the enable bit after the end of a data transfer, but the SDIO_DTCTRL must be updated to enable a new data transfer

Note: At least seven HCLK clock periods are needed between two write accesses to this register.

20.4.10 SDIO Data Counter Register (SDIO_DTCNTR)

Address offset: 0x30

Reset value: 0x00000000

When DPSM moves from the idle state to the Wait_R or Wait_S state, the SDIO_DTCNTR register loads the value from the data length register (See [SDIO DTLEN](#)). As data is transferred, the counter decrements the value until it reaches 0. DPSM then moves to the idle state, and the data status end flag, DTCMPL, is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CNT							
				res				r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:25				Reserved. Always read as 0.											

Bit 24:0	CNT: Data count value Reading this bit is returns the number of remaining data bytes to be transferred. Writing to this register has no effect.
----------	---

Note: This register should be read only when the data transfer is completed.

20.4.11 SDIO Status Register (SDIO_STS)

Address offset: 0x34

Reset value: 0x00000000

The SDIO_STS register is a read-only register, and it includes two types of flags:

- Static flags (Bit [23:22], Bit [10:0]): Writing to the SDIO interrupt clear register (See [SDIO_INTCLR](#)) will clear these bits.
- Dynamic flags (Bit [21:11]): These bits change state depending on the state of the underlying logic (For example, BUF full and empty flags are asserted and deasserted according to the data written to the BUF).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								ATAC MPL	SDIOI F	RXBWF	TXBWF	RXBUE	TXBUE	RXBFF	TXBFF
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBHF_H	TXBHF_H	DORX	DOTX	DOCMD	DTBLKCMPL	SBITERR	DTCMPL	CMDCMPL	CMDRSPCMPL	RXERRO	TXERRU	DTTİMEOUT	CMDTİMEOUT	DTFAIL	CMDFAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31:24	Reserved. Always read as 0.
Bit 23	ATACMPL: CE-ATA command completion signal received for CMD61
Bit 22	SDIOIF: SDIO interrupt received
Bit 21	RXBUFF: Data available in receive BUF
Bit 20	TXBUFF: Data available in transmit BUF
Bit 19	RXBUFF_E: Receive BUF empty
Bit 18	RXBUFF_E: Transmit BUF empty When Hardware Flow Control is enabled, TXBUFF_E signal becomes activated when BUF contains 2 words.
Bit 17	RXBUFF_F: Receive BUF full When hardware flow control is enabled, the RXBUFF_F signal becomes activated 2 words before BUF is full.
Bit 16	TXBUFF_F: Transmit BUF full
Bit 15	RXBUFF_H: Receive BUF half full: There are at least 8 words in BUF.
Bit 14	TXBUFF_H: Transmit BUF half empty: At least 8 words can be written into BUF.
Bit 13	DORX: Data receive in progress
Bit 12	DOTX: Data transmit in progress
Bit 11	DOCMD: Command transfer in progress
Bit 10	DTBLKCMPL: Data block sent/received (CRC check passed)
Bit 9	SBITERR: Start bit not detected on all data signals in wide bus mode
Bit 8	DTCMPL: Data end (The data counter SDIO_DTCNTR = 0)
Bit 7	CMDCMPL: Command sent (No response required)

Bit 6	CMDRSPCMPL: Command response received (CRC check passed)														
Bit 5	RXERRO: Received BUF overrun error														
Bit 4	TXERRU: Transmit BUF underrun error														
Bit 3	DTTIMEOUT: Data timeout														
Bit 2	CMDTIMEOUT: Command response timeout The command timeout period has a fixed value of 64 SDIO_CK clock periods.														
Bit 1	DTFAIL: Data block sent/received (CRC check failed)														
Bit 0	CMDFAIL: Command response received (CRC check failed)														

20.4.12 SDIO Clear Interrupt Register (SDIO_INTCLR)

Address offset: 0x38

Reset value: 0x00000000

The SDIO_INTCLR register is a write-only register. Writing ‘1’ to the corresponding bit clears the corresponding bit in the SDIO_STS status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					ATAC MPL	SDIOI F	Reserved					Reserved			
res					rw	rw	res					res			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				DTBL KCML	S BITER R	DTCM PL	CMD CMPL	CMD RSPC MPL	RXER RO	TXER RU	DTTI MEO UT	CMDT IMEO UT	DTFAI L	CMDF AIL	
res				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:24	Reserved. Always read as 0.														
Bit 23	ATACMPL: ATACMPL flag clear bit This bit is set by software to clear the ATACMPL flag.														
Bit 22	SDIOIF: SDIOIF flag clear bit This bit is set by software to clear the SDIOIF flag.														
Bit 21:11	Reserved. Always read as 0.														
Bit 10	DTBLKCMPL: DTBLKCMPL flag clear bit This bit is set by software to clear the DTBLKCMPL flag.														
Bit 9	SBITERR: SBITERR flag clear bit This bit is set by software to clear the SBITERR flag.														
Bit 8	DTCMPL: DTCMPL flag clear bit This bit is set by software to clear the DTCMPL flag.														
Bit 7	CMDCMPL: CMDCMPL flag clear bit This bit is set by software to clear the CMDCMPL flag.														
Bit 6	CMDRSPCMPL: CMDRSPCMPL flag clear bit This bit is set by software to clear the CMDRSPCMPL flag.														
Bit 5	RXERRO: RXERRO flag clear bit This bit is set by software to clear the RXERRO flag.														
Bit 4	TXERRU: TXERRU flag clear bit This bit is set by software to clear the TXERRU flag.														
Bit 3	DTTIMEOUT: DTTIMEOUT flag clear bit This bit is set by software to clear the DTTIMEOUT flag.														

Bit 2	CMDTIMEOUT: CMDTIMEOUT flag clear bit This bit is set by software to clear the CMDTIMEOUT flag.
Bit 1	DTFAIL: DTFAIL flag clear bit This bit is set by software to clear the DTFAIL flag.
Bit 0	CMDFAIL: CMDFAIL flag clear bit This bit is set by software to clear the CMDFAIL flag.

20.4.13 SDIO Interrupt Mask Register (SDIO_INTEN)

Address offset: 0x3C

Reset value: 0x00000000

This interrupt mask register determines which status flags generate an interrupt request by setting the corresponding bit to '1'.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								ATAC MPL	SDIOI F	RXBU F	TXBU F	RXBU F_E	TXBU F_E	RXBU F_F	TXBU F_F
res								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXBU F_H	TXBU F_H	DORX	DOTX	DOC MD	DTBL KCMP L	S BITER R	DTCM PL	CMD CMPL	CMD RSPC MPL	RXER RO	TXER RU	DTTI MEO UT	CMDT IMEO UT	DTFAI L	CMDF AIL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31:24	Reserved. Always read as 0.
Bit 23	ATACMPL: CE-ATA command completion signal received interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated after receiving CE-ATA command completion signal. 0: CE-ATA command completion signal received interrupt is disabled. 1: CE-ATA command completion signal received interrupt is enabled.
Bit 22	SDIOIF: SDIO mode interrupt received interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated after receiving SDIO mode interrupt. 1: SDIO mode interrupt received interrupt is disabled. 0: SDIO mode interrupt received interrupt is enabled.
Bit 21	RXBUF: Data available in Rx BUF interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by the presence of data available in Rx BUF. 0: Data available in Rx BUF interrupt is disabled. 1: Data available in Rx BUF interrupt is enabled.
Bit 20	TXBUF: Data available in Tx BUF interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by the presence of data available in Tx BUF. 0: Data available in Tx BUF interrupt is disabled. 1: Data available in Tx BUF interrupt is enabled.
Bit 19	RXBUF_E: Rx BUF empty interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Rx BUF empty. 0: Rx BUF empty interrupt is disabled. 1: Rx BUF empty interrupt is enabled.
Bit 18	TXBUF_E: Tx BUF empty interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Tx BUF empty. 0: Tx BUF empty interrupt is disabled. 1: Tx BUF empty interrupt is enabled.
Bit 17	RXBUF_F: Rx BUF full interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Rx BUF full. 0: Rx BUF full interrupt is disabled. 1: Rx BUF full interrupt is enabled.

Bit 16	TXBUF_F: Tx BUF full interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Tx BUF full. 0: Tx BUF full interrupt is disabled. 1: Tx BUF full interrupt is enabled.
Bit 15	RXBUF_H: Rx BUF half full interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Rx BUF half full. 0: Rx BUF half full interrupt is disabled. 1: Rx BUF half full interrupt is enabled.
Bit 14	TXBUF_H: Tx BUF half empty interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Tx BUF half empty. 0: Tx BUF half empty interrupt is disabled. 1: Tx BUF half empty interrupt is enabled.
Bit 13	DORX: Data receive acting interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data receive acting. 0: Data receive acting interrupt is disabled. 1: Data receive acting interrupt is enabled.
Bit 12	DOTX: Data transmit acting interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data transmit acting. 0: Data transmit acting interrupt is disabled. 1: Data transmit acting interrupt is enabled.
Bit 11	DOCMD: Command acting interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by command acting. 0: Command acting interrupt is disabled. 1: Command acting interrupt is enabled.
Bit 10	DTBLKCMPL: Data block end interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data block end. 0: Data block end interrupt is disabled. 1: Data block end interrupt is enabled.
Bit 9	SBITERR: Start bit error interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by start bit error. 0: Start bit error interrupt is disabled. 1: Start bit error interrupt is enabled.
Bit 8	DTCMPL: Data end interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data end. 0: Data end interrupt is disabled. 1: Data end interrupt is enabled.
Bit 7	CMDCMPL: Command sent interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by sending command. 0: Command sent interrupt is disabled. 1: Command sent interrupt is enabled.
Bit 6	CMDRSPCMPL: Command response received interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by receiving command response. 0: Command response received interrupt is disabled. 1: Command response received interrupt is enabled.
Bit 5	RXERRO: Rx BUF overrun error interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Rx BUF overrun error. 0: Rx BUF overrun error interrupt is disabled. 1: Rx BUF overrun error interrupt is enabled.
Bit 4	TXERRU: Tx BUF underrun error interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by Tx BUF underrun error. 0: Tx BUF underrun error interrupt is disabled. 1: Tx BUF underrun error interrupt is enabled.

Bit 3	DTTIMEOUT: Data timeout interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data timeout. 0: Data timeout interrupt is disabled. 1: Data timeout interrupt is enabled.
Bit 2	CMDTIMEOUT: Command timeout interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by command timeout. 0: Command timeout interrupt is disabled. 1: Command timeout interrupt is enabled.
Bit 1	DTCFAIL: Data CRC fail interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by data CRC failure. 0: Data CRC fail interrupt is disabled. 1: Data CRC fail interrupt is enabled.
Bit 0	CMDFAIL: Command CRC fail interrupt enable This bit is set/cleared by software to enable/disable the interrupt generated by command CRC failure. 0: Command CRC fail interrupt is disabled. 1: Command CRC fail interrupt is enabled.

20.4.14 SDIO BUF Counter Register (SDIO_BUFCNTR)

Address offset: 0x48

Reset value: 0x00000000

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 data length register (See [SDIO_DTLEN](#)) when the data transfer enable bit, TFREN, is set in the data control register (SDIO_DTCTRL), and DPSM is at the Idle state. If the data length is not word aligned (multiple of 4), the remaining 1 ~ 3 bytes are regarded as a word.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								CNT							
			res					r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
Bit 31:24		Reserved. Always read as 0.													
Bit 23:0		CNT: Number of words to be written to or read from the BUF.													

20.4.15 SDIO Data BUF Register (SDIO_BUF)

Address offset: 0x80

Reset value: 0x00000000

The receive and transmit BUFS are a set of 32-bit wide registers which can be read or written. They contain 32 entries on 32 sequential addresses. This allows CPU to load and store multiple operands to read from/write to the BUF.

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

Bit 31:0	DT: Receive and transmit BUF data The BUF data occupies 32 entries of 32-bit words, from address: (SDIO base + 0x80) to (SDIO base + 0xFC)
----------	---

21 Universal Serial Bus Full-speed Device Interface (USBDEV)

21.1 Introduction

The USB peripheral implements an interface between a full-speed USB 2.0 bus and the APB1 bus. The USB peripheral supports suspend/resume operation, which allows to stop the device clocks for low-power consumption.

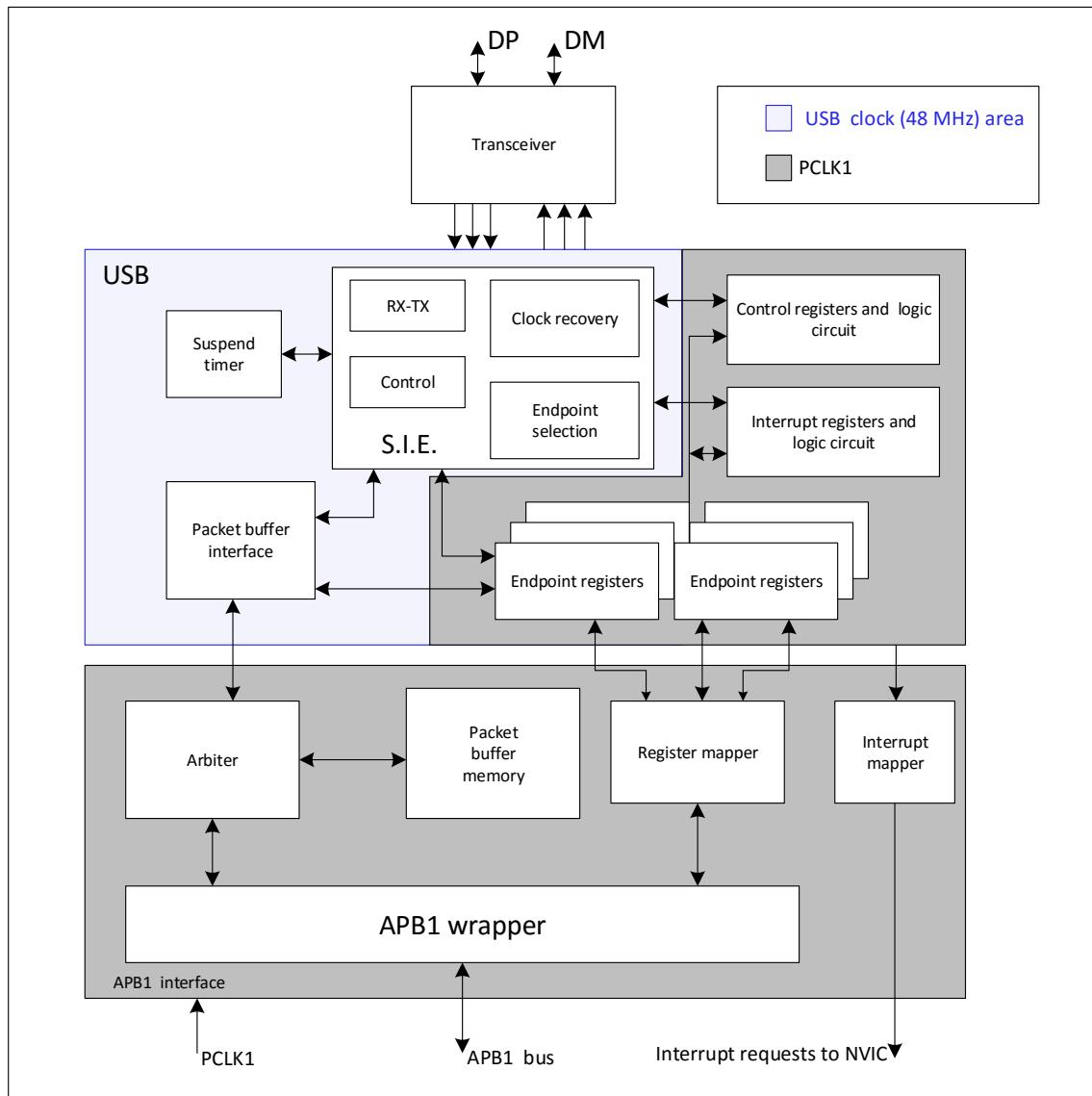
21.2 USB Main Features

- Compliance with USB2.0 full-speed device specification
- Configurable number of USB endpoints from 1 to 8
- CRC (Cyclic redundancy check) generation/checking; non-return-to-zero inverted (NRZI) encoding/decoding and bit-stuffing
- Supports isochronous transfers
- Supports double-buffered bulk/isochronous endpoint
- Supports USB suspend/resume operations
- Frame locked clock pulse generation

Note: USB and CAN share a dedicated SRAM memory for data transmission and reception, so they cannot be used concurrently (The shared SRAM is accessed through CAN and USB exclusively). USB and CAN can be used in the same application but not at the same time. When USB uses the shared SRAM memory, two address mapping spaces are available on APB1, and they can be selected with the USB768B bit in the RCC_MISC register. When USB768B is 0, the size of SRAM is 512 bytes, and the address range is 0x4000 6000 ~ 0x4000 63FF; when USB768B is 1, the size of SRAM is 768 bytes, and the address range is 0x4000 7800 ~ 0x4000 7FFF. When CAN uses this shared SRAM, since CAN accesses it with the internal control bus, the USB768B (control) bit is only valid for USB and has no effect on CAN accesses.

Figure 21-1 shows the block diagram of the USB peripheral.

Figure 21-1 USB Peripheral Block Diagram



21.3 Function Overview

The USB peripheral provides an USB standard compliant communication between the host PC and the function implemented by the microcontroller. Data transfer between the host PC and the microcontroller is completed through a shared dedicated data buffer memory accessed directly by the USB peripheral. The size of this dedicated buffer memory is determined by the number of endpoints used and the maximum packet size of each endpoint; each endpoint can use the maximum of 512-byte/768-byte buffer, which can be used for up to 16 mono-directional or 8 bidirectional endpoints. The USB peripheral interfaces with the PC host, detecting token packets, handling data transmission/reception, and processing handshake packets in compliance with the USB standard. Transaction formatting is performed by the hardware, including CRC generation and checking.

Each endpoint is with a buffer description block, indicating the buffer address used by the endpoint, its size, and the number of bytes should be transmitted. When a token for a valid function/endpoint pair is recognized by the USB peripheral, the related data transfer (if required, and if the endpoint is configured) will take place. Data transfer between the interface and the dedicated buffer is performed by the USB peripheral through an internal 16-bit register. After all data transfer is completed, if needed, the proper handshake packet over the USB is generated or expected according to the direction of the transfer.

At the end of the transaction, an endpoint-specific interrupt is generated by the USB peripheral; by reading status registers and/or using different interrupt response routines, the microcontroller can determine:

- Which endpoint to be served

- The type of transaction in progress if errors occur, such as bit stuffing, format, CRC, protocol, missing ACK, over/underrun, etc.

The USB peripheral provides special double buffer support for isochronous transfers and high throughput bulk transfers, which allows the USB peripheral to always have an available buffer while the microcontroller uses the other one.

The USB peripheral can be placed in low-power mode (SUSPEND mode) by writing in the control register, whenever it is not used. In this case, static power dissipation is avoided, and the USB clock can be slowed down or stopped. The detection of activity at the USB inputs can wake up the device in low-power mode. A special interrupt source can also be connected directly to a wakeup line, enabling the system to immediately restart the normal clock generation and to support direct clock start/stop.

21.3.1 USB Blocks Function Description

The USB peripheral implements all the features related to USB interfacing, and it consists of the following blocks:

- Serial interface engine (SIE): The functions of this block include: Synchronization pattern recognition, bit-stuffing, CRC generation and checking, PID verification/generation, and handshake evaluation. It must interface with the USB transceivers and use the virtual buffers provided by the packet buffer interface for local data storage. This unit also generates signals according to USB peripheral events and to endpoint related events like end of transmission or correct reception of a packet, such as Start of Frame (SOF), USB_Reset, data errors, etc. These signals can generate interrupts.
- Timer: This block generates a start-of-frame locked clock pulse and detects a global suspend (from the host) when no transfer is received for 3 ms.
- Packet buffer interface: This block manages the temporary local memory for transmission and reception. It can assign proper buffers according to requests coming from SIE, and locate them in the memory addresses pointed by the endpoint registers. It increments the address after each byte transfer until the end of packet, keeping track of the number of exchanged bytes and preventing the buffer from overrunning.
- Endpoint-related registers: Each endpoint has an associated register indicating the endpoint type and its current status. For mono-directional/single-buffered endpoints, a single register can be used to implement two distinct endpoints. There are 8 registers used for up to 16 mono-directional/single-buffered endpoints, or up to 7 double-buffered endpoints and their combinations. For example, 4 double-buffered endpoints and 8 single-buffered/mono-directional endpoints can be implemented simultaneously.
- Control registers: These are the registers containing information about the status of the whole USB peripheral and triggering USB events, such as resume and power-down, etc.
- Interrupt registers: These registers contain the interrupt masks and the records of interrupt events. Configuring and accessing these registers can obtain information like interrupt sources and interrupt status, also clearing status flags of pending interrupts.

Note: Endpoint 0 is always used as the control endpoint in single-buffer mode.

The USB peripheral is connected to the APB1 bus through an APB1 interface, which contains the following blocks:

- Packet memory: This is the local memory that contains the packet buffers. It is controlled by the packet buffer interface, which creates the data structure and can be accessed directly by the application software. The size of the packet memory is 512/768 bytes, consisting of 256/384 16-bit words.
- Arbiter: This block handles memory requests coming from the APB1 bus and from the USB interface. It resolves bus conflicts by giving higher priority to APB1 accesses, and always reserves half of the memory bandwidth to complete all the USB transfers. It adopts time-duplex scheme to implement a virtual dual-port SRAM, which allows memory accesses while an USB transaction is in progress. This scheme also allows multi-word APB1 transfers with any lengths.
- Register mapper: This block maps various byte-wide and bit-wide registers of the USB

peripheral in a structured 16-bit wide word set, which can be addressed by APB1.

- APB1 wrapper: This block provides an interface to APB1 for the memory and register. It also maps the whole USB peripheral in the APB1 address space.
- Interrupt mapper: This block is used to map the USB events that can possibly generate interrupts to three different NVIC lines:
 - USB low-priority interrupt (Channel 20): Triggered by all USB events (correct transfer, USB reset, etc.). The firmware has to check the interrupt source first before serving the interrupt.
 - USB high-priority interrupt (Channel 19): Triggered only by a correct transfer event of isochronous and double-buffer bulk transfer, in order to reach the highest transfer rate.
 - USB wakeup interrupt (Channel 42): Triggered by wakeup events from the USB suspend mode.

21.3.2 Generic USB Device Programming

This section describes the main tasks done by the application to implement the USB function. In addition to general actions taken in USB events, this section also emphasizes the special cases of double-buffered endpoints and isochronous transfers. These relevant actions are always initialized by the USB peripheral and triggered by one of the USB events described below.

21.3.2.1 System Reset and Power-on Reset

Upon system reset or power-on reset, the application should first provide all required clock signals to the USB peripheral and then de-assert its reset signal, which allows the USB peripheral to access the registers. The complete initialization sequence is described as follows:

First, the application should activate the macrocell register clock and de-assert macrocell-specific reset signal by configuring the relevant control bits of the device clock management logic.

After that, the analog part associated with the USB transceiver must be switched on by configuring the PDWN bit in the CTRL register; this step requires special attention. This bit can switch on the internal voltage references that supply the port transceiver. This circuit has a defined startup time ($t_{STARTUP}$ specified in the datasheet) during which the behavior of the USB transceiver is undefined. Therefore, it is necessary to wait for a period of time before clearing the reset signal on the USB part (by clearing the FRST bit in the CTRL register). Clearing the INTSTS register will remove any spurious pending interrupt before any other macrocell operation is enabled.

Finally, the application should provide 48 MHz clock as defined in the standard for the USB peripheral by configuring the corresponding control bits of the device clock management logic.

During system reset, the microcontroller must initialize all required registers and the packet buffer description table, to make the USB peripheral able to properly generate interrupts and complete data transfers. All registers not specific to any endpoint must be initialized according to the needs of application (e.g. selection of interrupt enable or address of packet buffers, etc.). Then, the process is the same as the one for the USB reset case (Please refer to the next section).

USB reset (RSTF interrupt)

When USB reset occurs, the USB peripheral will be at system reset state as described in the previous sections: communication is disabled in all endpoint registers (the USB peripheral will not respond to any packet). After USB reset, the USB function is enabled, at the same time enabling the default control endpoint (endpoint 0) of which the address is 0. This is accomplished by setting the EN bit in the USB_DEVADR register, the EPT0 register, and its related packet buffers accordingly. During USB enumeration process, the host assigns a unique address to this device, which must be written in the ADR [6:0] bits of the USB_DEVADR register, and configures any other necessary endpoints.

When a RESET interrupt is received, the application must enable the default endpoint 0 within 10 ms after the interrupt is generated.

Structure and usage of packet buffers

Each bidirectional endpoint may receive or transmit data from/to the host. The received data is stored in a dedicated memory buffer reserved for that endpoint, while the other memory buffer contains the data to be transmitted by the endpoint. Accesses to this memory are performed by the packet buffer interface block, which delivers a memory access request and waits for its acknowledgement. To avoid the access conflict between the microcontroller and the USB peripheral, the packet buffer interface block adopts an arbitration logic, using half APB1 cycle for microcontroller access and the other half for the USB

peripheral access. In this way, both the microcontroller and USB can operate as if the packet memory is a dual-port SRAM; no conflict will occur even when the microcontroller is performing accesses continuously.

The USB peripheral logic uses a dedicated clock. The frequency of this dedicated clock is fixed according to the requirements of the USB standard at 48 MHz. APB1 clock frequency can be higher or lower than this frequency.

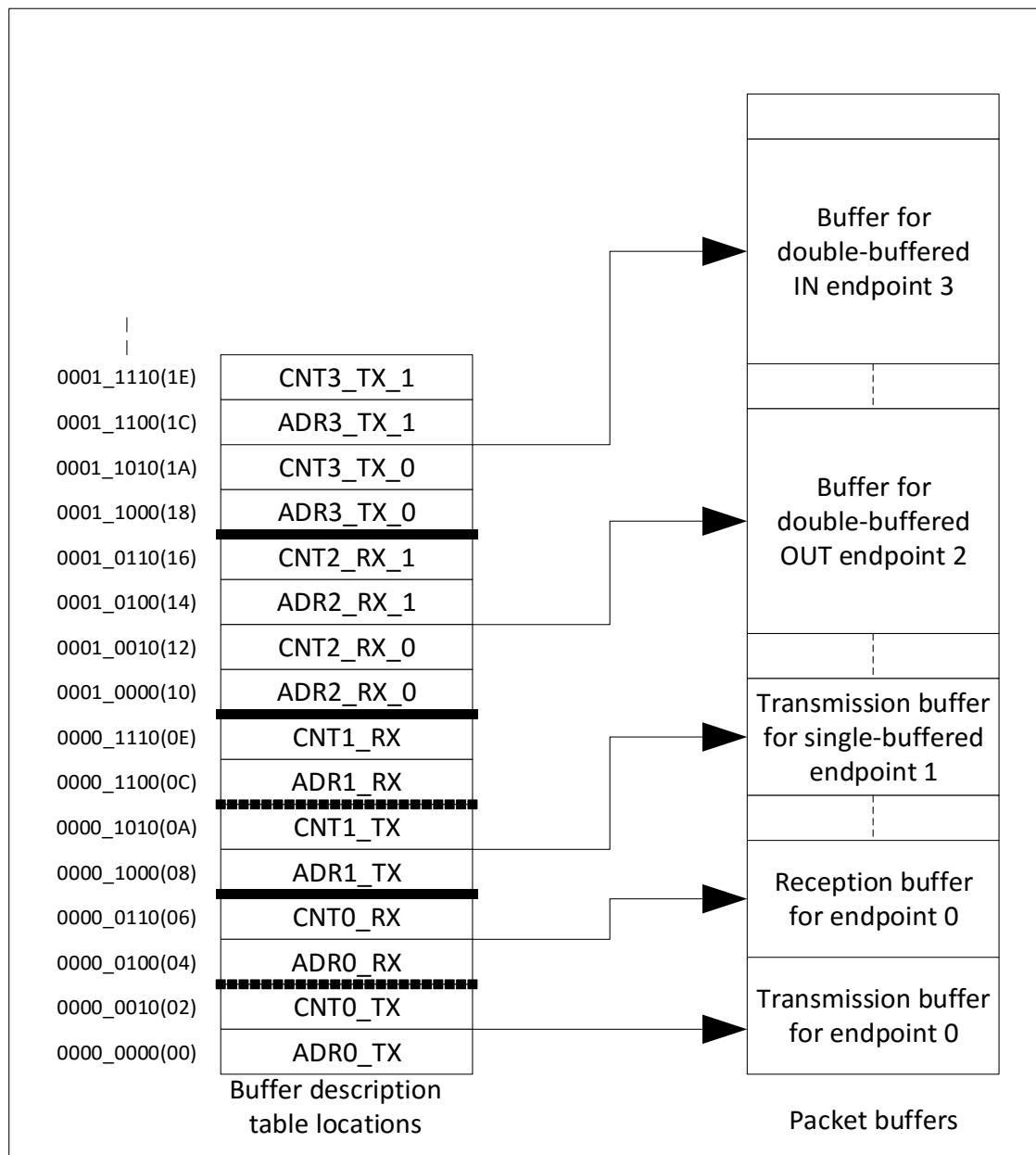
Note: *To fulfill USB data rate and packet memory interface requirements, the APB1 clock frequency must be greater than 8 MHz to avoid data overrun/underrun problems.*

Each endpoint is associated with two packet buffers (Usually, one for transmission and the other for reception). Buffers can be placed anywhere inside the packet memory because their locations and sizes are specified in a buffer description table, which is also located in the packet memory at the address indicated by the register.

Each table entry is associated to an endpoint register, which is composed of 4 16-bit words so that the start address of the table must always be aligned to an 8-byte boundary (the least significant 3 bits of the register are always '000'). If an endpoint is unidirectional, neither isochronous nor double-buffered bulk, only one packet buffer is required (the one related to the supported transfer direction).

Other table locations related to unsupported transfer directions or unused endpoints are available for other purposes. Isochronous and double-buffered bulk endpoints have special handling on packet buffers. The relationship between buffer description table entries and packet buffer areas is detailed in Figure 21-2.

Figure 21-2 Packet Buffer Areas and Associated Buffer Description Table Locations



Each packet buffer is used from the bottom either during reception or transmission. The USB peripheral will never change the contents of memory locations adjacent to the allocated memory buffers; if a packet bigger than the allocated buffer length is received, the buffer only receives data with its own size and discards the others; that is, a buffer overrun error will occur.

Endpoint initialization

The first step to initialize an endpoint is to write appropriate values to the ADRn_TX/ADRn_RX registers, so that the USB peripheral can find the data to be transmitted or the buffer already available to receive data. The EPT_TYPE bits in the USB_EPTx register must be set to determine the endpoint type, and the EPT_SUBTYPE bit to determine their special required features. On the transmit side, the endpoint must be enabled through the STS_TX bits in the USB_EPTx register, and CNTn_TX must be configured to define the transmission length. For reception, the STS_RX bits must be set to enable the endpoint, and the BLKSIZE and the NUM_BLK bits must be written with the allocated buffer size, to detect buffer overrun errors. For unidirectional, non-isochronous, and non-double-buffered bulk endpoints, only one register related to the supported direction should be configured. Once the endpoint is enabled, the USB_EPTx register and locations of ADRn_TX/ADRn_RX, CNTn_TX/CNTn_RX can no longer be modified by the application software, as the hardware can change their values on the fly. When the data transfer is completed, a CTFRF interrupt will be generated, and the above-mentioned registers can be accessed again to re-enable a new operation.

IN packets (for data transmission)

When receiving an IN token packet, if the received address matches the configured endpoint address, the USB peripheral will access the corresponding ADRn_TX and CNTn_TX registers based on the entry of buffer description table; then, the contents of these registers are stored in its internal 16-bit registers, ADDR and COUNT (not accessible by software). At this time, the USB peripheral starts sending a DATA0 or DATA1 packet according to the DTOG_TX bit and accesses the buffer. After IN packet is completed, the first byte read from buffer memory is loaded into the output shift register, starting to be transmitted. After the last data byte is transmitted, the computed CRC is sent. If the addressed endpoint is not valid, a NAK or STALL handshake packet is sent instead of the data packet, according to the STS_TX bits in the USB_EPTx register.

The ADDR internal register is used as a pointer to the current location of buffer memory, while the COUNT register is used to count the number of remaining bytes to be transmitted. Each word read from the packet buffer memory is transmitted over the USB bus, starting from the least significant byte. Transmission buffer memory is read from the address pointed by ADRn_TX, and the length is CNTn_TX/2 words. If a transmitted packet is composed of an odd number of bytes, only the lower 8 bits of the last word accessed will be used.

On receiving the ACK by the host, the USB_EPTx register is updated in the following way: The DTOG_TX bit is toggled, the endpoint is made invalid by setting STS_TX = 10, and the CTFR_TX bit is set. The application must first identify the endpoint, which generates interrupts, by examining the EPT_ID and DIR bits in the USB_INTSTS register. Service of the CTFR_TX event first clears the interrupt bit, then prepares another buffer to be sent, updates CNTn_TX with the number of byte to be transmitted during the next transfer, and finally sets STS_TX to '11' (endpoint valid) to re-enable transmissions. When the STS_TX bits are '10' (endpoint NAK), any IN request addressed to that endpoint is NAKed, and the USB host will re-send the IN request until the endpoint confirms the request to be valid. It is required to execute the sequence of operations in the above mentioned order to avoid losing the next IN transaction following the previous CTFR interrupt request.

OUT and SETUP packets (data reception)

These two tokens are handled by the USB peripheral more or less in the same way; the differences in the handling of SETUP packets are detailed in the following section about control transfers. When receiving an OUT/SETUP packet, if the address matches a valid endpoint, the USB peripheral accesses the buffer descriptor table, finds the ADRn_RX and CNTn_RX registers related to the addressed endpoint, and stores the content of the ADRn_RX register in its internal register, ADDR. At the same time, COUNT is reset; the values of BLKSIZE and NUM_BLK, read from CNTn_RX, are used to initialize an internal 16 bit register, BUF_COUNT, which is used to check the buffer overrun condition (All these internal registers are not accessible by software). Data bytes subsequently received by the USB peripheral are packed in words (The first byte received is stored as the least significant byte) and then transferred to the packet buffer pointed by ADDR. At the same time, the value of BUF_COUNT is decremented automatically, while COUNT is incremented at each byte transfer. When the end signal of data packet is detected, the correctness of the received CRC is tested. If no errors occur during the reception, an ACK handshake packet is sent back to the host. In case of wrong CRC or other kinds of errors (bit-stuff violations, frame errors, etc.), data bytes are still copied into the packet memory buffer, at least until the error detection point, but the ACK packet is not sent, and the ERRF bit in USB_INTSTS register is set. However, there is usually no software action required in this case; the USB peripheral recovers from reception errors automatically and remains ready for the next transaction to come. If the addressed endpoint is not ready, a NAK or STALL handshake packet is sent, according to the STS_RX bits in the USB_EPTn register; no data is written in the reception memory buffers.

The location of reception memory buffer starts from the address contained in ADRn_RX, and the length is determined by the data packet length (i.e. valid data length + 2), including CRC, but cannot exceed the buffer length defined by BLKSIZE and NUM_BLK. If the length of the data packet payload is greater than the allocated buffer, data out of the range will not be written into the buffer, and the USB peripheral detects a buffer overrun condition. In this case, a STALL handshake is sent to notify this failure to the host, and no interrupt is generated.

When the transaction is completed correctly, the USB peripheral sends the ACK handshake packet, and the value of the internal COUNT register is copied into the corresponding CNTn_RX register, leaving the BLKSIZE and NUM_BLK fields unaffected and not required to be re-written. The USB_EPTn register is updated in the following way: The DTOG_RX bit is toggled, the endpoint is made invalid by setting STS_RX = 10 (NAK), and the CTFR_RX bit is set (If CTFRF interrupt is enabled, an interrupt will be

triggered). If errors or buffer overrun condition occur during the transaction, none of the previously listed actions take place. When a CTFR interrupt occurs, the application software must first identify its endpoint by examining the EPT_ID and DIR bits in the USB_INTSTS register. The CTFR_RX interrupt event is serviced by first determining the transaction type (based on the SETUP bit in the USB_EPTn register); the application must clear the interrupt flag bit and obtain the number of received bytes by reading the CNTn_RX register pointed by the buffer description table entry. After the received data is processed, the application should set the STS_RX bits to '11' in USB_EPT, enabling further transactions.

When the STS_RX bits are '10' (NAK), any OUT request addressed to that endpoint is NAKed; the USB host will retry the transaction until it succeeds. It is necessary to execute the sequence of operations in the above mentioned order to avoid losing the second OUT transaction following the previous CTFR interrupt.

Control transfer

Control transfer consists of 3 phases: First, a SETUP transaction, followed by zero or more data stages, and finally a status stage, consisting of a data packet in the opposite direction. SETUP transactions are handled by control endpoints only and are very similar to the OUT process (data reception). To enable SETUP transfer, the DTOG_TX and DTOG_RX bits are initialized to '1' and '0', respectively; in addition, STS_TX and STS_RX are set to '10' (NAK) to allow software to decide if subsequent transactions are IN or OUT depending on the SETUP contents. A control endpoint must check the SETUP bit in the USB_EPTn register at each CTFR_RX interrupt event to distinguish normal OUT transactions from the SETUP transactions. A USB device can determine the number and direction of transfer at data stage according to the corresponding data in SETUP transaction, and can send STALL transaction in the case of errors to reject data transfer. Therefore, at all data stages, the unused direction should be set to STALL; its opposite direction is still set as NAK during the last data transaction at data stage, so that if the host reverses the transfer direction too soon (entering status phase), it remains the state of waiting for control transfer completion. After the control operation is completed successfully, the software will change NAK to VALID; otherwise, it will be changed to STALL if errors occur. At the same time, if the status stage will be an OUT, the STATUS_OUT bit (EPT_SUBTYPE in the USB_EPTn register) should be set, so that an error is generated if a status transaction is performed with nonzero data. When the status transaction is finished, the application clears the STATUS_OUT bit and sets STS_RX to VALID, indicating "ready to accept a new command", and STS_TX to NAK, indicating that no data transfer request is accepted before the next SETUP transaction is completed.

The USB specification states that a SETUP packet cannot be answered with a handshake different from ACK; if SETUP packet transfer fails, a new one will be started. Therefore, it is not allowed to answer a SETUP token received from the host with a NAK or STALL packet.

When the STS_RX bits are set to '01' (STALL) or '10' (NAK), and a SETUP token is received, the USB peripheral accepts the data, performing the required data transfers and sending back an ACK handshake. If that endpoint has a previously issued CTFR_RX request not yet acknowledged by the application (i.e. the CTFR_RX bit is still set), the USB discards the SETUP transaction and does not answer with any handshake packet, which simulating a reception error and forcing the host to send the SETUP token again. This is done to avoid losing the notification of a SETUP transaction addressed to the same endpoint following the CTFR_RX interrupt.

21.3.2.2 Double-buffered Endpoints

The USB standard not only defines the different endpoint types for different traffic models, but also describes the typical requirements of these data transfer operations. Among these operations, the bulk endpoint type is the most suited model for large portions of data transfer between the host PC and the USB peripheral; this is because the host can fill all the available bandwidth in the frame, maximizing the actual transfer rate. However, if the USB function is still busy with the previous transaction when the next one arrives, it will answer with a NAK handshake, and the host PC will issue the same transaction again until the USB function is ready to handle it; this reduces the actual transfer rate due to the bandwidth occupied by re-transmissions. For this reason, a dedicated feature called "double-buffering" with bulk endpoints is introduced to increase transfer rate.

When "double-buffering" is activated, unidirectional data transfer will use both the transmission and reception data buffer of the endpoint. The data toggle bit is used to select which buffer is to be used by the USB peripheral to perform the required data transfers. While the USB peripheral is accessing one of the buffer, the application can perform operations on the other one. For example, during an OUT transaction directed to a double-buffered bulk endpoint, the USB peripheral will store data coming from the PC host into a buffer, and the other one is available for the application usage (The same would

happen in an IN transaction). Since the swapped buffer management requires all four buffer description table locations to indicate the address pointers and the buffer length on each direction, the USB_EPTn registers used to implement double-buffered bulk endpoints are forced to be unidirectional. Therefore, only the STS_RX bit (double-buffered bulk endpoint for reception) or the STS_TX bit (double-buffered bulk endpoint for transmission) should be set. If a double-buffered bulk endpoint is required, then two USB_EPTn registers are needed.

To exploit the double-buffering feature and reach the highest possible transfer rate, its endpoint flow control structure is a little different from other endpoints. It sets endpoint to NAK only when a buffer conflict occurs, instead of each time at the end of a successful transaction.

The DTOG bit is used to indicate the memory buffer currently being used. The receive direction of the double-buffer bulk endpoint is indicated by DTOG_RX (the bit 14 of USB_EPTn register), while the transmission direction is indicated by DTOG_TX (the bit 6 of USB_EPTn register). At the same time, the USB peripheral should know which packet buffer is currently in use by the application, to avoid any conflict. Since in the USB_EPTn register, there are two DTOG bits, and the USB peripheral only uses one to indicate the buffer used by hardware, so the application can use the other bit to show which buffer is currently being used; this new flag is called the SW_BUF bit. Table 21-1 lists the correspondence between the DTOG and SW_BUF bits in the USB_EPTn register when transmission and reception are implemented with double-buffered bulk endpoints.

Table 21-1 Double-buffering Bulk Endpoint - Buffer Flag Definition

Buffer Flag Bit	As Transmission Endpoint		As Reception Endpoint
DTOG	DTOG_TX (The bit 6 of the USB_EPTn register)		DTOG_RX (The bit 14 of the USB_EPTn register)
SW_BUF	The bit 14 of the USB_EPTn register		The bit 6 of the USB_EPTn register

The memory buffer currently being used by the USB peripheral is defined by the DTOG buffer flag, while the buffer currently in use by application software is identified by the SW_BUF buffer flag. The indication of these two bits is the same and listed in Table 21-2.

Table 21-2 Double-buffered Bulk Endpoint Flags

Endpoint Type	DTOG	SW_BUF	Packet Buffer Used by the USB Peripheral	Packet Buffer Used by the Application Software
IN	0	1	ADRn_TX_0/CNTn_TX_0	ADRn_TX_1/CNTn_TX_1
	1	0	ADRn_TX_1/CNTn_TX_1	ADRn_TX_0/CNTn_TX_0
	0	0	None ^(Note)	ADRn_TX_0/CNTn_TX_0
	1	1	None ^(Note)	ADRn_TX_0/CNTn_TX_0
OUT	0	1	ADRn_RX_0/CNTn_RX_0	ADRn_RX_1/CNTn_RX_1
	1	0	ADRn_RX_1/CNTn_RX_1	ADRn_RX_0/CNTn_RX_0
	0	0	None ^(Note)	ADRn_RX_0/CNTn_RX_0
	1	1	None ^(Note)	ADRn_RX_1/CNTn_RX_1

Note: Endpoint is at NAK state.

Double-buffering feature for a bulk endpoint is activated by:

- Setting the EPT_TYPE bit field as '00' in the USB_EPTn register, to define the endpoint as a bulk
- Setting the EPT_SUBTYPE bit as '1' in the SB_EPTn register, to define the endpoint as double-buffered

The application software initializes the DTOG and SW_BUF bits according to the buffer used when transfer starts; this should take the special toggle-only property of the two bits into consideration. After setting DBL_BUF, each time when a transfer is completed, the USB peripheral controls the operation according to the flow of double-buffered bulk endpoints, which is used for all other transactions addressed to this endpoint until DBL_BUF becomes invalid. At the end of each transaction, the CTFR_RX or CTFR_TX bit is set, depending on the enabled direction. At the same time, the DTOG bit is set by hardware, implementing buffer swap mechanism in a complete software-independent way. After the DBL_BUF bit is set, after each transfer completion, the value of STAT bit at double-buffered bulk endpoint is not affected like other common transactions, and it remains '11' (Valid). However, as the token packet of a new transaction is received, the actual endpoint status will be masked as '10' (NAK) when a buffer conflict between the USB peripheral and the application software is detected (i.e. DTOG

and SW_BUF have the same value). The application software responds to the CTFR event by first clearing the interrupt flag and then handling the completed transaction. After the application accesses the buffer, the software toggles the SW_BUF bit, to notify the USB peripheral about the availability of that buffer. In this way, the number of NAKed transactions is determined only by the application elaboration time of a transaction data: If the elaboration time is shorter than the time required to complete a transaction on the USB bus, no re-transmissions will take place, and the actual transfer rate will be limited only by the USB host. The application software can always override the special flow control implemented for double-buffered bulk endpoints, writing an explicit status different from '11' (Valid) into the STS bit pair of the related USB_EPTn register. In this case, the USB peripheral will always use the programmed endpoint status, regardless of the actual buffer usage condition.

21.3.2.3 Isochronous Transfers

The USB standard supports full speed peripherals requiring a fixed and accurate data transfer frequency: Isochronous transfer. Isochronous transfer is commonly used for audio samples, compressed video streams, and in general any sort of data having strict requirements for the delivered frequency. When an endpoint is defined to be "isochronous" at the enumeration phase, the USB host allocates each frame with fixed bandwidth and ensures that each frame transfers exactly one IN or OUT packet (depending on endpoint direction). To fulfill the bandwidth requirements, there is no retransmission of failed transactions for isochronous traffic; this leads to the fact that an isochronous transaction does not have a handshake phase, and no ACK packet is sent after the data packet transactions. For the same reason, isochronous transfers do not support data toggle sequencing and always uses DATA0 PID (packet ID) to start any data packet.

The isochronous behavior for an endpoint is selected by setting the EPT_TYPE bits at '10' in the USB_EPTn register. Since there is no handshake phase, the only legal values for the STS_RX/STS_TX bit pairs are '00' (Disabled) and '11' (Valid) according to the USB standard. Isochronous endpoints implement double-buffering to ease the software application development, using both transmission and reception buffer areas to ensure that application can access one buffer while the USB peripheral fills the other.

The memory buffer used by the USB peripheral is defined by the DTOG bit related to the endpoint direction (DTOG_RX for "reception" isochronous endpoints, DTOG_TX for "transmission" isochronous endpoints, both in the related register) according to Table 21-3.

Table 21-3 Isochronous Memory Buffers Flags

Endpoint Type	DTOG	Packet Buffer Used by USB Peripheral	Packet Buffer Used by Application Software
IN	0	ADRn_TX_0/CNTn_TX_0	ADRn_TX_1/CNTn_TX_1
	1	ADRn_TX_1/CNTn_TX_1	ADRn_TX_0/CNTn_TX_0
OUT	0	ADRn_RX_0/CNTn_RX_0	ADRn_RX_1/CNTn_RX_1
	1	ADRn_RX_1/CNTn_RX_1	ADRn_RX_0/CNTn_RX_0

As it happens with double-buffered bulk endpoints, the USB_EPTn registers only implement unidirectional transfers of isochronous endpoints. If it is required to have isochronous endpoints enabled for two directions, then two USB_EPTn registers must be used. The application software is responsible for the DTOG bit initialization according to the first buffer used; this has to be done considering the special toggle-only property that these two bits have. At the end of each transaction, the CTFR_RX or CTFR_TX bit of the addressed endpoint USB_EPTn register is set. At the same time, the related DTOG bit is hardware toggled, making buffer swapping completely software independent. The STS_RX or STS_TX bits are not affected after transaction completion; due to the lack of handshake phase, no flow control is needed, and the endpoint always remains '11' (Valid). During isochronous transfers, CRC errors or buffer-overrun conditions are still considered as correct transactions and can trigger a CTFR_RX interrupt event. However, CRC errors will set the ERRF bit in the USB_INTSTS register to notify the software of the possible data corruption.

21.3.2.4 Suspend/Resume Events

The USB standard defines a special peripheral state, called SUSPEND, in which the average current drawn from the USB bus is under 500 μ A. This requirement is very important for bus-powered devices, while self-powered devices are not required to comply with this strict power consumption constraint. The

USB host enters the suspend mode if it does not send any traffic on the USB bus for more than 3 ms; normally, a SOF packet is sent every ms by the USB peripheral, so the lack of 3 consecutive SOF packets is identified as a suspend request from the host, and USB will set the SUSPF bit to '1' in the USB_INTSTS register, generating an interrupt. Once the device is suspended, its normal operation can be restored by a "RESUME" sequence, which can be started from the host or directly from the peripheral itself, but it is always terminated by the host. The suspended USB peripheral must be able to detect a RESET sequence and react to this event as a normal USB reset operation.

The actual procedure used to suspend the USB peripheral is device-dependent, since different actions may be required to reduce the total consumption for different devices. A brief description of a typical suspend procedure is provided below, focusing on how the application software responds to the SUSPF notification of the USB peripheral:

1. Set the FSUSP bit in the USB_CTRL register to '1', activating the suspend mode within the USB peripheral. As soon as the suspend mode is activated, the check on SOF reception is disabled to avoid any further SUSPF interrupts issued while USB is suspended.
2. Remove or reduce any static power consumption in other blocks aside from the USB peripheral.
3. Set the LPWR bit in the USB_CTRL register to '1' to remove static power consumption in the analog USB transceivers but keep them able to detect resume signal.
4. Optionally turn off external oscillator and PLL of the device to stop any activity inside the device. When an USB event occurs while the device is in suspend mode, the "RESUME" procedure must be invoked to restore nominal clocks and regain normal USB behavior. Particular attention must be given to ensure that this process does not take more than 10 ms if the wakening event is an USB reset sequence (Please refer to "Universal Serial Bus Specification" for more details). When the USB peripheral is suspended, the start of a resume or reset sequence clears the LPWR bit in the USB_CTRL register. Even if this event can trigger a WKUPF interrupt immediately, the interrupt response routine must be carefully evaluated because of the long latency required by system clock restart; to reduce the latency before re-activating the nominal clock, it is suggested that users put the resume procedure right after the end of the suspend code, so its code is immediately executed as soon as the system clock restarts.

The following is the resume procedure: Enable external oscillator and PLL of the device (optional)

1. Clear the FSUSP bit in the USB_USB_CTRL register.
2. The DPSTS and DMSTS bits in the USB_FRNUM register can be used to identify the resume triggering event, as shown in [Table 21-4](#), which also lists the intended software action in all the cases. If required, the end of resume or reset sequence can be detected by monitoring the time when the two bits reach '10' (which indicates the idle bus state); moreover, at the end of a reset sequence, the RSTF bit in USB_INTSTS register is set to '1', issuing an interrupt if enabled. This interrupt should be handled as a usual reset.

Table 21-4 Resume Event Detection

[DPSTS, DMSTS] Status	Wakeup Event	Required Software Resume Action
00	Reset	None
10	None (Noise on the bus)	Go back to suspend state
01	Resume	None
11	Undefined (Noise on the bus)	Go back to suspend state

A device may require to exit from the suspend mode as an answer to particular events not directly related to the USB protocol (e.g. a mouse movement wakes up the whole system). In this case, the resume sequence can be started by setting the RESUME bit in the USB_CTRL register to '1' and resetting it to 0 after an interval between 1 ms and 15 ms (this interval can be timed by ESOF interrupts, which occur every 1 ms period when the system clock is running at normal frequency). Once the RESUME bit is cleared, the resume sequence will be completed by the host PC, and its completion can be monitored by the DPSTS and DMSTS bits in the USB_FRNUM register.

Note: *The RESUME bit must be configured only after the USB peripheral is put in suspend mode (setting the FSUSP bit in the USB_CTRL register to '1').*

21.4 USB Registers

The USB peripheral registers include the following three groups:

- Common registers: Interrupt registers and control registers
- Endpoint registers: Endpoint configuration registers and status registers
- Buffer description table registers: Registers that define the addresses to store packet memory

The base addresses of buffer description table registers are specified by the USB_BUFTBL register. All other registers are expressed with respect to the base address of the USB peripheral: 0x4000 5C00. Since APB1 addresses on a word (32-bit) basis, all register addresses are aligned to 32-bit word boundaries even though they are 16-bit wide. The same address alignment is used to access packet buffer memory locations, which are located starting from 0x4000 6000/0x4000 7800.

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

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	CTFR_RX	15
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USB_EPT0	Reserved																0	CTFR_RX	15
		0x0000																0	DTOG_RX	14
0x04	USB_EPT1	Reserved																0	STS_RX[1:0]	13
		0x0000																0	SETUP	12
0x08	USB_EPT2	Reserved																0	EPT_TYPE[1:0]	10
		0x0000																0	EPT_SUBTYPE	8
0x0C	USB_EPT3	Reserved																0	CTFR_RX	7
		0x0000																0	DTOG_RX	6
0x10	USB_EPT4	Reserved																0	STS_TX[1:0]	5
		0x0000																0	0	4

0x14	USB_EPT5	Reserved										CTFR_RX	0	CTFR_RX	0
												DTOG_RX	0	DTOG_RX	0
0x18	USB_EPT6	Reserved										STS_RX[1:0]	0	STS_RX[1:0]	0
												SETUP	0	SETUP	0
0x1C	USB_EPT7	Reserved										EPT_TYPE[1:0]	0	EPT_TYPE[1:0]	0
												EPT_SUBTYPE	0	EPT_SUBTYPE	0
0x40	USB_CTRL	Reserved										CTFR_RX	0	CTFR_RX	0
												DTOG_RX	0	DTOG_RX	0
0x44	USB_INTSTS	Reserved										STS_RX[1:0]	0	STS_RX[1:0]	0
												SETUP	0	SETUP	0
0x48	USB_FRNUM	Reserved										RESUME	0	RESUME	0
												FSUSP	0	FSUSP	0
0x4C	USB_DEVADR	Reserved										LPWR	0	LPWR	0
												PDWN	0	PDWN	0
0x50	USB_BUFTBL	Reserved										FRST	0	FRST	0
												ADR[6:0]	0	ADR[6:0]	0

21.4.1 Common Registers

These registers define the USB peripheral's operating mode, interrupt handling, device address, and the access to the current frame number.

21.4.1.1 USB Control Register (USB_CTRL)

Address offset: 0x40

Reset value: 0x0003

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTFR_IEN	PMOVR_IEN	ERR_IEN	WKUP_IEN	SUSP_IEN	RST_IEN	SOF_IEN	ESOF_IEN	Reserved	RESUME	FSU_SP	LP_WR	PD_WN	FR_ST		
rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw	rw	rw	rw	rw

Bit 15	CTFR_IEN: Correct transfer (CTFRF) interrupt mask 0: Correct transfer (CTFRF) interrupt is disabled. 1: Correct transfer (CTFRF) interrupt is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 14	PMOVR_IEN: Packet memory area over/underrun interrupt mask 0: PMOVR interrupt is disabled. 1: PMOVR interrupt is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 13	ERR_IEN: Error interrupt mask 0: Error interrupt is disabled. 1: Error interrupt is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 12	WKUP_IEN: Wakeup interrupt mask 0: Wakeup interrupt is disabled. 1: Wakeup interrupt is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 11	SUSP_IEN: Suspend mode interrupt mask 0: Suspend mode (SUSPF) interrupt is disabled. 1: Suspend mode (SUSPF) interrupt is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 10	RST_IEN: USB reset interrupt mask 0: USB RSTF interrupts is disabled. 1: USB RSTF interrupts is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 9	SOF_IEN: Start of frame interrupt mask 0: SOFF interrupts is disabled. 1: SOFF interrupts is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 8	ESOF_IEN: Expected start of frame interrupt mask 0: ESOF interrupts is disabled. 1: ESOF interrupts is enabled; an interrupt is generated if the corresponding bit in the interrupt register is set.
Bit 4	RESUME: Resume request Setting this bit will send a resume signal to the host. According to the USB specification, if this bit remains active for 1 ms to 15 ms, the host will perform the resume sequence to the USB peripheral.
Bit 3	FSUSP: Force suspend When no traffic is received by the USB peripheral for 3 ms, the SUSPF interrupt is issued, and then the software must set this bit. 0: No effect 1: Enter suspend mode. Clocks and static power dissipation in the analog transceiver are left unaffected. If low-power consumption is required (bus-powered devices), the application software should set the FSUSP bit and then the LPWR bit.

Bit 2	LPWR: Low-power mode This mode is used to lower the power consumption at USB suspend state. In this mode, all static power dissipation is avoided, except the one required to supply the external pull-up resistor. The system clocks may stop or reduce their frequencies to meet the power consumption requirements. The USB activity (wakeup event) during the suspend mode will reset this bit (it can also be reset by software). 0: No low-power mode 1: Low-power mode
Bit 1	PDWN: Power down This bit is used to completely disable the USB peripheral. When this bit is set, the USB peripheral cannot be used. 0: Exit power-down mode 1: Enter power-down mode
Bit 0	FRST: Force USB Reset 0: Clear the USB reset signal 1: Force a reset to the USB peripheral, similar to the reset signal on the USB bus. The USB peripheral is held at reset state until software clears this bit. A reset interrupt is generated if the USB reset interrupt is enabled.

21.4.1.2 USB Interrupt Status Register (USB_INTSTS)

Address offset: 0x44

Reset value: 0x0400

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTF RF	PMOV ERF	ERRF	WK UPF	SU SPF	RSTF	SOFF	ESOF	Reserved	DIR	EPT_ID[3:0]					
r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	res	r	r				

This register contains the status of all the interrupt sources, allowing the application to determine the event that generates an interrupt request.

Each of the upper 8 bits in this register represents an interrupt source. These bits are set by the hardware when the corresponding event occurs; if the corresponding bit in the USB_CTRL register is set, an interrupt request is generated. The interrupt routine should examine each bit, and must clear the corresponding status bits after performing all necessary operation; otherwise, the interrupt line will be kept high, causing the same interrupt to be triggered again. If several bits are set simultaneously, only a single interrupt will be generated. Interrupts can be handled in a different way by the application to reduce interrupt response latency. The CTFRF bit is set by the hardware as soon as an endpoint successfully completes a transaction, generating an interrupt request if the corresponding bit in USB_CTRL is set. An endpoint dedicated interrupt flag is independent of the CTFR_IEN bit in the USB_CTRL register. Both the interrupt flag bits remain active until software clears the pending bit in the corresponding USB_EPTn register (the CTFRF bit is read-only). The USB peripheral has two interrupt request sources:

High priority USB IRQ: Used for interrupt requests of endpoints with higher priority (isochronous and double-buffered bulk endpoints), and the interrupt cannot be masked.

Low priority USB IRQ: Used for other interrupt events, such as unmaskable interrupts with lower priority or maskable interrupts identified by the upper 8 bits in the USB_INTSTS register. For endpoint-related interrupts, the software can use the DIR register and the EP_ID read-only bits to identify which endpoint makes the last interrupt request and to call the corresponding interrupt service routine.

When handling simultaneous pending interrupt events, users can check the order of each bit in the USB_INTSTS register to determine the priority of these events in the interrupt service routine. After handling the interrupt of the corresponding bits, the interrupt flags are cleared. At the end of the service routine, another interrupt will be requested to handle the remaining interrupts.

To avoid spurious clearing of some bits, it is recommended that users use load instruction, and all bits, that must not be altered, are written with '1', while all bits, that need to be cleared, are written with '0'. Read-modify-write cycles should be avoided for this register because hardware may need to set some bits between the read and the write operations, and these bits will be cleared during the next write.

Each bit is detailed as follows:

Bit 15	CTFRF: Correct transfer This bit is set by the hardware to indicate that an endpoint successfully completes a transaction; using the DIR and EPT_ID bits, software can identify which endpoint completes the transaction successfully. This bit is read-only.
Bit 14	PMOVEF: Packet memory area over/underrun This bit is set by hardware if the microcontroller cannot respond to an USB memory request in time for a long while. The USB peripheral handles this event in the following way: During reception, an ACK handshake packet is not sent; during transmission, a bit-stuff error is forced on the transmitted stream; in both cases, the host will retry the transaction. The PMOVERF interrupt never occurs during normal operations. Since the failed transaction will be retried by the host, the application software has the chance to speed-up device operations during this interrupt handling, to be ready for the next transaction retry; however, this interrupt is not generated during isochronous transfers ("retry" is not supported in isochronous transaction), so this may lead to a loss of data. This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 13	ERRF: Error This flag is set by hardware whenever one of the errors listed below occurs. NANS: No answer. The timeout for a host response expires. CRC: Cyclic redundancy check error. One of the received CRCs in the token or in the data is wrong. BST: Bit stuffing error. A bit stuffing error is detected in PID, data, and/or CRC. FVIO: Framing format violation. A non-standard frame is received (e.g. EOP not in the right place, wrong token sequence, etc.). The USB software can usually ignore these errors, since the USB peripheral and the PC host will activate retransmission in case of errors. This interrupt can be useful at the software development phase, to monitor the quality of transmission over the USB bus and to flag possible errors to users (e.g. loose connector, serious noisy environment, broken conductor in the USB cable, and so on). This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 12	WKUPF: Wakeup This bit is set by the hardware when wakeup signal is detected in suspend mode. This event clears the LPWR bit in the CTRL register and activates the USB_WAKEUP line, notifying the rest parts of the device (e.g. wakeup unit) of the start of the resume process. This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 11	SUSPF: Suspend mode request This bit is set by the hardware when no traffic is received for 3 ms, indicating a suspend mode request from the USB bus. The suspend condition check is enabled immediately after any USB reset, and it is disabled by the hardware in the suspend mode (FSUSP = 1) until the end of a resume sequence. This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 10	RSTF: USB reset request This bit is set when the USB peripheral detects USB reset signal input. The USB peripheral resets its internal protocol state machine and responds to the signal by generating a reset interrupt if interrupt is enabled. Reception and transmission on USB are disabled until this bit is cleared. All configuration registers are not reset, unless the application clears these registers; this ensures that the USB transaction can still be successfully completed after the reset. The device addresses and endpoint registers, however, are reset by an USB reset event. This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 9	SOFF: Start of frame This bit signals the beginning of a new USB frame, and it is set by hardware when a SOF packet on the USB bus is detected by the USB peripheral. The interrupt service routine can monitor the SOF events to complete 1 ms synchronization with the host and to safely read the updated content after the register receives SOF packet (This could be very useful for isochronous transfers). This bit is r/w, but only '0' can be written; writing '1' has no effect.

Bit 8	ESOF: Expected start of frame This bit is set by the hardware when an SOF packet is expected but not received. The host sends an SOF packet each ms, but if the USB peripheral does not receive it properly, the suspend timer will issue this interrupt. If three consecutive ESOF interrupts are generated, i.e. three SOF packets are lost consecutively, a SUSPF interrupt will be generated. Even if missing SOF packets occur while the suspend timer is not yet locked, this bit is still set. This bit is r/w, but only '0' can be written; writing '1' has no effect.
Bit 7:5	Reserved
Bit 4	DIR: Direction of transaction This bit is written by the hardware according to the transaction direction after the transaction completion generates interrupt request. If the DIR bit = 0, the CTFR_TX bit related to the interrupting endpoint is set, indicating the completion of an IN transaction (data transmitted by the USB peripheral to the host PC). If the DIR bit = 1, the CTFR_RX bit related to the interrupting endpoint is set, indicating the completion of an OUT transaction (data received by the USB peripheral from the host PC). If the CTFR_TX bit is set at the same time, it indicates two pending transactions (IN and OUT) are waiting to be processed. This information can be used by the application software to access the USB_EPTN bits related to the triggering transaction since it represents the direction of the pending interrupt. This bit is read-only.
Bit 3:0	EPT_ID[3:0]: Endpoint identifier These bits are written by the hardware according to the endpoint of the interrupt request after the transaction completion generates interrupt request. If several endpoint transactions are pending at the same time, the hardware writes the endpoint with the highest priority. The priority is defined in the following way: Isochronous and double-buffered bulk endpoints are considered the highest priority, and other endpoints are lower priority. If several endpoints with the same priority are requesting an interrupt, then the priority is determined by the number of the endpoint; Endpoint 0 has the highest priority, i.e. the smaller the number, the higher the priority. The application software can order the endpoint requests according to this priority scheme. These bits are read-only.

21.4.1.3 USB Frame Number Register (USB_FRNUM)

Address offset: 0x48

Reset value: 0x0XXX, where X is an undefined value.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPSTS	DMSTS	LCK	LSOF[1:0]	FRNUM[10:0]											
r	r	r	r										r		

Bit 15	DPSTS: Receive data + line status This bit can be used to observe the received data plus line status and to monitor the occurrence of wakeup event at pending state.
Bit 14	DMSTS: Receive data - line status This bit can be used to observe the received data minus line status and to monitor the occurrence of wakeup event at pending state.
Bit 13	LCK: Locked This bit is set by the hardware when at least two consecutive SOF packets are received after the end of an USB reset condition or after the end of an USB resume sequence. Once locked, the frame timer stops counting, and it resumes counting after an USB reset or USB suspend event.
Bit 12:11	LSOF[1:0]: Lost SOF When an ESOF event occurs, these bits are written by the hardware with the number of lost SOF packets. At the reception of another SOF packet, these bits are cleared.
Bit 10:0	FRNUM[10:0]: Frame number This bit field contains the 11-bit frame number in the last received SOF packet. The frame number is incremented at each frame sent by the host, and this is very useful for isochronous transfers. This bit field is updated at the generation of an SOF interrupt.

21.4.1.4 USB Device Address Register (USB_DEVADR)

Address offset: 0x4C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
Reserved										EN	ADR[6:0]									
res										rw	rw									
Bit 7	EN: Enable function This bit is set by the software to enable the USB device. If this bit is '0', the USB peripheral stops working, ignores the settings of all the registers, and do not respond to any USB communication.																			
	ADR[6:0]: Device address These bits contain the USB function address assigned by the host during the enumeration process. Both this field and the endpoint address (EPTADR) field must match with the address information contained in USB token, to handle a transaction to the required endpoint successfully.																			

21.4.1.5 USB Packet Buffer Description Table Address Register (USB_BUFTBL)

Address offset: 0x50

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
ADR[15:3]														Reserved			
rw														res			
Bit 15:3	ADR[15:3]: Buffer table address These bits contain the start addresses of the buffer description table inside the dedicated packet memory. This table describes the buffer location and the size of each endpoint, and it must be aligned to an 8-byte boundary (the 3 least significant bits are always 000). At the beginning of every transaction, the USB peripheral reads the table related to the addressed endpoint, to obtain its buffer start location and the buffer size.																
	Bit 2:0 Reserved. Forced to be '0' by hardware.																

21.4.2 Endpoint Registers

The number of these registers is determined by the number of endpoints supported by the USB peripheral. The USB peripheral supports up to 8 bidirectional endpoints. Each USB device must support a control endpoint, of which address (the EPTADR bit) must be set to 0. Different endpoints must use different numbers; otherwise, endpoints will behave in an undefined way. For each endpoint, there is a corresponding USB_EPTn register available to store the endpoint specific information.

21.4.2.1 USB Endpoint n Register (USB_EPTn), n = [0...7]

Address offset: 0x00 ~ 0x1C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTFR_RX	DTOG_RX	STS_RX[1:0]	SETUP	EPT_TP YE[1:0]	EPT_SUBTYPE	CTFR_TX	DTOG_TX	STS_TX[1:0]	EPTADR[3:0]						
rc_w0	t	t	t	rw	rw	rc_w0	t	t	rw						

When the USB peripheral receives reset signal from the USB bus, or the FRST bit in the CTR register is set, the USB peripheral will perform a reset. All the bits are reset except for the CTFR_RX and CTFR_TX bits, which are kept unchanged to handle the following transactions. Each endpoint has its

corresponding USB_EPTn register, where n is the endpoint identifier.

Read-modify-write cycles on these registers should be avoided because some bits may be set by the hardware between the read and the write operations; the next write will then modify these bits again, causing the application to miss the related operations. Therefore, all these bits have an ‘invariant’ value that must be used whenever their modification is not required. It is recommended that users modify these registers with a load instruction to prevent the application from modifying bits that do not need to be modified.

Bit 15	CTFR_RX: Correct transfer for reception This bit is set by the hardware when an OUT/SETUP transaction is successfully completed; software can only clear this bit. If the CTFR_IEN bit is set accordingly, a related interrupt is generated. The type of the transaction, OUT or SETUP, can be determined from the SETUP bit described below. A transaction ended with a NAK or STALL handshake does not set this bit, since no data is actually transferred. This bit is r/w, but only 0 can be written; writing 1 has no effect.
Bit 14	DTOG_RX: Data toggle for reception transfers If the endpoint is not isochronous, this bit is set by hardware to indicate the expected value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be received. After receiving correct data packet of PID, the USB peripheral sends the ACK handshake and toggles this bit; if the endpoint is defined as a control one, hardware clears this bit at the reception of a SETUP packet. If the endpoint is double-buffered, this bit is also used to support the double buffer swapping. If the endpoint is isochronous, since only DATA0 is sent, this bit is used only to support double buffer swapping without toggling. Hardware sets this bit right after the end of data packet reception, since no handshake is used for isochronous transfers. Software can initialize this bit (initialization is necessary when the endpoint is not a control one) or toggle this bit for special usages. This bit is r/w, but writing 0 has not effect; writing 1 toggles this bit.
Bit 13:12	STS_RX[1:0]: Status bits for reception transfers This bit is used to indicate the current status of the endpoint. After a correct OUT or SETUP transaction is completed (CTFR_RX=1), hardware will automatically set this bit at NAK state, allowing application to have enough time to answer the next data packet after finishing the current transfer. Double-buffered bulk endpoints implement a special transaction flow control, which controls the transfer status based upon the buffer availability condition. If the endpoint is defined as isochronous, its status can be only “VALID” or “DISABLED”, so the hardware cannot change its status after a successful transaction. If the software sets these bits to “STALL” or “NAK”, the USB peripheral’s responding behavior is undefined. This bit is r/w, but writing 0 has not effect; writing 1 toggles these bits.
Bit 11	SETUP: Setup transaction completed This bit is set by the hardware when the USB peripheral receives a successful SETUP transaction. This bit is used only by control endpoints. It must be examined by the application after a successful receive transaction (CTFR_RX = 1) to determine whether this transaction is the SETUP type. To prevent next incoming tokens from modifying this bit while the interrupt service routine is handling SETUP transaction, this bit is kept frozen while CTFR_RX is at 1; it can only be changed when CTFR_RX is at 0. This bit is read-only.
Bit 10:9	EPT_TYPE[1:0]: Endpoint type These bits are used to configure the current type of endpoint. All USB devices must have at least one control endpoint of which address is 0; there may be other control endpoints if required. Only control endpoints can handle SETUP transactions, which are ignored by endpoints of other kinds. SETUP transactions cannot be answered with NAK or STALL. If a control endpoint is defined as NAK when a SETUP transaction is received, the USB peripheral will not answer, generating a receive error. If the control endpoint is defined as STALL, then the SETUP packet will be accepted successfully, transferring data and issuing a CTR interrupt correctly. An OUT transaction of a control endpoint is handled in the same way as the normal endpoint. Bulk and interrupt endpoints have very similar behaviors, and they differ only in the handling of the EPT_SUBTYPE bit.

Bit 8	EPT_SUBTYPE: Endpoint kind The definition of this bit depends on the endpoint type configured by the EPT_TYPE bits. DBL_BUF: This bit is set by the software to enable the double-buffering feature for bulk endpoint. STATUS_OUT: This bit is set by the software to indicate that the USB peripheral is expecting a status of OUT transaction from the host; in this case, all OUT transactions, of which length is non-zero, will be answered "STALL" by the device. This function is only used for control endpoints; it is useful for the application to detect protocol errors. When STATUS_OUT is cleared, OUT transactions can contain data with any lengths, as required.
Bit 7	CTFR_TX: Correct transfer for transmission This bit is set by the hardware when an IN transaction is successfully completed; corresponding interrupts are generated if the CTFR_IEN bit is set. Software needs to clear this bit after this event is handled. At the end of IN transaction, if the host answers with a NAK or STALL handshake, then this bit will not be set, since no data is actually transferred. This bit is r/w, but only 0 can be written; writing 1 has no effect.
Bit 6	DTOG_TX: Data Toggle for transmission transfers If the endpoint is non-isochronous, this bit contains the required value of the data toggle bit (0 = DATA0, 1 = DATA1) for the next data packet to be transmitted. After a successful data packet transmission, if the USB peripheral receives the ACK handshake from the host, this bit will be toggled. If the endpoint is defined as a control one, this bit is set at the reception of a correct SETUP PID. If the endpoint is double-buffered, this bit is also used to support packet buffer swapping. If the endpoint is isochronous, since only DATA0 is sent, this bit is used only to support double buffer swapping. Hardware sets this bit right after the end of data packet reception, since no handshake is used for isochronous transfers. Software can initialize this bit (initialization is necessary when the endpoint is not a control one) or configure this bit for special usages. This bit is r/w, but writing 0 has no effect; writing 1 toggles this bit.
Bit 5:4	STS_RX[1:0]: Status bits for transmission transfers This bit is used to indicate the current status of the endpoint; Table 21-8 lists all the status. The application can toggle these bits to initialize the status information. After a correct IN transaction is completed (CTFR_TX=1), hardware will automatically set this bit at NAK state, allowing application to have enough time for the following data transfer. Double-buffered bulk endpoints implement a special transaction flow control, which controls the transfer status based upon the buffer availability condition. If the endpoint is defined as isochronous, its status can only be "VALID" or "DISABLED", so the hardware cannot change its status after a successful transaction. If the software sets these bits to "STALL" or "NAK", the USB peripheral's responding behavior is undefined. This bit is r/w, but writing 0 has not effect; writing 1 toggles these bits.
Bit 3:0	EPTADDR[3:0]: Endpoint address Software must write these 4 bits; before enabling the corresponding endpoint, its address must be defined.

Table 21-5 Reception Status Encoding

STS_RX[1:0]	Description
00	DISABLED: All reception requests addressed to this endpoint are ignored.
01	STALL: All reception requests result in a STALL handshake.
10	NAK: All reception requests result in a NAK handshake.
11	VALID: This endpoint is enabled for reception.

Table 21-6 Endpoint Type Encoding

EPT_TYPE[1:0]	Description
00	BULK: Bulk endpoint
01	CTRL: Control endpoint
10	ISO: Isochronous endpoint
11	INT: Interrupt endpoint

Table 21-7 Endpoint Kind Meaning

EPT_TYPE[1:0]		EPT_SUBTYPE Definition
00	BULK	DBL_BUF: Double-buffer endpoint
01	CTRL	STATUS_OUT
10	ISO	Unused
11	INT	Unused

Table 21-8 Transmission Status Encoding

STS_TX[1:0]	Description
00	DISABLED: All transmission requests addressed to this endpoint are ignored.
01	STALL: All transmission requests result in a STALL handshake.
10	NAK: All transmission requests result in a NAK handshake.
11	VALID: This endpoint is enabled for transmission.

21.4.3 Buffer Description Table Registers

Although the buffer description table is located inside the packet buffer memory, it can be regarded as additional registers used to configure the location and size of the packet buffers shared by the USB peripheral and the microcontroller core. Since APB1 bus is addressed by 32 bits, all packet memory locations are accessed by 32-bit aligned address, instead of the actual memory location address of the USB_BUFTBL register and the buffer description table.

In the following sections, two location addresses are introduced: The one to be used by application software while accessing the packet memory, and the other is the local one associated with the USB peripheral access. To obtain the correct memory address in the microcontroller, the packet memory address used by the application must be multiplied by 2. The first packet memory location is 0x4000 6000/0x4000 7800. The buffer description table entry associated with the USB_EPTn registers is described below.

21.4.3.1 Transmission Buffer Address Register n (USB_ADRn_TX)

Address offset: [USB_BUFTBL] x 2 + n x 16

USB local address: [USB_BUFTBL] + n x 8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADRn_TX[15:1]														Reserved	
rw														res	
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Bit 15:1</td> <td style="padding: 2px;">ADRn_TX[15:1]: Transmission buffer address These bits point to the starting address of the packet buffer containing data to be transmitted for the next IN transaction.</td> </tr> <tr> <td style="padding: 2px;">Bit 0</td> <td style="padding: 2px;">Must always be written as '0' since all packet buffers must be word-aligned.</td> </tr> </table>		Bit 15:1	ADRn_TX[15:1]: Transmission buffer address These bits point to the starting address of the packet buffer containing data to be transmitted for the next IN transaction.	Bit 0	Must always be written as '0' since all packet buffers must be word-aligned.										
Bit 15:1	ADRn_TX[15:1]: Transmission buffer address These bits point to the starting address of the packet buffer containing data to be transmitted for the next IN transaction.														
Bit 0	Must always be written as '0' since all packet buffers must be word-aligned.														

21.4.3.2 Transmission Byte Count Register n (USB_CNTn_TX)

Address offset: [USB_BUFTBL] x 2 + n x 16 + 4

USB local address: [USB_BUFTBL] + n x 8 + 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								CNTn_TX[9:0]							
res														rw	
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">Bit 15:10</td> <td style="padding: 2px;">These bits are ignored by the USB peripheral since packet size is limited by USB specifications to 1023 bytes.</td> </tr> </table>		Bit 15:10	These bits are ignored by the USB peripheral since packet size is limited by USB specifications to 1023 bytes.												
Bit 15:10	These bits are ignored by the USB peripheral since packet size is limited by USB specifications to 1023 bytes.														

Bit 9:0	CNTn_TX[9:0]: Transmission byte count These bits contain the number of bytes to be transmitted for the next IN transaction.
---------	---

Note: Double-buffered and isochronous IN endpoints have two USB_CNTn_Tx registers: USB_CNTn_TX_1 and USB_CNTn_TX_0.

21.4.3.3 Reception Buffer Address Register n (USB_ADRn_RX)

Address offset: [USB_BUFTBL] x 2 + n x 16 + 8

USB local address: [USB_BUFTBL] + n x 8 + 4

ADRn_RX[15:1]														Reserved	
rw														res	

ADRn_RX[15:1]														Reserved	
rw														res	

ADRn_RX[15:1]														Reserved	
Bit 15:1														res	

Bit 15:1														res	
Bit 0														This bit must always be written as '0' since all packet buffers must be word-aligned.	

21.4.3.4 Reception Byte Count Register n (USB_CNTn_RX)

Address offset: [USB_BTABLE] x 2 + n x 16 + 12

USB local address: [USB_BTABLE] + n x 8 + 6

BLKSIZE					NUM_BLK[4:0]				CNTn_RX[9:0]						
rw					rw				rw						

This register is used to store two different values, both required during packet reception. The most significant 6 bits define the reception buffer size to allow buffer overflow detection; while the least significant 10 bits are used by the USB peripheral to give the actual number of received bytes. Due to the restrictions on the number of available bits, buffer size is represented by the number of allocated memory blocks, and the block size is determined by the required buffer size. The size of allocated buffer is defined during the enumeration process according to the maxPacketSize parameter value of the endpoint description (Please refer to “Universal Serial Bus 2.0 Specification” for more details).

Bit 15														BLKSIZE: Block size This bit defines the size of memory block used to determine the allocated buffer area. If BLKSIZE = 0, the size of memory block is 2 bytes, so the allocated buffer size ranges from 2 to 62 bytes. If BLKSIZE = 1, the size of memory block is 32 bytes, which allows to reach the maximum packet length defined by USB specifications, so the allocated buffer size ranges from 32 to 512/768 bytes, which is the maximum packet size allowed by USB standard specifications.	
Bit 14:10														NUM_BLK[4:0]: Number of blocks These bits define the number of memory blocks allocated, which then determining the size of the packet buffer finally being used.	
Bit 9:0														CNTn_RX[9:0]: Reception byte count These bits are written by the USB peripheral to record the number of bytes received by the endpoint during the last OUT/SETUP transaction.	

Note: Double-buffered and isochronous IN endpoints have two USB_CNTn_RX registers: USB_CNTn_RX_1 and USB_CNTn_RX_0.

Table 21-9 Definition of Allocated Buffer Memory

NUM_BLK[4:0] Value	Memory allocated when BLKSIZE = 0	Memory allocated when BLKSIZE = 1
00000	Not allowed	32 bytes
00001	2 bytes	64 bytes
00010	4 bytes	96 bytes
00011	6 bytes	128 bytes
...
01111	30 bytes	512 bytes
10000	32 bytes	544 bytes ^(Note)
10001	34 bytes	576 bytes ^(Note)
10010	36 bytes	608 bytes ^(Note)
...
10111	46 bytes	768 bytes ^(Note)
11000	48 bytes	Reserved
...
11110	60 bytes	Reserved
11111	62 bytes	Reserved

Note: When USB768B = 1 in the RCC_MISC register, the packet buffer can be set to the maximum of 768 bytes; when USB768B = 0, the size of packet buffer should be restricted within 512 bytes; areas that exceed 512 bytes are reserved.

22 MCU Debug (MCUDBG)

22.1 Introduction

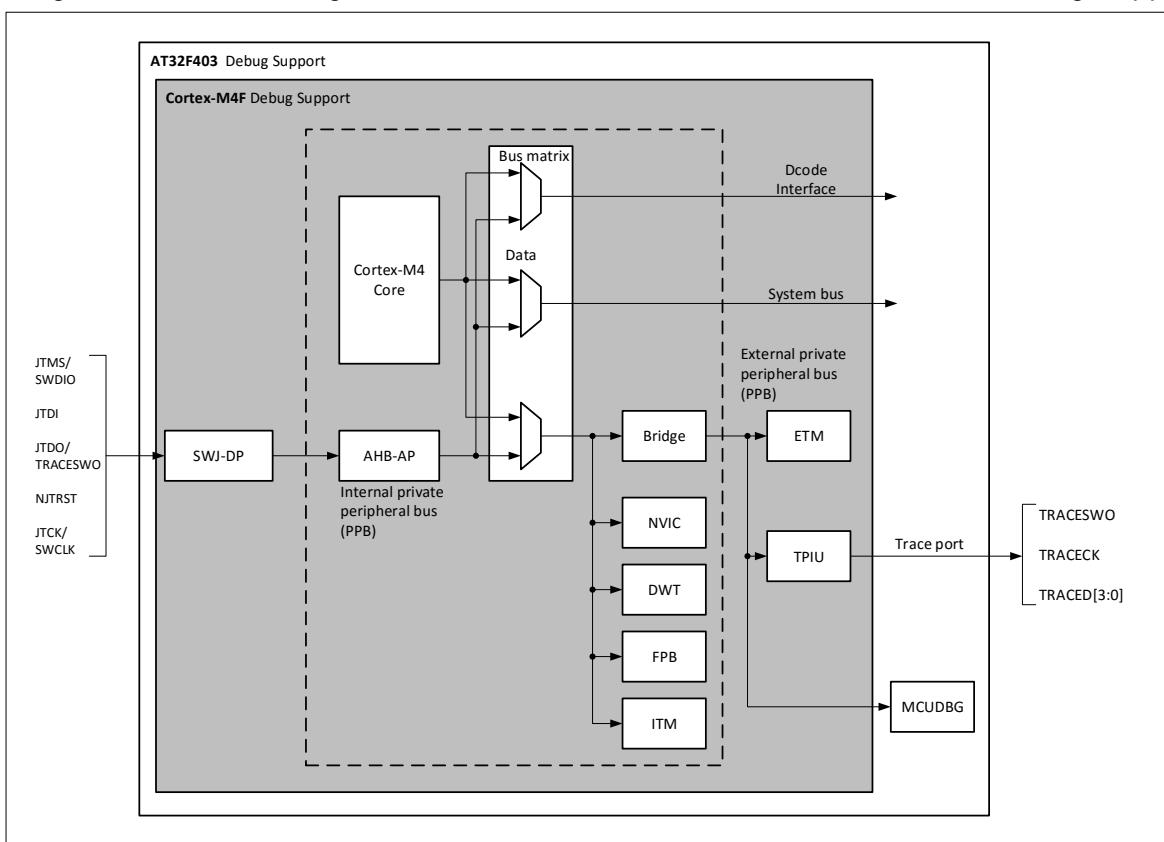
AT32F403 are built with the Cortex®-M4F core, which contains hardware extensions for advanced debugging features. The debug extensions allow the core to be stopped either on a given instruction fetch (breakpoint) or data access (watchpoint). When stopped, the core's internal state and the system's external state can be examined. Once the examination is completed, the core and the system can be restored, and the program execution will resume.

When the AT32F403 microcontroller is connected to the debugger and begins to debug, the debugger implements debugging operation with the hardware debug module.

Two interfaces for debug are available:

- Serial wire
- JTAG debug port

Figure 22-1 Block Diagram of AT32F403-Level and Cortex®-M4F-Level Debug Support



The ARM Cortex®-M4F core provides integrated on-chip debug support. Please refer to the following:

- Cortex®-M4 Technical Reference Manual (TRM)
- ARM Debug Interface V5
- ARM CoreSight Design Kit (Revision r1p0) Technical Reference Manual

MCUDBG module component helps the debugger to debug low-power mode, timer, I²C, bxCAN, WWDG, and IWDG. When the corresponding bits are set, MCUDBG provides clocks or keeps counter timers, WWDG, IWDG, CAN, or I²C, at the current state in low-power mode. The MCU debug component helps the debugger provide supports for:

- Low-power mode
- Clock control for timers, watchdog, I²C, and bxCAN during a breakpoint
- ID code
- Control of trace pin assignment

22.2 Function Overview

22.2.1 Debug Support for Low-power Mode

To enter low-power mode, the instruction of WFI and WFE are executed. MCU supports several low-power modes, which can either deactivate the CPU clock or reduce the power consumption of the CPU. The core does not allow FCLK or HCLK to be disabled during a debug session. Since these clocks are required for the debug operation, they must remain active during a debug. MCU integrates a special way to allow users to debug software in low-power modes.

For this purpose, the debugger must first set some debug configuration registers to change the behavior of low-power modes:

- In Sleep mode, the DBG_SLEEP bit of the MCUDBG_CTRL register must be previously set by the debugger. This will feed HCLK with the same clock provided for FCLK (system clock previously configured by the software).
- In Stop mode, the DBG_STOP bit must be previously set by the debugger. This will enable the internal RC oscillator clock to feed FCLK and HCLK in Stop mode.

22.2.2 Debug Support for Timer, Watchdog, bxCAN, and I²C

During a breakpoint, it is necessary to select the operation mode for counters according to the different usages of timers and watchdogs:

- During a breakpoint, counters continue to count. This is usually required when an output PWM is controlling a motor.
- During a breakpoint, counters stop counting. This is required for the counters of watchdogs.

For bxCAN, users can choose to block the update of the receive register during a breakpoint.

For I²C, users can choose to block the SMBUS timeout during a breakpoint.

22.2.3 ID Code

There are several ID codes inside the AT32F403 microcontroller. It is strongly recommended that tool designers use the MCUDEVICEID mapped in the external PPB memory at address 0xE0042000 to lock their debuggers. This ID identifies MCU's part number and the die revision. It is part of the MCUDBG component and is mapped on the external PPB bus.

This code is accessible by using the JTAG debug port (4 ~ 5 pins), or the SW debug port (2 pins), or by the user software. It is even accessible while MCU is under system reset.

Only the DEV_ID[11:0] should be used for identification by the debugger/programmer tools.

22.2.4 SWJ Debug Port Pins

Five general-purpose I/O ports of AT32F403 can be used as SWJ-DP port pins. These pins are available on all packages. After reset (SYSRESETn or PORESETn), all five pins used for the SWJ-DP are immediately initialized and assigned as dedicated pins for the debugger.

AT32F403 microcontroller implements the AF remap and debug I/O configuration register (AFIO_MAPR) (Please refer to [Section 7.5.9](#)) to disable some parts of the SWJ-DP port or all functions of the pins; these dedicated pins are released for general-purpose I/O usages. Register programming is done by the user software program instead of the debugger host. Please refer to [Section 7.4.4](#).

22.2.5 Trace Pin Assignment

By default, trace pins are not dedicated pins. They can be assigned by setting the TRACE_IOEN and TRACE_MODE bits in the MCUDBG_CTRL register. Trace pins are not assigned by default, and the configuration should be done by the debugger host.

The register is mapped on the external PPB and is reset by PORESET (and not by system reset). It can be written by the debugger under system reset.

Table 22-1 Flexible Trace Pin Assignment

DBGMCU_CTRL Register		Pin Assigned For	Trace Pin Assignment					
TRACE_IOEN	TRACE_MODE[1:0]		PB3/JTDO/TR ACESWO	PE2/TRA CECK	PE3/TRA CED[0]	PE4/TRA CED[1]	PE5/TRA CED[2]	PE6/TRA CED[3]

0	XX	No trace (By default)	Released ^(Note)	Released (Used as GPIOs)				
1	00	Asynchronous trace	TRACES WO					
1	01	Synchronous trace 1 bit		TRAC ECK	TRAC ED[0]	Released (Used as GPIOs)		
1	10	Synchronous trace 2 bits		TRAC ECK	TRAC ED[0]	TRAC ED[1]	Released (Used as GPIOs)	
1	11	Synchronous trace 4 bits		TRACE CK	TRACE D[0]	TRACE D[1]	TRACE D[2]	TRACE D[3]
Note: When serial wire mode is used, this pin is released; when JTAG is used, it is assigned as JTDO.								

22.3 MCUDBG Registers

Table 24-2 lists MCUDBG register map and reset values. These peripheral registers must be accessed by words (32-bit).

Table 24-2 MCUDBG Register Address Map and Reset Value

Address	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE004 2000																																	
0xE004 2004	MCUDBG_CTRL	DBG_I2C3_SMBUS_TIMEOUT																															
	Reset Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

22.3.1 MCUDBG Control Register (MCUDBG_CTRL)

The MCUDBG_CTRL register is mapped on the external PPB bus at base address, 0xE0042000. It is asynchronously reset by PORESET (and not by system reset). It can be written by the debugger when the core is at reset state.

Address: 0xE0042004 (Accessible only for 32-bit accesses)

POR reset: 0x00000000 (Not reset by system reset)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBG_I2C3_SMBUS_TIMEOUT	DBG_TMR1_1_STOP	DBG_TMR1_0_STOP	DBG_TMR9_STOP	DBG_TMR1_4_STOP	DBG_TMR1_3_STOP	DBG_TMR1_2_STOP	Reserved	DBG_TMR1_5_STOP	Reserved	DBG_TMR7_STOP	DBG_TMR6_STOP	DBG_TMR5_STOP	DBG_TMR8_STOP	DBG_I2C2_SMBUS_TIMEOUT	
rw	rw	rw	rw	rw	rw	rw	res	rw	res	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_I2C1_SMBUS_TIMEOUT	DBG_CAN1_STOP	DBG_TMR4_STOP	DBG_TMR3_STOP	DBG_TMR2_STOP	DBG_TMR1_STOP	DBG_WWDG_STOP	DBG_IWDG_STOP	TRACE_MODE[1:0]	TRAC_E_IOEN	Reserved	DBG_STANDBY	DBG_STOP	DBG_SLEEP		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	res	rw	rw	rw		

Bit 31	DBG_I2C3_SMBUS_TIMEOUT: SMBUS timeout mode stops when the core is halted. 0: The same behavior as in the normal mode 1: The SMBUS timeout is frozen.
Bit 24:23 Bit 21	Reserved. Must be kept as 0.
Bit 30:25 Bit 22 Bit 20:17	DBG_TMRx_STOP: Timer counter stops when the core is halted. (x = 15...5) 0: The clock of the involved timer counter is fed even if the core is halted, and the outputs behave normally. 1: The clock of the involved timer counter stops when the core is halted, and the outputs are disabled (just like an emergency stop in response to a break event).
Bit 16	DBG_I2C2_SMBUS_TIMEOUT: SMBUS timeout mode stops when the core is halted. 0: The same behavior as in the normal mode 1: The SMBUS timeout is frozen.
Bit 15	DBG_I2C1_SMBUS_TIMEOUT: SMBUS timeout mode stops when the core is halted. 0: The same behavior as in the normal mode 1: The SMBUS timeout is frozen.
Bit 14	DBG_CAN1_STOP: CAN1 stops when the core enters debug state. 0: The same behavior as in the normal mode 1: The receive registers of CAN1 are disabled.
Bit 13:10	DBG_TMRx_STOP: Counter stops when the core enters debug state. (x = 4...1) 0: The counter of the involved timer still works normally. 1: The counter of the involved timer stops.
Bit 9	DBG_WWDG_STOP: Debug window watchdog stops when the core enters debug state. 0: The window watchdog counter still works normally. 1: The window watchdog counter stops.
Bit 8	DBG_IWDG_STOP: Watchdog stops when the core enters debug state. 0: The watchdog counter still works normally. 1: The watchdog counter stops.

Bit 7:5	TRACE_MODE[1:0] and TRACE_IOEN : Trace pin assignment control - When TRACE_IOEN = 0: TRACE_MODE = xx: Trace pins are not assigned (Default state). - When TRACE_IOEN = 1: TRACE_MODE = 00: Trace pin assignment for asynchronous mode TRACE_MODE = 01: Trace pin assignment for synchronous mode, with data size of 1 TRACE_MODE = 10: Trace pin assignment for synchronous mode, with data size of 2 TRACE_MODE = 11: Trace pin assignment for synchronous mode, with data size of 4
Bit 4:3	Reserved. Must be kept as 0.
Bit 2	DBG_STANDBY : Debug Standby mode 0: (FCLK is OFF and HCLK is OFF) The whole digital part is unpowered. From the software point of view, exiting the Standby mode is the same as reset (except for a few status bits, which indicate that MCU just exits from Standby mode). 1: (FCLK is ON and HCLK is ON) In this case, the digital part is not unpowered, and FCLK and HCLK are clocked by the internal RLD oscillator. In addition, MCU generates a system reset to exit Standby mode, which is the same as reset.
Bit 1	DBG_STOP : Debug Stop mode 0: (FCLK is OFF and HCLK is OFF) In Stop mode, the clock controller disables all clocks (including HCLK and FCLK). When exiting the STOP mode, the clock configuration is identical to the one after reset (MCU is clocked by the 8 MHz internal RC oscillator (HSI)). Consequently, the software must reprogram the clock controller to enable PLL, Xtal, etc. 1: (FCLK is ON and HCLK is ON) In Stop mode, FCLK and HCLK are clocked by the internal RC oscillator. When exiting the Stop mode, the software must reprogram the clock controller to enable PLL, Xtal, etc. (in the same way to configure this bit when it is 0).
Bit 0	DBG_SLEEP : Debug Sleep mode 0: (FCLK is ON and HCLK is OFF) In Sleep mode, FCLK is clocked by the system clock as previously configured, while HCLK is disabled. In this case, the clock controller configuration is not reset; consequently, when exiting the Sleep mode, the software does not need to reconfigure the clock controller. 1: (FCLK is ON and HCLK is ON) In Sleep mode, FCLK and HCLK are clocked by the system clock as previously configured by the software.

23 Revision History

Document Revision History

Date	Version	Revision Note
2018.03.19	1.00	Initial release.
2019.04.16	1.01	<ul style="list-style-type: none"> 1. Updated Section 5.4.7: The description of the option byte register, FLASH_UOB 2. Updated Section 7.5.1: The description of MDEy[1:0] (y=0…7) of Port Configuration Register Low. For general-purpose output mode, value of 0b10 is suggested. 3. Updated Section 7.5.5: Port Bit Set/Reset Register (GPIOx_BSRE) is write only. 4. Updated Section 7.5.6: Port Bit Reset Register (IOx_BRE) Bit [15:0] are write only. 5. Updated Chapter 10 (TIMER): Prescaler ratio of all kinds of timers in the function table is 1~65536. 6. Updated Chapter 10 (TIMER): General-purpose timers, TMR9 & TMR10/TMR11 do not support DMA.
2020.08.05	1.02	<ul style="list-style-type: none"> 1. Updated Section 1.2.4 : The main block consists of pages, each of them is 2 KB for 256K and above, and 1KB for 128K and below. Modified the external memory address to 0x08400000-0x093FFFF. 2. Updated the description of bit 30:29, bit 21:18, PLL output x48 and PLL output x64 in Section 3.3.2. 3. Added Flash memory organization of 128K and below in Table 5.1. 4. Modified the description of main flash programming in Section 5.3.2.3. 5. Updated the number of option bytes to 24 in Section 5.3.2.5. 6. Updated the description of Section 5.3.3: The base unit of write protection is two pages for 256K and beyond, and four pages for 128K. 7. Updated the description of Section 6.2.: X32 + X26 + X23 + X22 + X16 + X12 + X11 + X10 + X8 + X7 + X4 + X2 + X +1 is changed to X32 + X26 + X23 + X22 + X16+ X12 + X11 + X10 + X8 + X7 + X5 + X4 + X2 + X +1. 8. Modified Section 10.4.3.19: Slave mode: trigger mode, Write C2P = 0 in the TMRx_CCE register to confirm the polarity (and detect rising edges only). 9. Bit 4 of TMRx_CCE register is changed from C2NEN to C2EN in Section 10.4.4.9. 10. Moved the note in RTC_DIVH to RTC_DIVL in Section 12.4.3. 11. Modified the description of Section 13.3.8: If the DMAEN bit is set, and the DMA controller transfers the converted data of regular group channels to SRAM after each DOR register update. 12. Updated note 2 of Section 13.3.11. Before starting a calibration, the ADC must be at power-on state (ADON bit = '1') for at least two ADC clock cycles. 13. Section 14.5.1: 10 in WAVE2 and WAVE1 bits is changed to 01. 14. Updated the description of bit 3 (ORERR) in Section 16.6.2: USART_CTRL register is changed to USART_DT register.

2021.02.04	1.03	<ol style="list-style-type: none">1. Modified the description of Section 17.3.1.4: In master mode, the communication starts immediately once the SPI is enabled. The current reception will not stop until it is finished when the SPIEN bit is cleared.2. Modified the description of Section 17.3.2.6: Four status flags are provided for users to monitor the state of the I2S bus. Section 17.3.2.7: The bit is cleared by a read operation to the SPI_DT register followed by a read operation to the SPI_STS register.3. Modified the address of option bytes of Section 1.2 to 0xFFFF800 - 0xFFFF82F4. Updated the description of Section 1.3: Boot loader reprograms the Flash memory through USART1, USART2 or USB interface.5. Updated the description of WRP3 in Section 5.3.4.6. Updated the description of bit 11, bit 7 and bit 3 in Section 10.2.4.9.
------------	------	---

IMPORTANT NOTICE – PLEASE READ CAREFULLY

Purchasers are solely responsible for the selection and use of ARTERY's products and services; ARTERY assumes no liability for purchasers' selection or use of the products and the relevant services.

No license, express or implied, to any intellectual property right is granted by ARTERY herein regardless of the existence of any previous representation in any forms. If any part of this document involves third party's products or services, it does NOT imply that ARTERY authorizes the use of the third party's products or services, or permits any of the intellectual property, or guarantees any uses of the third party's products or services or intellectual property in any way.

Except as provided in ARTERY's terms and conditions of sale for such products, ARTERY disclaims any express or implied warranty, relating to use and/or sale of the products, including but not restricted to liability or warranties relating to merchantability, fitness for a particular purpose (based on the corresponding legal situation in any judicial districts), or infringement of any patent, copyright, or other intellectual property right.

ARTERY's products are not designed for the following purposes, and thus not intended for the following uses: (A) Applications that have specific requirements on safety, for example: life-support applications, active implant devices, or systems that have specific requirements on product function safety; (B) Aviation applications; (C) Auto-motive application or environment; (D) Aerospace applications or environment, and/or (E) weapons. Since ARTERY products are not intended for the above-mentioned purposes, if purchasers apply ARTERY products to these purposes, purchasers are solely responsible for any consequences or risks caused, even if any written notice is sent to ARTERY by purchasers; in addition, purchasers are solely responsible for the compliance with all statutory and regulatory requirements regarding these uses.

Any inconsistency of the sold ARTERY products with the statement and/or technical features specification described in this document will immediately cause the invalidity of any warranty granted by ARTERY products or services stated in this document by ARTERY, and ARTERY disclaims any responsibility in any form.

© 2021 ARTERY Technology Company - All Rights Reserved.